U3 Platform 1.0 SDK

Application Deployment Guide

Version 1.0
September 2005
Revision 4.0

DOCUMENT CONTROL INFORMATION

Document No.: 04-UM-0405-00

	Title	Name	Date
Issued by:	U3 Chief Architect	Daniel Goodman	September 2005
Edited by:			

Table of Contents

List of Figures

List of Tables

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, page 3, 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, page 6, 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, page 9, 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, page 13, 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, page 25, 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, page 35, 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, page 46, 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.

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, which is included in the Smart Logo Compliance Kit 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.
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*.

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:

1. 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.
2. 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.
3. 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 user (U3) applications installed on the U3 device are accessed through the U3 Launchpad. The U3 Launchpad 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.

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 Section 7, Manifest File Specification), instruct the U3 Launchpad which commands to execute at each stage of the U3 application’s lifecycle.

Figure 1 illustrates the U3 Launchpad GUI that is displayed after clicking the U3 icon in the system tray.

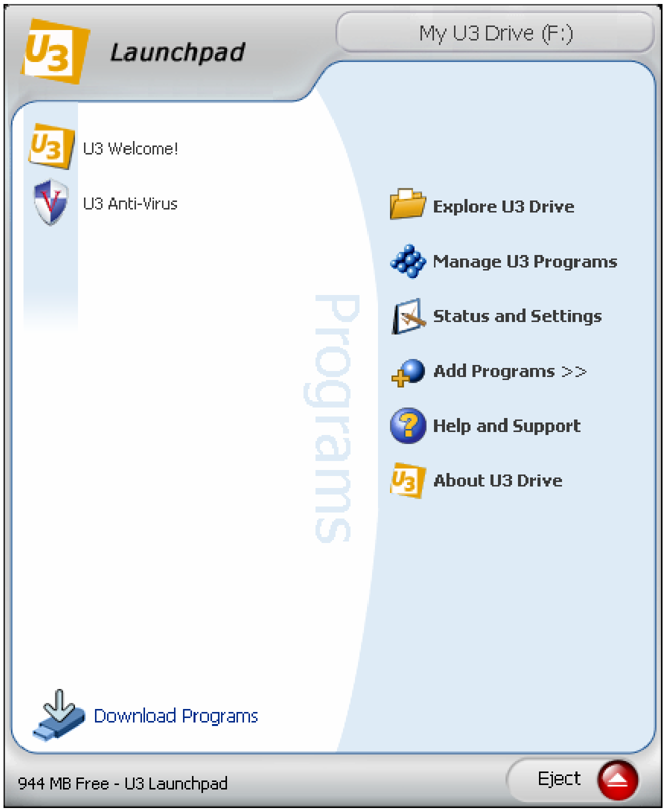


Figure 1: The U3 Launchpad GUI

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 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. Requirements for U3 applications, followed by best-practice recommendations, are covered in Section 4.4, U3 Application Requirements and Section 4.4, U3 Application Recommendations.

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.

Sections 6 to 9 detail the U3 package specification, the U3 package's manifest file containing configuration actions, and the runtime environment that is created by the U3 Launchpad for the U3 application. The worked example that is introduced in Section 5, Worked Example is used to demonstrate a sample implementation.

4.1 U3 Application Lifecycle

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

1. Installing the U3 package on a U3 device
2. Configuring the host machine
3. Starting the U3 application
4. Stopping the U3 application
5. Cleaning up the host machine
6. 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.

