Micro Focus Desktop Containers 12.1

User Guide

April 2017

Legal Notices

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

TABLE OF CONTENTS

1 OVERVIEW

Thank you for using Micro Focus Desktop Containers Studio!

This product will allow you to convert your Windows, .NET, Java, AIR, Flash, Shockwave, or other Windows-compatible application into a self-contained application container that can be streamed from the web and run instantly on an end-user device. Unlike traditional deployment methods, application containers do not require reboots, administrative privileges, or separate setup steps for external components and runtimes. Application containers are isolated from other system applications, preventing DLL conflicts and other deployment nightmares.

This guide explains how to use Micro Focus Desktop Containers Studio to create your own application containers and begin enjoying the benefits of this next-generation deployment technology.

1.1 WHAT IS AN APPLICATION CONTAINER?

Application containers enable application publishers and IT administrators to significantly reduce the cost and complexity associated with development, setup, configuration, deployment, and maintenance of software applications. An application container is a virtual machine image pre-configured with files, registry data, settings, components, runtimes, and other dependencies required for a specific application to execute immediately. This application container improves both the user experience and reduces test and support complexity associated with deploying the application. For example, a publisher of an application based on the Microsoft .NET Framework or Java runtime engine can create an application container combining the application with the required runtime engine. An end-user can then run this application immediately, even if he has an incompatible (or has not installed the) runtime engine.

Because application containers run in isolated environments, applications that otherwise interfere with one another can run simultaneously. For example, applications which overwrite system DLLs or require different runtime engine versions can run concurrently on a single host device. Application containers can also provide access to internal copies of privileged system resources, enabling unprivileged users to directly execute applications without security exceptions or Windows User Account Control prompts.

Micro Focus application container technology surpasses other virtualization systems in several ways:

- Micro Focus application containers run immediately on an end-user machine without changes to system infrastructure. No "player" software or separate installation is required.

- Micro Focus application container technology is low-overhead and enables applications to run with the same performance characteristics as native executables. No significant processing or filesystem overhead is incurred.

- Micro Focus application containers provide all required virtualized operating system functionality within the internal container environment. No operating system must be installed onto the application container.

1.2 MICRO FOCUS DESKTOP CONTAINERS FEATURES OVERVIEW

Micro Focus Desktop Containers enables you to:

- Create application containers which are streamed from the web, eliminating potentially long installation and download times. Application containers run from any desktop with broadband internet access.
- Create an application as a single executable. Application files, registry settings, runtimes, and components are packaged into a single executable that runs instantaneously.
- Run Java and .NET without separate runtime installations. Java and/or .NET-based applications run immediately, with no separate installation steps or runtime versioning conflicts.
- Improve desktop security. You can run applications without granting administrative permissions to end-users. You can also stabilize desktop images by deploying applications in Spoon sandboxed virtual environments.
- Eliminate third-party setup dependencies. MFDC integrates third-party components, COM/VB controls, and content viewers such as Acrobat, Flash, and Shockwave, directly into your application.
- Eliminate Windows User Account Control prompts and compatibility errors. You can deploy Micro Focus application containers regardless of access to privileged system resources.
- Leverage Terminal Services and Citrix investments: By isolating applications from global resource areas, Micro Focus application container technology allows non-compliant applications to function properly in Terminal Server and Citrix environments.
- Improve mobile productivity. By placing Micro Focus application containers onto a USB flash-memory drive, you can run applications immediately on remote PCs, with no installation steps, administrative privileges or driver installations.
- Dramatically reduce test and support costs. Micro Focus application containers eliminate versioning conflicts, dependencies, and "DLL Hell." Micro Focus application containers also reduce test complexity and eliminate support requests associated with dependency installation and inter-application resource conflicts. Micro Focus application containers function entirely in user-mode, so no device drivers are installed or required.

1.3 DO MICRO FOCUS APPLICATION CONTAINERS REQUIRE ANY DEVICE DRIVERS?

No. Micro Focus application containers function entirely in user-mode, so no device drivers are installed or required.

1.4 How are Micro Focus application containers different from hardware virtualization?

Unlike hardware virtualization systems like Microsoft Virtual PC and VMware, Micro Focus application containers only replicate the operating system features required for application execution. This enables application containers to operate efficiently, with the same performance characteristics as native executables.

There are several advantages in choosing Micro Focus application containers over hardware virtualization systems:

- Optimal performance. Micro Focus application containers run at the same speed as applications running natively against the host hardware, with a minimal memory footprint. In contrast, applications running within hardware-virtualized environments experience significant slow-downs and impose a large memory footprint.

- Dramatically reduced application size. Micro Focus application containers require a footprint proportional to the size of the application, data, and included components. As a result, Micro Focus application containers are small enough to be quickly downloaded by end-users. Hardware virtualization requires an entire host operating system image, including many basic subsystems that are already present on the end-user device. Each virtual machine may occupy several gigabytes of storage.

- Multiple application capability. You can run multiple simultaneous Micro Focus application container environments per processor. Due to the high overhead of hardware virtualization, only a small number of hardware-virtualized environments per processor can run simultaneously.

- Reduced licensing costs. Micro Focus does not require the purchase of separate operating system licenses to use an application container. Hardware virtualization systems require a host operating system in order to function, which can impose additional licensing costs and restrictions.

Hardware virtualization can be appropriate in certain specialized scenarios:

- Non-Windows operating systems. Micro Focus application containers currently only run using the Windows operating system. Hardware virtualization can execute any operating system compatible with the underlying virtualized hardware, such as Linux.

- Kernel mode virtualization. Micro Focus application containers only replicate user-mode operating system features, whereas hardware virtualization systems emulate the entire OS stack, including kernel mode components. Applications requiring device drivers or other non-user-mode software may require a hardware-virtualized environment to function properly.

Carefully evaluate the advantages and disadvantages of different approaches before deciding which technology best fits your needs.

1.5 WHAT PLATFORMS ARE SUPPORTED?

Micro Focus Desktop Containers support the following platforms for application container build, snapshot, and execution:

- Windows XP Service Pack 3
- Windows Embedded XP
- Windows Server 2003 Service Pack 2
- Windows Vista Service Pack 2
- Windows 7
- Windows Server 2008
- Windows 8
- Windows Server 2012
- Windows 8.1
- Windows Server 2012R2
- Windows 10

Micro Focus Desktop Containers support these operating systems running within VMware and Microsoft hardware virtualization and hypervisor environments.

Micro Focus Desktop Containers also has limited support for the Windows Preinstallation Environment (WinPE), though certain applications depending on operating system features unavailable in WinPE may not function properly.

Micro Focus Desktop Containers creates 32- and 64-bit executables. Both 32-bit (under 32-bit mode) and 64-bit executables can be run on x64-based platforms.

