



U3 Integration Factors Worksheet

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

Standard integration tasks:

Summary:

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

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

Integration Factors Checklist

Each integration factor in the following table should be reviewed to determine whether or not it applies to the application under consideration. Each integration factor refers to a separate section in this document that contains additional information as to how to evaluate the applicability of the integration factor itself and guidelines to resolve it. For any integration factors that have been checked, the project manager should consult with the appropriate resources to determine the number of resources and time required for each factor's solution.

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

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



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

Description:

Application uses COM objects. This is problematic as the objects would need to be registered temporarily and then unregistered during the Host Clean Up runtime phase. If other applications attach to the object while it is registered, it can't be removed from the system thus violating one of the main precepts of the U3 value statement which is to leave the system configuration in the same state as it was before the U3 device was inserted. Note that object registration requires Administrator permissions. The application would not be able to use these objects in a User Mode scenario which may be a common occurrence on public systems.

Solution:

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

IF #2: Application uses the registry

Description:

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

There are three principle issues that registry use causes.

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

Solution:

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



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

Description:

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

Solution:

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

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

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

Description:

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

Solution:

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

IF #5: Implements an upsell/activation feature

Description:

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

Solution:

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

IF #6: Uses or implements a DRM

Description:

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

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

Solution:

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

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

2) Dynamic Path Support:

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

3) Use host machine resources:

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



IF #7: Requires implementation of a multiple instance strategy

Description:

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

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

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

Solution:

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

IF #8: Exceeds flash memory friendly file update rate

Description:

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

Solution:

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

P/E cycle rate calculator:

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

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

Description:

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

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

Solution:

Solutions for these situations may require that install and upgrade actions support the business rules used to address these situations in the field. Transferring a license to another device could be handled by a special installation executable that manages the license transfer as an example. Explicit solutions are not prescribed here. The intent of this integration factor is to make the developer aware of the inevitability that one or more of these situations will occur. Most developers find that their existing business rules suffice with regard to this IF.



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

Description:

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

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

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

Solution:

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

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

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

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

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

Description:

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

Solution:

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

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

IF #12: Application written in Java

Description:

Application requires presence of an appropriate JVM.

Solution:

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

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

IF #13: Requires .NET framework runtime package

Description:

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

Solution:

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

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

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