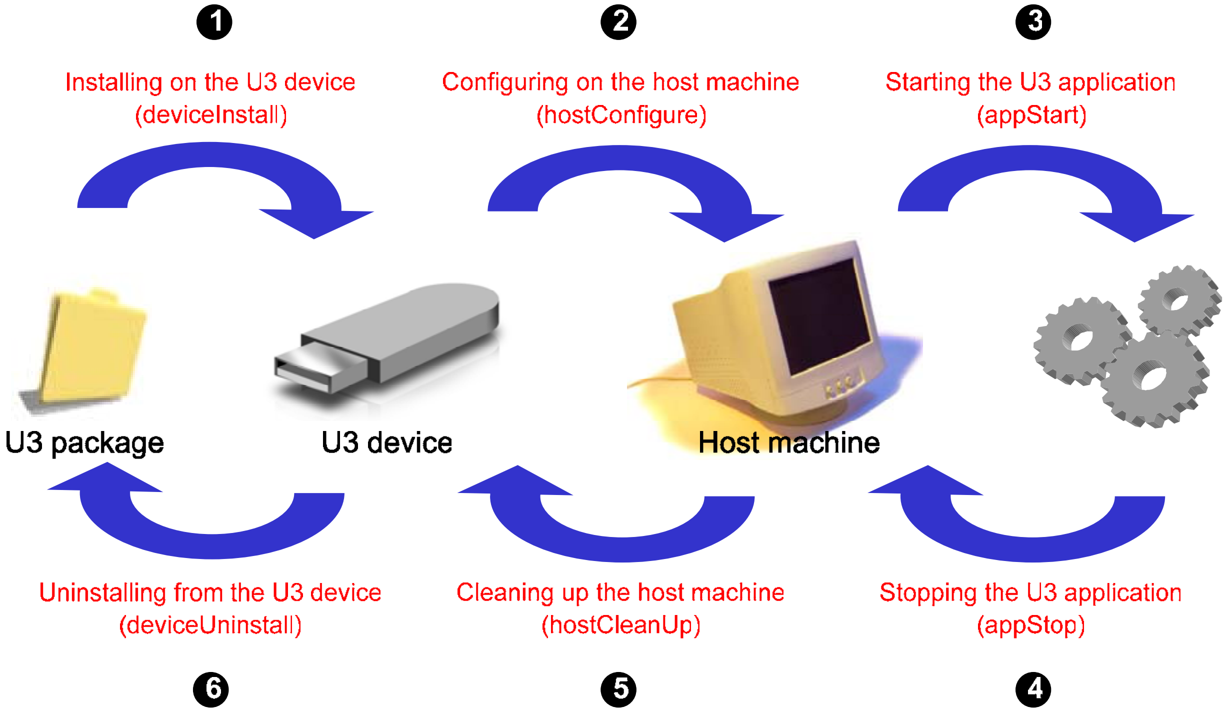

❶
Installing on the U3 device (deviceInstall)
❷
Configuring on the host machine (hostConfigure)
❸
Starting the U3 application (appStart)
U3 package
U3 device
Host machine
Uninstalling from the U3 device (deviceUninstall)
Cleaning up the host machine (hostCleanUp)
Stopping the U3 application (appStop)
❻
❺
❹

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 Section 8, U3 Actions.

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 (such 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.

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.

In the migration analysis of the application in the worked example, each U3 requirement and recommendation, as listed in Section 4.4, U3 Application Requirements and Section 4.4, U3 Application Recommendations will be discussed.

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 that were introduced in Sections 4.4 and 4.4 regarding the worked example. Implementation considerations are described in detail to demonstrate architectural thinking behind U3 applications.

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.**

   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.

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.**

   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

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

   Not applicable.

4. **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.

5. **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.

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

6. **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.

7. **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.

8. **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.

9. **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 Section 7, Manifest File Specification for a detailed specification 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 described in Section 6.4, 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. 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, i.e. files needed by U3 applications to execute. Any files stored in these directories are deleted when the U3 application is uninstalled.

User data files created by the user during application use, such as backups, documents, music, and downloads should not be stored in these directories. Ideally, these files should be stored on the U3 device in a directory created by the U3 application at install time.

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 defined in Section 9.1.3, 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 should never prompt the user to save his data files in any of the U3 runtime directories.

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 5.2, Migration Analysis described the architecture considerations for mobilizing CWmaster. In this section, the reader can track file placement considerations when turning an application into a mobile U3 application. Each U3 application file is placed in one of the three designated directories according to its role. New directories are defined on the U3 device for use by the U3 application, and helper applications are created to perform certain tasks during the U3 application lifecycle stages.

According to the application description in Section 5.1, Application Description, 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.

i.e. 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 (new files)	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 Section 8.2.2, hostConfigure for a more detailed description of the hostConfigure action.

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 (new files)	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 (new files, updated 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 (see Section 8.2.4, appStop) 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:

1. Analyze the software file structure and define which files should be placed in each of the designated directories.
2. 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).
3. 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.
4. 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.
5. 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.
6. 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.