1.6 What applications can be virtualized using Micro Focus Desktop Containers?

Micro Focus Desktop Containers support most major Windows desktop applications. However, certain applications are unsuitable for using Micro Focus application container user-mode technology. These include application features which contain, or directly depend upon, interaction with specialized kernel-mode device drivers or other kernel-mode extensions, operating system components and extensions, anti-virus applications, and kernel event filtering, monitoring, and intrusion detection applications.

Micro Focus application containers are compatible with most major anti-virus, runtime, and security packages.

2 GETTING STARTED

This section describes the system requirements for installing and running Micro Focus Desktop Containers Studio, provides an overview of the Micro Focus Desktop Containers Studio user interface, and walks you through the basic steps of creating an application container.

2.1 SYSTEM REQUIREMENTS

Micro Focus Desktop Containers Studio requires a Windows XP, Windows 2003, or higher operating system. The Micro Focus Desktop Containers Studio graphical interface assumes a screen resolution of at least 800×600, although a screen resolution of at least 1024×768 is recommended.

2.2 User Interface Overview

The Micro Focus Desktop Containers Studio control panel allows you to configure your application container filesystem and registry, embed external runtimes and components, take snapshots of the application, and create application container executables. The primary interface consists of a ribbon bar and several panes grouped by a functional area.

Located above the ribbon bar are:

- The **Start menu** button- located in the circle on the top left of the window- allows application container configurations to be imported, opened, applied, saved, and closed.
- The **Options** bar provides Micro Focus Desktop Containers Studio interface customization options and the ability to install license certificates.
- The **Help** bar provides access to the Micro Focus Desktop Containers Studio documentation, including a searchable version of this document.

The ribbon bar accesses common Micro Focus Desktop Containers Studio features:

- The **Virtual Application** tab provides access to the snapshot and build features, as well as output configuration options such as the startup file, output directory, and diagnostic-mode selection.
- The **Runtimes** tab provides a selection of auto-configurable runtime engines which can be embedded into your application container with a single click. These include .NET Framework, Java, Flash, Shockwave, Acrobat Reader, and SQL runtimes.
- The **Advanced** tab provides advanced Micro Focus Desktop Containers Studio functions such as Platform Merge and Streaming.

Functions in the main panel are accessed by clicking the appropriate buttons along the left side of the interface:

- The **Filesystem** panel displays the application virtual filesystem, and allows adding and removing virtual files and directories.
- The **Registry** panel displays the application virtual registry, and allows adding and removing virtual registry keys and data values.
- The **Settings** panel allows configuration of application container metadata, startup image, startup and shutdown shims, and process configuration options.
- The **Components** panel allows layering of external virtual application components, such as toolbars and optional features.
- The **Setup** panel allows configuration of MSI setup package, shortcuts, and shell integration options.
- The **Expiration** panel allows configuration of application expiration options.
- The **Network** panel allows configuration of application proxy and DNS settings.

- The **ZENworks** panel allows configuration of Bundle Publishing features.

Note**:** Micro Focus Desktop Containers Studio users are individually responsible for assuring compliance with licensing for any third-party redistributable components included using virtualization.

2.3 METHODS OF CREATING APPLICATION CONTAINERS

Micro Focus Desktop Containers Studio offers three ways to create and configure application containers.

- Build am application container from a template: Micro Focus Desktop Containers Studio includes templates and TurboScripts for many popular applications along with the option to import container images from the Turbo.net hub. This option automatically configures the applications using a guided wizard and user-provided or downloaded media. This method is recommended for first-time users of Micro Focus Desktop Containers Studio. When viewing the list of available applications, templates are marked with an asterisk (*) and do not require the software to be installed during packaging. Templates are pre-configured and may require changes for custom configurations. TurboScripts require the application to be installed and configured on the local system. The settings from the local system will be applied to the configuration.

- Snapshot a third-party application or component: In this method, snapshots capture the system state before and after an application is installed. Based on the observed system changes, the application container settings are automatically configured. This method is ideal for off-the-shelf applications (Refer to the section “Snapshotting applications” for more information on this method).

- Install application into a container: In this method, an empty container is started and you are able to install your application directly to it as if on a clean machine. This method is ideal for simple applications when a clean snapshot machine is not readily available.

- Manually configure an application: This method is most often used by developers with internally developed applications. Manual configuration requires a high degree of technical knowledge but allows the maximum amount of control over application container settings (Refer to the sub-section “Manually configuring a simple application container” in the “Walkthroughs” section for more information on this method).

All methods allow additional configuration and customization once the initial application container configuration has been constructed.

2.4 CREATING YOUR FIRST APPLICATION CONTAINER

Micro Focus Desktop Containers Studio includes automated application container configuration wizards for certain popular software applications. Micro Focus recommends that first-time users begin by building one of these auto-configurable application containers using the Micro Focus Desktop Containers Studio Configuration Wizard.

To build an auto-configured application container:

- Open the Micro Focus Desktop Containers Studio Configuration Wizard. The wizard is displayed on program startup, or can be opened by pressing the **Configuration Wizard** button on the **Virtual Application** ribbon bar.
- Click on the box labeled **Build a virtual application from a template**.
- Select an application from the **Application** list. There are three types of application configurations, **templates**, **TurboScripts**, and r**epositories**. Applications that are built from a **templates** or **repositories** do not require a local installation. Applications built from **TurboScripts** require that there be a local installation of the application on the machine.
- Follow the wizard steps to construct the application container.

After completing the wizard, the application container configuration will remain loaded in the Micro Focus Desktop Containers Studio interface. This allows the configuration settings generated by the wizard to be inspected and additional customization to be performed (Refer to the following sections “Configuring application containers” and “Customizing application containers” for more information on configuration and customization).

Congratulations on building your first application container!

2.5 CONFIGURING APPLICATION CONTAINERS

Application containers enable you to simplify application deployment by directly embedding files, registry settings, components, and other dependencies into the application executable. This reduces setup complexity, prevents DLL collisions, and enables applications to simulate the use of privileged disk and registry resources without administrative privileges.

2.6 SNAPSHOTTING APPLICATIONS

Most commercial applications require complex combinations of filesystem and registry entries. To facilitate virtualization of these applications, Micro Focus Desktop Containers Studio can *snapshot* application installations and automatically configure them based on modifications made to the host system during application setup.

2.6.1 Snapshots

Snapshots use "before" and "after" images of the host machine to determine the application container configuration:

- "Before" snapshot: Taken prior to installing the application. This captures the state of the host device without the target application installed.
- "After" snapshot: Taken after installing the application. This captures all changes to the host device during application installation. Micro Focus Desktop Containers Studio then computes the changes, or *deltas*, between the before and after snapshots, and inserts these changes into the configuration.