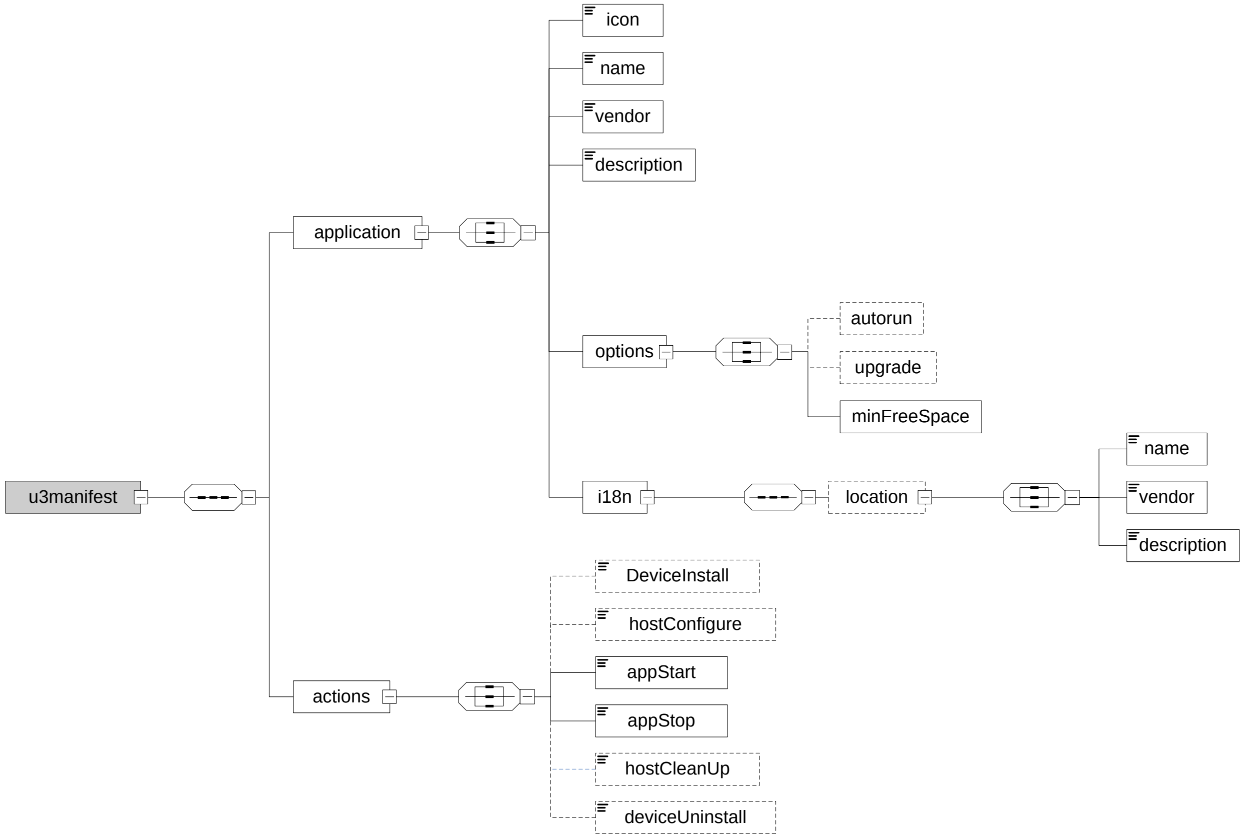


Figure 3: Manifest File Structure

Note: Elements with dashed 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. A full sample can be found in Section 7.4.

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>
```

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. Names longer than 24 characters 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 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\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

The options and their use are detailed in Sections 7.2.5.1 to 7.2.5.3.

7.2.5.1 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"/>
```

7.2.5.2 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/>
```

7.2.5.3 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>
```

7.2.6 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>
```

7.2.6.1 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>
```

7.2.6.1.1 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.

7.2.6.1.2 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.

7.2.6.1.3 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 (see Section 9, U3 Runtime 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

Action	Type	Description
		application. See Section 8.2.3, appStart.
appStop	Required	Executes this command to stop the 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.
deviceUninstall	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%"</hostConfigure>
      <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 detailed in Section 5, 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
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
<?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 defined in Section 9, U3 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.

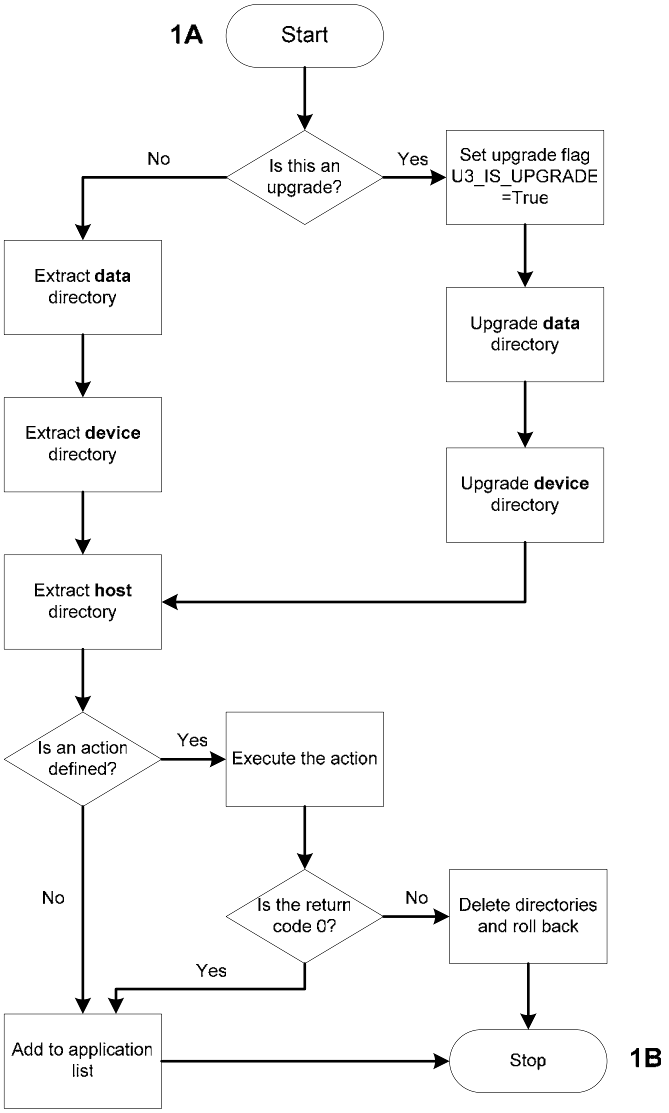


Figure 4: Installing the U3 Application (Stage 1)

Requirements

1. 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.
2. The deviceInstall executable must execute without requiring the hostConfigure action to be executed.

Information

1. The U3 Launchpad extracts the files from the U3 Package to the U3_APP_DATA_PATH and U3_DEVICE_EXEC_PATH directories on the U3 device, and U3 application files are extracted to the U3_HOST_EXEC_PATH temporary directory before deviceInstall is executed.
2. 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 U3_IS_UPGRADE flag is set to true by the U3 Launchpad. See Section 8.2.5, hostCleanUp and Section 8.2.6, deviceUninstall.
3. 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.