To use the snapshot feature:

- Prepare the host device by either removing the target application and all dependencies, or copying Micro Focus Desktop Containers Studio onto a "clean" machine.
- Click on the **Virtual Application** tab on the ribbon bar and click **Capture Before**. This captures the "before" snapshot image. Snapshotting iterates through the filesystem and registry, and therefore may take several minutes to complete.
- (Optional) Micro Focus recommends you save the "before" snapshot before continuing. This allows you to skip this step when snapshotting subsequent applications from the same clean machine image. To save the snapshot, click on the down arrow underneath the **Capture Before** button and select **Save Snapshot**. Note that while Micro Focus Desktop Containers Studio automatically saves the most recent captured "before" snapshot, this snapshot is reset once the Capture and Diff process is complete.
- Install your application along with other files, settings, runtimes, and components you wish to include in the application container (Refer to the following sub-section "Adding runtimes and components" for more information on additional configuration). If the application setup requests a reboot, be sure to save the "before" snapshot and then proceed with the reboot.
- On the **Virtual Application** tab on the ribbon bar, click **Capture and Diff**. This captures the "after" snapshot, computes the deltas between the two snapshots, and populates the application container with the delta entries.
- (Optional) Review the filesystem and registry entries, and remove any files or settings which are not required for proper execution of your application container. Removing unused entries will reduce application container size, but be sure to avoid accidental removal of required resources, as it will cause your application container to no longer function properly.

2.6.2 Saving snapshots

Sometimes the before snapshot remains fixed while several after snapshots are taken. It is recommended that you to save the before snapshot image so that the before snapshot does not need to be re-captured each time. Because snapshots can take up to several minutes, this can significantly reduce the time required to build application containers.

To save the "before" snapshot, click on the down arrow underneath the **Capture Before** button on the **Virtual Application** ribbon bar and select **Save Snapshot** from the dropdown menu. Select an appropriate filename and location and press **Save**. Similarly, to load a saved snapshot, select the **Load Snapshot** menu item and navigate to the saved snapshot file. To clear the current "before" snapshot image, select the **Clear Snapshot** menu item.

2.6.3 Internet Explorer snapshot compatibility mode

Internet Explorer snapshot compatibility mode should be used when snapshotting Internet Explorer application setups. This mode ensures a more complete capture of the Internet Explorer portions of the registry and filesystem, whereas a normal snapshot process ignores those parts of the registry and filesystem.

To use Internet Explorer snapshot compatibility mode, click the **Options** menu above the ribbon bar in Micro Focus Desktop Containers Studio, and select **Internet Explorer snapshot compatibility mode** from the dropdown.

2.7 ADDING RUNTIMES AND COMPONENTS

Many components and runtime systems consist of large, complex sets of filesystem entries and registry settings. To simplify configuration of the most common components, Micro Focus Desktop Containers Studio contains a collection of pre-configured component settings which can be added to your application container with a single click. For example, if your application is a .NET Framework 4.0 application, then selecting the **.NET Framework 4.0** component will allow your executable to run on machines without the .NET Framework installed.

Additional runtimes and components should be added to the application container during the snapshot process, *before* the "After" snapshot has been taken.

To add a runtime or component:

- Click on the **Runtimes** tab on the ribbon bar.
- Click on the appropriate runtime or component to select it for inclusion. Selected components are indicated with a highlighted button. To remove a component, click on the button again. The button will no longer be highlighted and the component will not be included.

Note: The displayed runtimes apply to the currently selected target architecture (see process settings). If the target architecture is changed, architecture specific runtimes, like .NET 2 or .NET 3.x will still be included but will not display as selected. To deselect them, the target architecture must be changed back, and then the runtimes can be deselected.

Note: Depending on the size of the component, selecting a component for inclusion can significantly increase the size of the resulting executable. Therefore, you should only select components which are required for proper execution of your application.

Note: You are responsible for assuring compliance with licensing for any third-party redistributable components included in your application container.

2.7.1 Configuring the Java runtime

Micro Focus Desktop Containers Studio provides specialized support for the Java runtime. If your application is based on Java runtime, press the **Sun Java Runtime** button on the **Runtimes** ribbon bar. This displays the Java configuration menu.

Select the appropriate version of the Java runtime from the **Java runtime version** drop-down. If you are deploying your application as a set of .class files, then select the **Class** option from the **Startup type** drop-down; if you are deploying within a .jar file, select the **Jar** option. Enter the startup class name or Jar name in the appropriate textbox, along with any additional Java runtime options.

2.8 LOADING AND SAVING CONFIGURATIONS

Once you have configured your application container, and taken your "After" snapshot, save the configuration for future use or modification. It is important to save the application container snapshot in its original state in case errors are introduced during customization and optimization. This also allows the application to be tested and modified without the need to re-snapshot after each iteration.

To save a configuration:

- Click on the **Start button** menu and select **Save Configuration As...**
- Select a filename and location and click **Save**. This saves the application container configuration file. By default, configuration files use the extension **.xappl**.

Note: Configuration files do not store the *contents* of virtual filesystem files. The configuration file specifies only the *source path* for each virtual filesystem entry. The source file must exist at build time or the application container will not build successfully.

Micro Focus Desktop Containers Studio automatically stores source file locations as paths relative to the location of the saved **XAPPL** file, in the same directory as the **XAPPL** file (the folder marked **Files**).

Note: Micro Focus Desktop Containers Studio users can create their own default settings for application building by saving a **XAPPL** file with a few customization settings (E.g. changing the default sandbox location) and no associated application, then loading this **XAPPL** file (and saving as a new file) before beginning the snapshot process in a new project.

2.9 SPECIFYING A STARTUP FILE

Although the virtual filesystem can contain several executable files (such as .exe, .cmd, and .java) and viewable file formats (such as .html and .swf), your application container is consolidated into a single executable. It is necessary for the application container designer to indicate a *startup file*- the executable or viewable file which is opened when the user executes the application container.

A startup file should be specified *after* the application has been installed and configured with runtimes, and the "After" snapshot has been taken.

To select the startup file:

- Click on the **Virtual Application** tab on the ribbon bar.
- Click on the **Startup File.** This opens a drop-down menu listing all files in the virtual filesystem.
- Select the file to be used as the startup file, or navigate to the desired startup file in the virtual filesystem display, right-click the file, and select **Set as Startup File**.

Files located on the host device (outside of the virtual filesystem) may also be used as startup files. To select a file on the host device as the startup file, enter the full path to the desired startup file in the **Startup File** text box. Remember to use well-known root folder variables such as **@WINDIR@** and **@PROGRAMFILES@** as the root of the full path to ensure that the startup file can be properly located on all base operating systems.

Note: While any file can be selected as the startup file, you should only select a file which is executable or viewable. Selecting a file which cannot be opened will cause an error when the application container is started.

2.10 SPECIFYING MULTIPLE STARTUP FILES (JUKEBOXING)

In some situations an application container can expose multiple startup files. For example, when using a virtualized office productivity suite, you may want to launch the word processor or spreadsheet component of the suite while still deploying a single executable file.

Micro Focus Desktop Containers Studio enables triggering multiple entry points into the application container based on a command-line argument to the executable. For example, in the office suite scenario described, you could use the command line argument **office word** (to trigger the word processor) and **office spreadsheet** (to trigger the spreadsheet). Refer to the sub-section "Building OpenOffice via snapshot process" in the "Walkthroughs" section for an example of Jukeboxing during the OpenOffice build process.