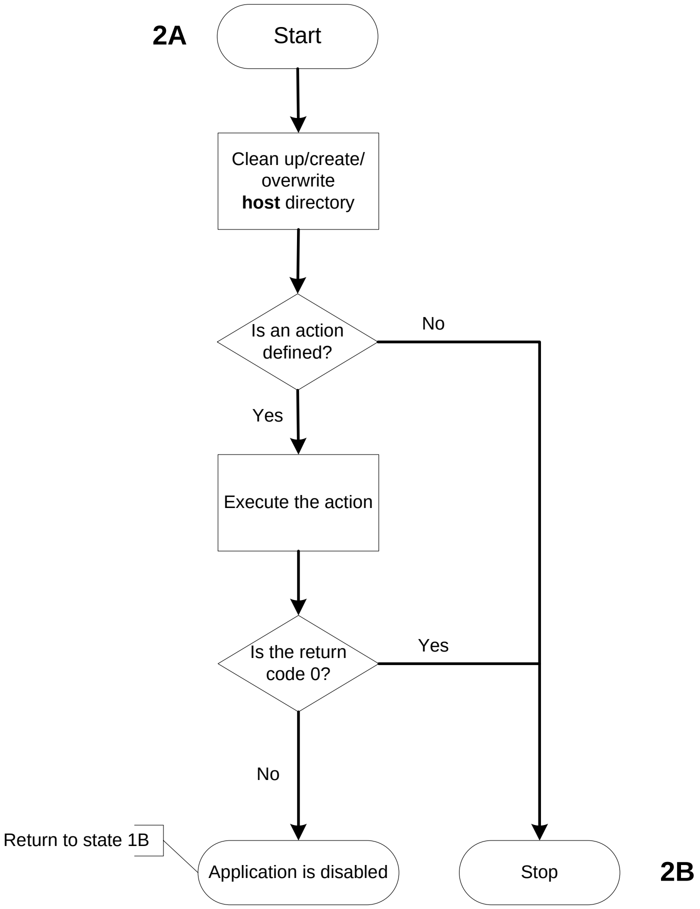


Figure 5: Configuring the Host Machine (Stage 2)

Requirements

1. Any system changes performed during the hostConfigure action must be undone during the hostCleanUp action.
2. 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

1. 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.
2. 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.

3. 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.

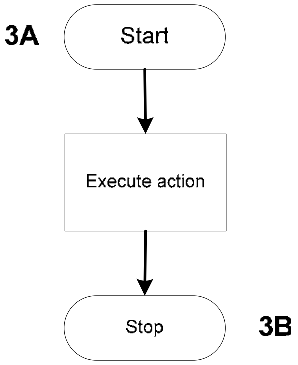


Figure 6: Starting the U3 Application (Stage 3)

Information

1. The U3 Launchpad runs each U3 application in a separate process space.
2. 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.

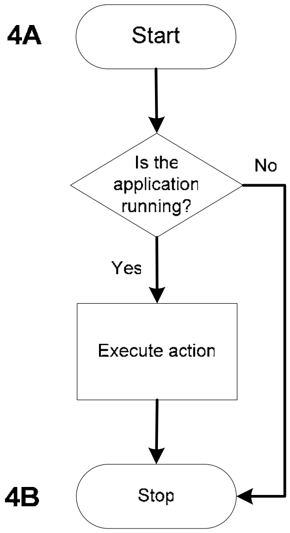


Figure 7: Stopping the U3 Application (Stage 4)

Requirements

1. 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.
2. The appStop executable must be stored in the host exec directory to ensure it is available even if the U3 device is ejected.

Information

1. 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. <u>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.</u>
2. If the U3 device is ejected before appStop is called, the U3_DEVICE_AVAILBLE flag is set to false.

8.2.5 hostCleanUp

Required	No
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.

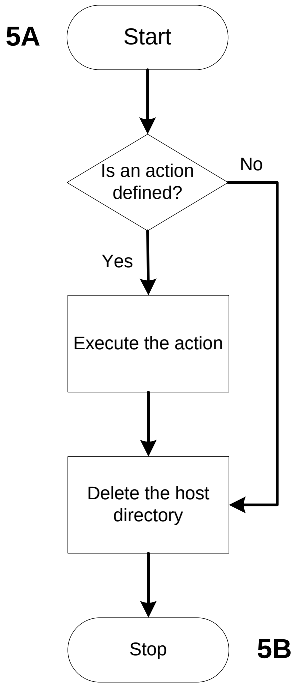


Figure 8: Cleaning up the Host Machine (Stage 5)

Requirements

1. 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.
2. 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.
3. 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.
4. The hostCleanUp action implementation must be placed in the host exec directory.

Information

1. 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.
2. 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.
3. 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.
4. Use the U3_* defined paths to maintain U3 application isolation.
5. If the U3 device is ejected before hostCleanUp is called, the U3_IS_DEVICE_AVAILBLE 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.

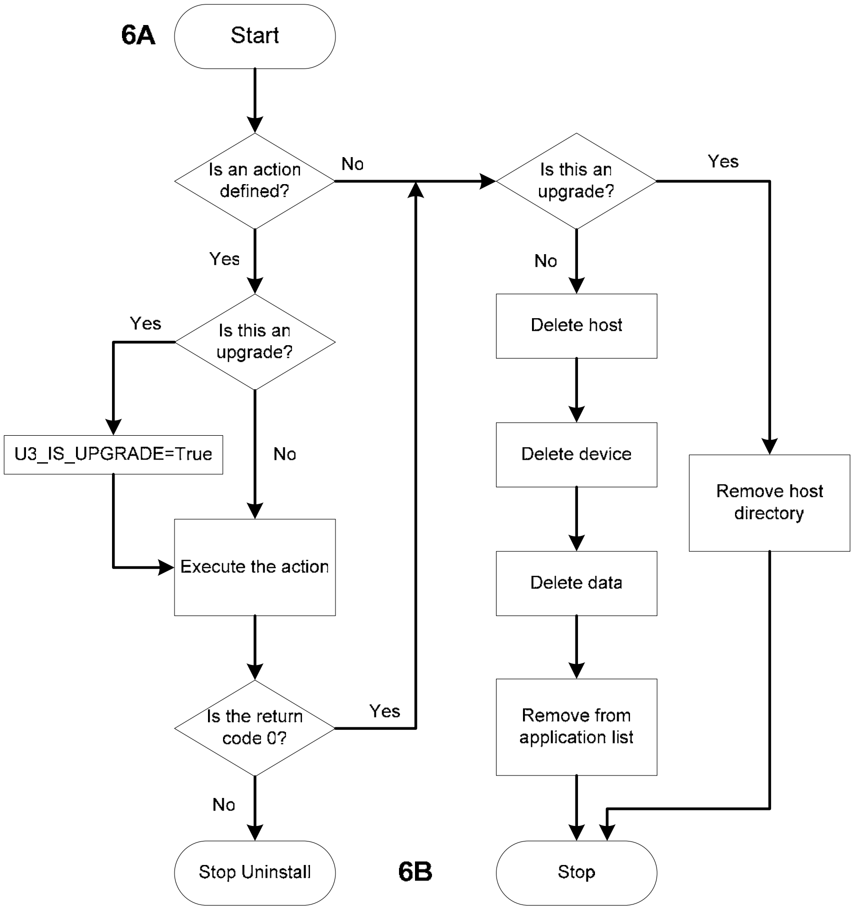


Figure 9: Uninstalling the U3 Application (Stage 6)

Requirements

1. 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.

2. If the Device API (DAPI) was used to write cookies, these should be removed during the deviceUninstall action.
3. The deviceUninstall action should execute without requiring the hostConfigure action to be executed.

Information

1. deviceUninstall may interact with the user. It may prompt the user to confirm removal of the U3 application.
2. Before deviceUninstall is executed, appStop is executed if the U3 application is running.
3. If hostConfigure has been executed, then hostCleanUp is executed before deviceUninstall is called.
4. After deviceUninstall is executed, the U3_HOST_EXEC_PATH, U3_DEVICE_EXEC_PATH and U3_APP_DATA_PATH directories are removed.
5. 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.
6. In an upgrade procedure, if deviceUninstall returns a non-zero error code, then the upgrade is cancelled.
7. 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 Section 6.7.3, 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.