To specify multiple startup files:

- Click the **Multiple** button next to the **Startup File** textbox on the **Virtual Application** ribbon bar. This displays the **Startup Files** selection dialog.
- Click on the **File** column on the first empty row in the startup file list and select the desired file from the drop-down list. Files located on the host device (outside of the virtual filesystem) can also be used as startup files. To select a file on the host device as the startup file, enter the full path to the desired startup file in the **Startup File** text box.
- Enter the desired command line arguments, if any, in the **Command Line** column.
- Enter the desired command line trigger in the **Trigger** column. For example, in the command line **office word**, the trigger would be **word**.
- Check the **Auto Start** checkbox if you want the startup file to always be automatically launched on application container startup.
- After adding a new startup file, hit **Enter** in order to save

Note: When specifying a startup file located outside of the virtual filesystem, remember to use well-known root folder variables such as **@WINDIR@** and **@PROGRAMFILES@** as the root of the full path to ensure that the startup file can be properly located on all base operating systems.

Note: The **Auto Start** flag can be specified for multiple startup files to automatically launch multiple applications that are typically used together in a single session (also known as "shotgunning").

2.11 EDITING THE VIRTUAL FILESYSTEM

Micro Focus Desktop Containers Studio allows you to embed a *virtual filesystem* into your executable. Embedded files are accessible by your application container as if they were present in the actual filesystem. Virtual files, unlike files on the host device, are not visible from and do not require changes to the host device. Virtual files do not require security privileges on the host device, regardless of whether the virtual files reside in a privileged directory. Because virtual files are embedded in the application container executable, shared DLLs do not interfere with or are overwritten by other applications on the host device.

To add virtual files:

- Click on the **Filesystem** button located on the left side of the Micro Focus Desktop Containers Studio window.
- Using the view on the right, add the files and folders you wish to embed in the application executable. The **Application Directory** root folder represents the folder containing the application container binary on the executing device; the other root folders represent the corresponding folders on the host device.

Note: When running an application container on Windows 7 or later, the **All Users Directory\Application Data** and **All Users Directory** root folders will map to the same folder at runtime. To prevent one setting from overriding another, verify that the isolation settings for these folders are the same.

2.11.1 Virtualization Semantics

In the event of a collision between a file in the virtual filesystem and a file present on the host device, the file in the virtual filesystem takes precedence.

Folders may be virtualized in **Full**, **Merge**, **Write Copy** and **Hide** mode:

- Full mode: Only files in the virtual filesystem will be visible to the application- even if a corresponding directory exists on the host device- and writes are redirected to the sandbox data area. Full mode is generally used when a complete level of application container isolation is desired.
- Merge mode: Files present in a virtual folder will be merged with files in the corresponding directory on the host machine, if such a directory exists. Writes to host files are passed through to the host device and writes to virtual files are redirected into the sandbox data area. Merge mode is generally used when some level of interaction with the host device is desired. For example, Merge mode might be used to allow the application container to write to the host device's **My Documents** folder.
- Write Copy mode: Files present on the host device are visible to the virtual environment, but any modifications to folder contents are redirected to the sandbox data area. Write Copy mode is generally used when a application container needs to read from files already present on the host device but isolation of the host device is still desired.

- Hide mode: Files and folders with Hide isolation enabled will return a ‘File Not Found’ message to the application at runtime. This applies to both the virtual filesystem and the host filesystem. Hide mode is generally used when a file on the host machine may interfere with the application’s ability to run properly. By adding the file or folder to the virtual package with Hide isolation enabled, the application will receive a ‘File Not Found’ message, even if the file or folder exists on the host machine.

Tip: To apply the selected isolation mode to all subfolders, right-click on the folder, choose **Isolation**, click on the checkbox for **Apply to subfolders**, and click **OK**.

2.11.2 File Attributes

Files and folders may optionally be hidden from shell browse dialogs and other applications enumerating virtual directory contents. This is often used to prevent internal components and data files from being displayed to the user. To hide a file or folder, click on the checkbox in the **Hidden** column next to the desired file or folder.

Note: The hidden flag should not be confused with Hide isolation mode. Enabling the hidden flag only prevents a file or folder from being displayed in browse dialogs or from directory enumeration APIs. It does not prevent the application (and potentially the end-user) from accessing the folder or file contents by direct binding. To prevent the file or folder from being found by the application, enable Hide isolation mode.

Flagging files as read-only will prevent the application from modifying the file. To make a file read-only, click on the checkbox in the **Read Only** column next to the desired file. It is not possible to set the read-only attribute at the folder level.

2.11.3 No Upgrade

By default, Micro Focus Desktop Containers Studio enables files in the virtual filesystem to be upgraded with patches. If there are files in the virtual filesystem that should not be upgraded, such as user data files, select the **No Upgrade** flag to prevent these files from being upgraded.

2.11.4 Filesystem Compression

To reduce executable size, Micro Focus Desktop Containers Studio can compress virtual filesystem contents. This typically reduces application container payload size by approximately 50%, but also prevents profiling and streaming of the application. By default, the **Compress Payload** option in the **Process Configuration** area of the **Settings** panel is unchecked. This option should remain unchecked during the build process in order to be able to profile and stream applications.

Note: Disabling payload compression may significantly increase the size of the application container binary.

2.12 EDITING THE VIRTUAL REGISTRY

Micro Focus Desktop Containers Studio allows you to embed a *virtual registry* into your executable. Embedded registry keys are accessible by your application container as if they were present in the actual registry. Unlike data present in the registry on the host device, virtual registry keys and values are not visible from and do not require changes to the host device. The use of a virtual registry does not require any security privileges on the host device, even if the virtual registry entries reside in a privileged section of the registry, such as **HKEY_LOCAL_MACHINE**. Also, because virtual registry entries are embedded in the application executable, other applications are unable to disrupt application execution by inadvertent modification of registry entries.

To add virtual registry data:

- Click on the **Registry** button located on the left side of the Micro Focus Desktop Containers Studio window.
- Add or remove the registry keys and values you want to embed in the application executable. When adding blob data, enter the values in hexadecimal format.

The **Classes** root, **Current user** root, **Local machine**, and **Users** root folders correspond to the **HKEY_CLASSES_ROOT**, **HKEY_CURRENT_USER**, **HKEY_LOCAL_MACHINE**, and **HKEY_USERS** keys on the host machine.

Registry string values may include well-known root folder variables such as **@PROGRAMFILES@** and **@WINDIR@**.

2.12.1 Virtualization Semantics

In the event of a collision between a key or value in the virtual filesystem and data present on the host device registry, information in the virtual registry takes precedence.