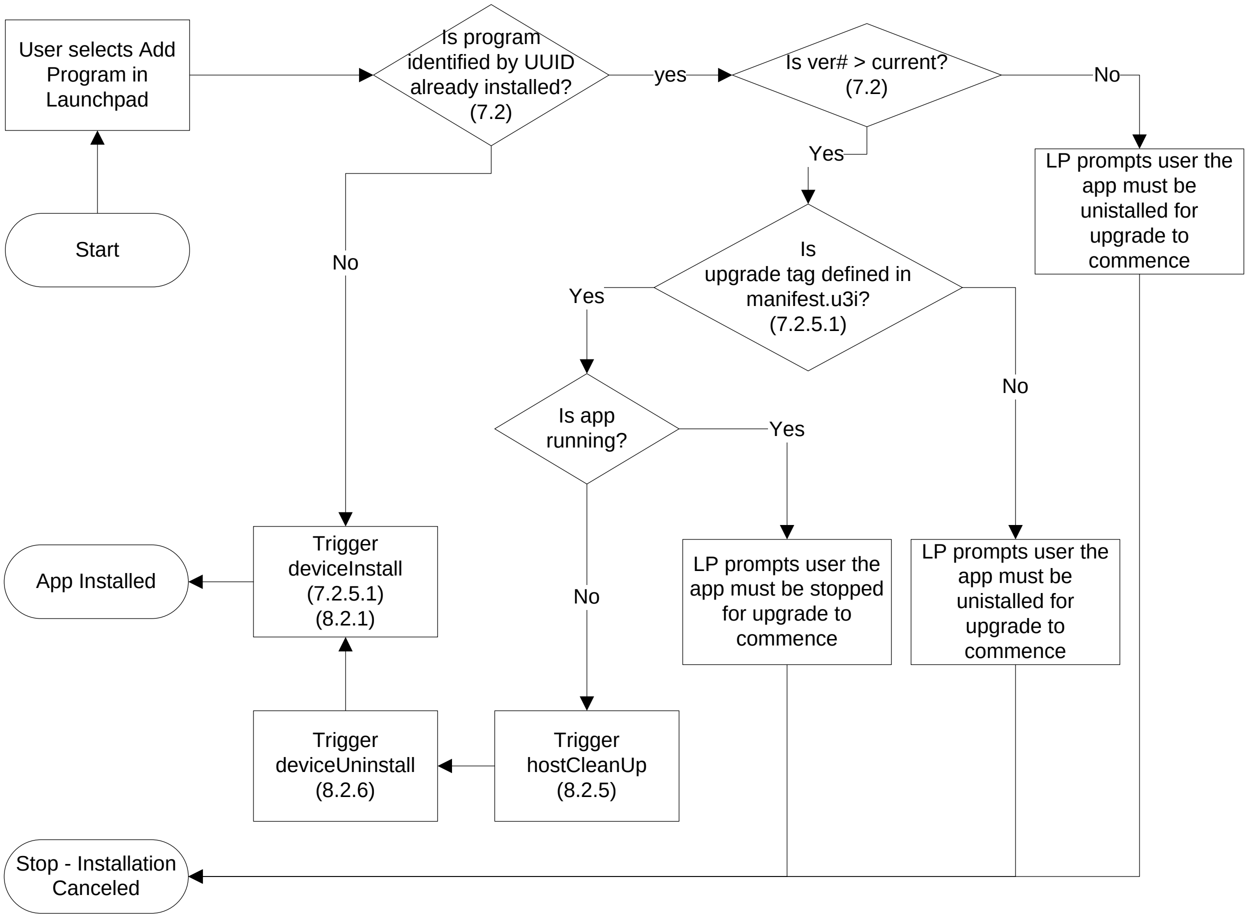

User selects Add Program in Launchpad
Start
Is program identified by UUID already installed? (7.2)
yes
No
Is ver# > current? (7.2)
No
Yes
LP prompts user the app must be unistalled for upgrade to commence
Is upgrade tag defined in manifest.u3i? (7.2.5.1)
Yes
No
Is app running?
Yes
No
App Installed
Trigger deviceInstall (7.2.5.1) (8.2.1)
LP prompts user the app must be stopped for upgrade to commence
LP prompts user the app must be unistalled for upgrade to commence
Trigger deviceUninstall (8.2.6)
Trigger hostCleanUp (8.2.5)
Stop - Installation Canceled

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 U3 Launchpad creates a runtime environment for the U3 application and passes this information through the environment variables. Due to the mobile nature of U3 applications, the values may be different each time the U3 application is executed anew. The variable values are defined when the U3 application is run, and do not change even if there is a change in the environment, such as the U3 device no longer being available.

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.

See Section 6.3, Host Exec Directory.

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.

See Section 6.4, Device Exec Directory.

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 Section 9, U3 Runtime Variables.

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 Section 3.4 in the *DAPI Reference Guide* or the example in Section 5.1 step 7 in the *SDK Developer Guide* for more information.