Keys may be virtualized in **Full, Merge, Write Copy** or **Hide** mode:

- Full mode: Only values in the virtual registry will be visible to the application- even if a corresponding key exists on the host device- and writes are redirected to the user registry area.
- Merge mode: Values present in a virtual key will be merged with values in the corresponding key on the host machine, if such a key exists. Writes to host keys are passed through to the host registry and writes to virtual keys are redirected to the user registry area.
- Write Copy mode: Used when an application container must be read from registry keys already present on the host device, but isolation of the host device is still desired. Keys and values present on the host device are visible to the virtual environment, but any modifications to keys or values are redirected to the sandbox data area.
- Hide mode: Keys and values in the virtual registry or the corresponding host registry will not be found by the application at runtime.

Tip: To apply the selected isolation mode to all subkeys, right-click on the key, choose **Isolation**, click on the checkbox for **Apply to subkeys**, and click **OK**.

2.12.2 Importing Registry Hive Files

Micro Focus Desktop Containers Studio can import registry hive (.reg) files into the virtual registry. To import a .reg file, click the **Import** button in the **Registry** panel and select the registry hive file to be imported.

2.13 EMBEDDING A DATABASE ENGINE

Micro Focus Desktop Containers Studio allows SQL Server Express to be embedded directly into the executable file. This allows executables to run a SQL Server user-instance for a virtual .mdf database file.

To select SQL Server 2005 Express:

- Click on the **Runtimes** button on the ribbon bar.
- Choose the SQL Server 2005 Express option.

2.13.1 Configuring your application to use an embedded database

The code displayed on the database configuration dialog provides an example of how to create a connection string for connecting to the user-instance database that loads an .mdf file from the virtual filesystem.

The virtualized SQL Server 2005 Express default instance name is stored in a special environment variable named **%SQLXENOCODE%**. A unique instance of this name is created at build time and is not user-configurable.

To embed a database file:

- Verify that the desired .mdf database file has been added to the virtual filesystem.
- Choose the .mdf database file to attach using the **AttachDBFilename** drop-down. This path is stored in the **%AttachDBFilename%** environment variable.

To manually modify this selection at a later time, click the **Environment Variables** button on the **Process Configuration** tab of the **Settings** pane. To add additional databases, create an additional Environment Variable for each database.

Note: The .NET 2.0 Framework is automatically included when SQL Server Express 2005 engine is selected. SQL Server 2005 Express requires this component to be installed. If this component is removed, the application will only run on machines that have the .NET Framework 2.0 installed.

2.14 CREATING AND USING SHARED COMPONENTS

Sometimes multiple application containers share common sets of configuration options. For example, system administrators may want to share a common set of configuration options (browser bookmarks, application settings, etc.) across a department or enterprise. Micro Focus Desktop Containers Studio makes it easy to create, share, and consume virtual machine settings across multiple Micro Focus application containers using Micro Focus **SVM**-format components.

To create a shared component:

- Begin a new application container configuration.
- Configure the filesystem, registry, and settings as you would for a standard application container.
- Configure the application container settings exactly as in the case of a standard application container (i.e. using snapshotting, manual configuration, etc.).
- On the **Virtual Application** tab, select **Component** from the **Project type** drop-down.
- Press the **Build** button.

In **Component** mode, the build process results in creation of an **SVM** file instead of an executable file. An **SVM** contains the application container settings and data payload. **SVM** files are similar to virtual executable outputs, except that **SVM** files do ***not*** contain the Micro Focus Desktop Containers runtime engine. An **SVM** can only be used when combined as part of another application container.

To use an existing shared component:

- Open the component's application container configuration destination.
- Select the **Components** button to open the corresponding pane.
- Select **Import Components** if the component does not appear in the components table.
- Select the **SVM** to add as a component to the application container configuration. Select **OK**. The **SVM** metadata displays in the table when the process is complete.
- To include it in the application build, check the box adjacent to the **SVM** component. If the box is not checked the application is built without the component.
- Build the application.

To build without the component, uncheck the checkbox next to the component and rebuild. Project settings take precedence over settings in any loaded shared components.

To remove a component completely so it will not be available for other applications, select the component and click the **Remove Component** button.

Note: Imported components are stored under the user's profile, "%USERPROFILE%\Micro Focus\Components". If the project is moved from one computer to another, the components will need to be removed and re-imported on the new machine.

2.15 SANDBOX MERGE

Sandbox merge allows sandbox content and settings generated during application container execution to be applied to another application container configuration. Sandbox merge is an alternative to manual registry or filesystem configuration, and is particularly useful for applying additional customizations to existing application container configurations or configurations generated from an application container template.

To merge an existing sandbox into the active configuration:

- Click the **Sandbox Merge** button in the **Tools** section of the **Virtual Application** toolbar.
- Enter the path of the sandbox to be merged into the current configuration.
- Click **OK**.

For example, to customize the home page of the Firefox application container template:

- Use the **Configuration Wizard** to create a Firefox application container. (The wizard allows customization of the home page, but we will later use the sandbox merge feature to override the setting specified in the wizard.)
- Press **Build and Run** to launch the Firefox application container.
- Using the Firefox interface, specify a new browser home page.
- Exit the Firefox application container.
- Press **Sandbox Merge** to display the sandbox merge dialog. The sandbox path will be pre-populated with the location of the Firefox container virtual sandbox.
- Click **OK**.

The application container settings are updated with the configuration changes made during Firefox execution, including the updated browser home page.

3 Application Container Customization

This section describes advanced application container customization options, such as executable metadata, startup images, command-line arguments, and process startup options.

3.1 SELECTING A PROJECT TYPE

Application Containers supports two project types:

ISV Application: An application container project produces an executable file output (**.exe** file) that can be run directly from the operating system. Application output mode is appropriate for most users and is the default selection.

Portable Application: An application container project produces an executable file output (**.exe** file) that uses the Turbo runtime environment to execute the container image. This output mode is most appropriate when using container images imported from Turbo.net or the Turbo runtime environment.

Layer: A component project produces an **SVM** output (**.svm** file). **SVM** is a binary file format encoding all application container configuration and content into a single binary file. **SVM**s cannot be executed directly from the operating system. **SVM**s are used to exchange application container and component data between multiple application containers.

Note: In order to create **SVM**s for use in streaming applications, the project type must be set to component.

To set the project type, select the appropriate option from the **Project Type** menu under the **Virtual Application** tab

3.2 CUSTOMIZING EXECUTABLE METADATA

Executable metadata provides external applications such as the Windows shell with information regarding the application's identity, publisher, version, preferred display icon, and description. Metadata may be viewed and edited by clicking on the **Properties** tab of the **Settings** pane.

3.2.1 Standard metadata

Standard metadata includes information such as the product title, publisher, description, icon, web site URL, and version. By default, Micro Focus Desktop Containers Studio will apply metadata inherited from the application container startup file to the output application container executable. However, in some instances, it may be desirable to override the metadata.

To manually override executable metadata:

- Uncheck the **Inherit properties** option
- Enter the desired metadata in the appropriate fields in the **Properties** area.

To revert to the default inheritance behavior, recheck the **Inherit properties** option.