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 Guigen 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:

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

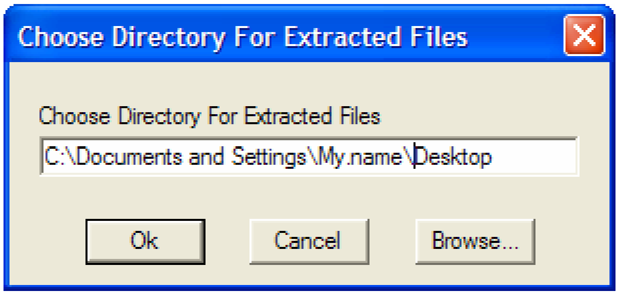

Choose Directory For Extracted Files
Choose Directory For Extracted Files
C:\Documents and Settings\My.name\Desktop
Ok
Cancel
Browse...

Figure 11: GUID Generation, Choose Directory

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

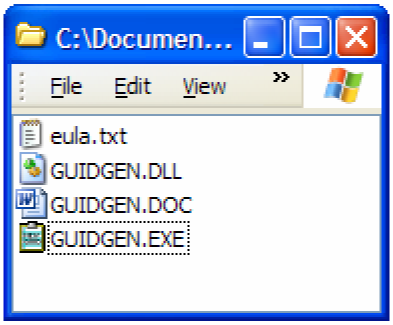
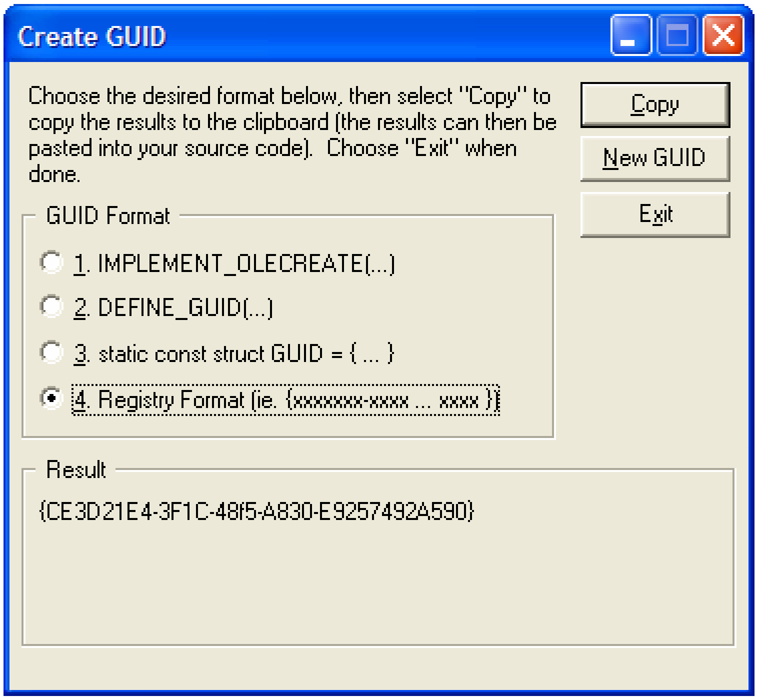


Figure 12: GUID Generation, Extracted Files

To use GUIDgen:

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



Choose the desired format below, then select "Copy" to copy the results to the clipboard (the results can then be pasted into your source code). Choose "Exit" when done.

GUID Format

○ 1. IMPLEMENT_OLECREATE(...)
○ 2. DEFINE_GUID(...)
○ 3. static const struct GUID = { ... }
● 4. Registry Format (ie. {xxxxxxx-xxxx ... xxxx })

Result

{CE3D21E4-3F1C-48f5-A830-E9257492A590}

Figure 13: GUID Generation, Create GUID window

2. Select option 4. “Registry Format…” To generate a new GUID.
3. Click **New GUID** to generate a new GUID. The GUID number is displayed in the Result area.
4. Click **Copy** to copy the generated GUID to the clipboard.
5. Paste the GUID into the manifest file.
6. 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 MANIFEST FILES

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	
Vendor	
Description	

Parameter	Description
Location: Japan	1041
Name	
Vendor	
Description	

U3 Platform 1.0 SDK, Application Deployment Guide

U3
303 Twin Dolphin Drive, 6th Floor
Redwood City, CA 94065
1-800-837-3654, info@u3.com

For more information, visit www.u3.com