3.2.2 Custom metadata

In addition to standard Windows shell metadata, Micro Focus Desktop Containers Studio allows introduction of custom metadata into the output executable. Custom metadata may be used by specialized external executable viewer applications, inventory scanners, and other asset and licensing management systems.

To add or modify custom metadata:

- Click the **Custom Metadata...** button. This displays the **Custom Metadata** dialog.
- Enter the custom metadata property names and values into the dialog. Only string-type custom metadata values are supported.

For information on programmatically reading custom executable metadata, please consult the Microsoft Windows Software Development Kit.

3.3 ADDING A STARTUP IMAGE

Micro Focus Desktop Containers Studio allows you to specify a startup "splash" image to be displayed during application container startup. Startup images improve application branding and are especially useful if your application requires several seconds to initialize.

To add a startup image:

- Click on the **Startup Settings** tab of the **Settings** pane
- Click the **Select...** button next to the **Splash Image** textbox.
- Navigate to a BMP-format image to use as the startup graphic, and click **Open**.

If you wish to remove the current startup image, click the **Reset** button.

3.3.1 Transparency keying

Transparency keying allows the startup image to contain transparent regions. Transparencies can improve the visual effectiveness of your startup image.

To select the transparency key color:

- Click the **Select...** button next to the **Transparency key** label. This displays the transparency key selection dialog.
- Select the color which represents transparent regions in the startup image and click **OK**.

3.3.2 Previewing the startup image

To preview the startup image, press the **Preview** button. Previewing is particularly useful to assure that the transparency key has been set properly.

3.4 Compile and Add a Startup Shim

Micro Focus Desktop Containers Studio enables you to specify a compiled startup shim invoked prior to the startup file. Startup shims are used to perform customized licensing checks and other initialization tasks. The shim must conform to the Micro Focus Desktop Containers Studio interface in order to validate. For a more complete example startup shim, please view the topic "Example Startup Shim" in the Advanced Topics section.

The startup shim must compile with an **OnInitialize** method.

C-style startup shim signature:

```
typedef BOOL (__stdcall *FnOnInititialize) (LPCWSTR pwcsInitilizationToken);
```

The return value indicates whether virtual machine execution proceeds.

Methods are acquired via **::LoadLibrary** followed by **::GetProcAddress** calls.

For example:

```
LPCWSTR pwcsInitToken = "VendorSpecificToken";

HMODULE hShim = ::LoadLibrary("Shim.dll");

FnOnInititialize fnOnInit = (FnOnInititialize)::GetProcAddress(hShim,
"OnInitialize");

BOOL fResult = fnOnInit(pwcsInitToken);
```

Complete the following steps to add a compiled startup shim DLL:

- Select the **Startup Settings** tab in the **Settings** pane
- Choose the **Select** button adjacent to the **Startup Shim DLL** field.
- Navigate to a DLL-format file to use as the startup shim and select **Open**.

Use the **OnInitialize Parameter** field to specify parameters for your startup shim.

To remove the current startup shim select **Reset**.

3.4.1 Active Directory Binding

Micro Focus Desktop Containers Studio enables you to limit where an application will run, based on queries to an Active Directory Domain Controller.

- Required domain
  Choose the DNS Domain name that a computer must be a member of to run the application.

- Required group
  Choose the Active Directory security group that a user must be a member of to run the application. The Required Group is not case sensitive.

Note: Enabling this feature adds an Active Directory shim to the application container, which will run after user specified shims. Errors communicating with Active Directory are logged to debug output.

3.5 Compile and Add a Shutdown Shim

Micro Focus Desktop Containers Studio enables you to specify a compiled shutdown shim invoked prior to the application shutting down. Shutdown shims are used to perform custom actions before the application container shuts down. The shim must conform to the Micro Focus Desktop Containers Studio interface in order to validate.

The startup shim must compile with an **OnShutdown** method.

C-style shutdown shim signature:

```
typedef VOID (__stdcall *FnOnShutdown) (LPCWSTR pwcsShutdownToken);
```

Methods are acquired via **::LoadLibrary** followed by **::GetProcAddress** calls.

For example:

```
LPCWSTR pwcsShutdownToken = "VendorSpecificToken";

HMODULE hShim = ::LoadLibrary("Shim.dll");

FnOnShutdown fnOnShutdown = (FnOnShutdown)::GetProcAddress(hShim, "OnShutdown");

fnOnShutdown(pwcsShutdownToken);
```

Complete the following steps to add a compiled shutdown shim DLL:

- Select the **Startup Settings** tab in the **Settings** pane
- Choose the **Select** button adjacent to the **Shutdown Shim DLL** field.
- Navigate to a DLL-format file to use as the shutdown shim and select **Open**.

Use the **OnShutdown Parameter** field to specify parameters for your shim.

To remove the current shutdown shim, select **Reset**.

3.6 PROCESS CONFIGURATION OPTIONS

Micro Focus Desktop Containers Studio provides several options that control the startup of the primary and child processes. Options can be accessed by clicking the **Settings** button, then clicking the **Process Configuration** tab.

3.6.1 Command line arguments

By default, command line arguments specified by the user upon application container execution are passed to the application container startup executable. It is possible to override this behavior and specify a fixed set of command line arguments to be passed to the startup executable. For example, one can use this option to specify Java virtual machine behavior.

To specify an explicit command-line:

- Click on the **Settings** button
- Click on the **Process Configuration** tab
- Enter the command-line arguments in the **Command line** textbox.

Note that these arguments override any arguments that might be specified by the end-user.

3.6.2 Working directory

The working directory setting determines the active directory at the time the application container process is launched.

The **Use startup file directory** option sets the working directory to the directory of the application container startup file. In the case of a jukeboxed application, the working directory is set to the directory of the startup file specified on the jukebox command line.

The **Use current directory** option sets the working directory to the directory from which the application container is launched.

The **Use specified path** option allows an explicit working directory to be specified. The working directory specification can include environment and well-known root folder variables.

By default, the working directory is set to the directory of the startup file.

3.6.3 Application type

Windows applications may execute in either the GUI- or console-mode subsystems. If you have selected an executable startup file, Micro Focus Desktop Containers Studio will automatically configure the application container to execute in the same subsystem as the startup file. However, if you have selected a non-executable startup file, it may be necessary for you to manually override the application type. Most applications execute in the GUI subsystem.

To override the application type, select the appropriate mode from the **Application type** dropdown in the **Process Configuration** section of the **Settings** panel. The **Inherit** mode sets the application type based on the type of the startup file, if possible.

3.6.4 Target architecture

The **Target architecture** option matches the structure of the virtual environment to the desired host process architecture.

x86: Use this option for applications that were snapshotted on x86 systems. This option maps the **Program Files** directory to **C:\Program Files** on x86 systems or to **C:\Program Files (x86)** on x64 systems. .NET applications always run as 32-bit applications.

x64: Use this option for applications that were snapshotted on x64 systems. This option maps the **Program Files** directory to **C:\Program Files** on x64 systems. The **Program Files (x86)** directory is mapped to **C:\Program Files** on x86 systems and **C:\Program Files (x86)** on x64 systems. .NET applications run as 32-bit applications on x86 systems and 64-bit applications on x64 systems.

Any CPU: Use this option for .NET applications that are compiled to run on any CPU architecture. This option maps the **Program Files** directory to **C:\Program Files** on x86 systems and **C:\Program Files** on x64 systems. .NET applications run as 32-bit applications on x86 systems and 64-bit applications on x64 systems.

3.6.5 Environment variables

Some applications may depend on the presence of certain Windows environment variables in order to function properly. Micro Focus Desktop Containers Studio allows virtualization of environment variables to support such applications.

To add or modify virtual environment variables:

- Click the **Environment Variables...** button. This displays the **Environment Variables** dialog.
- Enter environment variable names and values into the environment variable list.

Most virtual environment variables overwrite any environment variables defined in the host environment. However, the special **PATH** and **PATHEXT** environment variables are always merged with the corresponding host environment variables.

Environment variables are automatically captured and merged during the snapshotting delta process. Therefore, it is generally unnecessary to manually configure environment variable settings.

3.6.6 Virtual services

Windows services are specialized applications that execute in the background and are typically responsible for providing system services such as database services, network traffic handling, web request processing, and other server functionality. Many applications install and require specific services in order to function properly.

Micro Focus Desktop Containers Studio fully supports virtualization of Windows services. To view or modify virtual service settings, press the **Virtual Services...** button. This displays the **Virtual Services** dialog.

The **Name** field specifies the internal name of the virtual service. For example, the Windows web server would use the name **w3svc**.

The **Friendly Name** field specifies the full display name of the service displayed to end users. For example, the Windows web server friendly name is **World Wide Web Publishing Service.**

The **Command Line** field specifies the full command line (including the service executable name and any parameters) used to launch the service.

The **Auto Start** flag indicates whether a virtual service automatically starts during application container startup, or whether the service must be launched manually or by the virtualized service control manager.

The **Keep Alive** flag indicates whether the virtual service process is automatically terminated when the primary application executable terminates, or whether the service (and, therefore, the host application container executable) continues to run until the service terminates itself.

Service installation and settings are automatically captured during the snapshotting process. Therefore, it is generally unnecessary to manually configure virtual service settings. The primary exception is the case of application containers that are intended to run as background worker services (for example, virtualized web servers); in this case, it is often required to explicitly enable the **Keep Alive** option.

3.6.7 Child processes

Some applications spawn new *child processes* during the course of their execution. Depending on the application container context, it may be preferable for such child processes to be created either within the application container, or outside of the application container in the host operating system.

Child processes include processes spawned to service COM local server requests.

Note: Child processes created outside of the application container will not have access to virtualized filesystem or registry contents. However, these processes will be able to access or modify host operating system contents, even if this would otherwise be forbidden by the application container configuration.

By default, child processes are created within the application container. To force child processes to be created outside of the application container, uncheck the **Spawn child process within virtualized environment** option. COM local servers can optionally be created within the application container context. To force COM local servers to be created outside of the application container, uncheck the **Spawn COM servers with virtualized environment** option.

Exceptions to the child process virtualization behavior specified by the **Spawn child process within virtualized environment** and **Spawn COM servers within virtualized environment** flags can be enumerated in the **Child Process Exception List.** Process names listed in the child process exception list behave *opposite to* the master child process virtualization setting. To edit the child process exception list, click the **Child Process Exception List** button. Process names may or may not include the process filename extension.

3.6.8 Read-only virtual environments

In some scenarios, it may be desirable to prevent the user from making any modifications to the virtual environment, including the virtual filesystem and registry.

To block all modifications to the virtual environment, check the **Virtual environment is read-only** option.

3.6.9 Automatic sandbox reset

The sandbox can optionally be configured to be reset automatically on application shutdown. An application publisher may want to do this to assure that any changes made to an application's settings are reverted when the application closes.

To enable the automatic sandbox reset feature, check the **Delete sandbox on application shutdown** option.

3.6.10 Shutdown process tree on root process exit

The **Shutdown process tree on root process exit** option enables the shutdown of all child processes when the root process exits.

Note: By default, the startup file is the root process. However, if a virtual service is specified in the application configuration file and is set to auto-start when the application is launched, the virtual service acts as the root process in the process tree.

3.6.11 Compress Payload

Both the application profiling and streaming processes require that packages be built uncompressed.

To build applications without compression, leave the **Compress Payload** option unchecked.

3.6.12 Startup executable optimization

The **Enable startup executable optimization** option attempts to launch the startup executable within the initial virtual machine process. This prevents the creation of a separate application process but may be incompatible with some applications.

3.6.13 Command-line arguments

Micro Focus Desktop Containers Studio supports command-line arguments of the form **/x[arg]** which modify application container behavior at run-time. In rare instances, these arguments may collide with command-line arguments designed for use by the application container.

To disable processing of these arguments, uncheck the **Enable MFDC command-line arguments** option.

3.6.14 Window class isolation

The **Enable window class isolation** option prevents the application container from viewing window classes that have been registered by external processes. For example, this option may be used to prevent interaction between virtualized and non-virtualized versions of the same application when the application checks for existing class registrations.

3.6.15 Enhanced DEP compatibility for legacy applications

The **Enhanced DEP compatibility for legacy applications** option provides compatibility for systems with Data Execution Protection (DEP) enabled. This configuration option is used primarily for application containers running on Windows 2003.

3.6.16 Enhanced DRM compatibility

The **Enhanced DRM compatibility** option provides additional compatibility with common DRM systems, such as Armadillo.

3.6.17 Trace process starts in debug output

The **Trace process starts in debug output** option sends a notification to **OutputDebugString** whenever a new process is started within the virtual environment. This notification takes the form of an XML snippet describing basic information, and it can be monitored with any debugging tool. The notification can also be monitored by a parent process within the virtual environment if a child process is being debugged.

3.6.18 Force read-share files

The **Force read-share files** option forces any file opened by any process within the virtual environment to do so with the **READ_SHARE** flag set. This option is used to overcome compatibility issues caused by sharing violations.

3.6.19 Always launch child processes as current user

By default, child processes launched by the virtual machine have reduced privileges. Enabling the **Always launch child processes as current user** option provides child processes with the same level of privileges as the virtual machine root process.

3.6.20 Emulate elevated security privileges

The **Emulate elevated security privileges** option forces an application to run as if it has elevated security privileges, even if the application does not. Enabling this option eliminates UAC security prompts for elevation and subsequent application crashes due to lack of elevated privileges.

3.7 CONFIGURING THE SANDBOX LOCATION

Depending on the configured isolation settings, certain edit and write operations may be redirected by the Micro Focus Desktop Containers engine into an *application sandbox*- a filesystem folder where isolated modifications are persisted. Typically, the sandbox is located in a folder or network share where the user has full read/write permissions, allowing sandbox contents to be accessed and modified by the end user without any authentication or User Account Control prompts.

3.7.1 Sandbox placement considerations

Please note the following recommended practices when configuring the sandbox location:

By default, the sandbox is placed in the "**@APPDATALOCAL@\Micro Focus\Sandbox\@TITLE@\@VERSION@"** folder, where the **@APPDATALOCAL@** token represents the local **Application Data** folder, and **@TITLE@**, and **@VERSION@** represent the application title and version respectively. The application title and version are configured in the **Properties** area. This location is the recommended default location for sandbox contents, as end users have full permissions to this location on standard Windows configurations. Note that distinct builds of the same application container use the same sandbox locations by default; you may want to modify this behavior if persisted user settings should not be preserved between application container updates.

When publishing a new version of an application container, direct the sandbox to the *same* location as the older version if you want user settings and data to be retained in the new version. Direct the sandbox to a *different* location (typically, by rolling the subdirectory version number forward) if you want user settings and data to be reset.

If deploying the application container on a USB device, place the sandbox in a subfolder of the **@APPDIR@** directory, which represents the location of the application container executable. This will have the effect of directing writes to the USB device. The recommended sandbox location for USB deployment is:

@APPDIR@\Micro Focus\Sandbox\@TITLE@\@VERSION@

If deploying the application container on an intranet file share, place the sandbox in a user-accessible subfolder on a shared network drive. The recommended sandbox location for intranet deployment is:

\\ServerName\ShareName\%USERNAME%\Micro Focus\Sandbox\@TITLE@\@VERSION@

Do not place the sandbox under any privileged folders, such as **@WINDIR@** or **@PROGRAMFILES@**. The application container will fail to execute properly if the Micro Focus Desktop Containers engine is unable to write to the sandbox location at runtime.

Environment variables may be referenced within the sandbox location by enclosing the variable between percent signs, i.e. **%VARIABLE%**.

3.7.2 Sandbox location variables

In addition to the standard root folder variables, the sandbox location can contain the following token variables:

@TITLE@: Product title

@PUBLISHER@: Product publisher

@VERSION@: Full version string, in dotted quad format

@WEBSITE@: Publisher web site

@BUILDTIME@ Application container build time, in a format similar to **2008.02.01T08.00.**

With the exception of the **@BUILDTIME@** variable (set automatically), these variables are based on the values specified in the **Properties** area of the **Settings** pane.

4 BUILDING MSI SETUP PACKAGES

Micro Focus Desktop Containers Studio includes the ability to generate Microsoft Windows Installer (MSI) setup packages to facilitate deployment of application containers. In addition to deploying the application container executable file to the host filesystem, Micro Focus Desktop Containers Studio-generated MSI packages also allow creation of desktop and Start Menu shortcuts, creation of shell file extension associations to application containerss, and Control Panel uninstallers for application cleanup.

This section describes configuration and build processes for MSI setup packages.

4.1 CONFIGURING PACKAGE INFORMATION

This section describes global MSI package configuration options. These options are located on the **MSI** root tree node of the **Setup** settings pane.

4.1.1 Setting the MSI package location

The MSI package generated by Micro Focus Desktop Containers Studio will be written to the file specified in the **Output Location** textbox. This textbox should contain the fully qualified name of the desired output file, including the output path and MSI file name.

By default, MSI packages are not automatically generated or updated when the application container is rebuilt.

To automatically update MSI packages after an application container is rebuilt, check the **Automatically generate MSI after successful application build** option.

Regenerating MSIs may significantly increase the time required to complete the build process. Therefore, Micro Focus recommends that this option be disabled during the application container development process.

It is also possible to manually force the MSI package to be regenerated. To manually build the MSI package, click the **Build MSI** button.

Note: You must build the application container executable *before* the MSI package may be generated. The **Build MSI** button will be disabled if the application container executable has not yet been built.

4.1.2 Specifying package metadata

MSI setup packages contain a package manifest describing the product's name, version (can only be numbers and periods), and manufacturer. To configure the MSI package metadata, enter the appropriate values in the **Product Info** area.

Note: The metadata published on the MSI package is distinct from the metadata published on the application container executable itself. To modify executable shell metadata, specify the appropriate metadata on the **Settings** pane.

4.1.3 Installation parameters

Installation parameters allow installation options such as install location and permissions parameters to be configured.

Applications may be installed either for the current user or for all users of the target device. To install the application for all users, check the **Install applications for All Users** option.

Note: Installing applications for all users requires privileged access to the host device. Do not enable this option if the MSI package is designed for use by end users with standard user permissions.

The **Application Folder** specifies the location where the application executable will be installed. The standard form is **[Application Data]\[Company Name]\[Product Name]**. These values are populated by the **Product Info** data by default.

In the event that a user runs the setup package on a device which already has a version of the application installed, the MSI package may be designed to update the existing application version or side-by-side install with the existing application version.

To automatically update existing versions, select the **Automatically upgrade earlier application versions** option; to use side-by-side installation, select the **Allow side-by-side versions of the same application** option.

Note: Building with the **Allow side-by-side versions of the same application** option enabled causes a new setup package GUID to be generated. Once a build is completed with this option enabled, previous installations will no longer be upgraded in place, even if you revert to **Automatically upgrade earlier application versions** mode.

4.1.4 Extended properties

MSI setup packages may also contain extended metadata, such as keywords, product author, product description, and publisher URL. To configure MSI package extended properties, click on the **Extended Properties** tree node in the **MSI** pane and enter the desired values.

4.2 CREATING DESKTOP AND START MENU SHORTCUTS

Desktop, Start Menu, and Send to shortcuts enable you to launch application containers directly from the Windows shell as if they were native applications. Shortcuts created during the snapshot process of a normal install will automatically be picked up in the configuration.

Complete the following steps to manually configure shortcuts for an application container:

- Select **Setup**.
- Expand **Shortcuts** and choose one of the following three destinations: **Desktop** (displays shortcuts which appear on the desktop); **Programs Menu** (displays shortcuts that appear in Start Menu > Programs); **SendTo** (displays shortcuts that appear when you Right-Click > Send To on a file or folder).
- Select **Add Shortcut** to add a shortcut to the selected destination.
- Enter the shortcut **Name**.
- Set the **Target** drop-down to the desired application entry point for the shortcut. This drop-down is populated with the startup file list, enabling shortcuts to quickly connect to jukebox entry points. Select **Edit Lis**t to make a quick change.
- Configure the **Arguments**, **Show As**, and **Description** fields.
- Select **Browse** to associate a shortcut with an icon. Locate the .ico, .dll, or .exe file that contains the icon and select Open. Choose the correct icon from **Icon Browser**.
- Select **OK** to finish adding the shortcut.

Select **Add Folder** to add a shortcut folder to the selected destination. Use folders to organize shortcuts. You can remove or modify a shortcut by selecting it from the tree view and choosing **Edit Shortcut** or **Remove**.

Application container shortcuts appear on the desktop, Start Menu, and Send to locations once the application is registered to the Microsoft Windows shell.

Note: For MSI installations, the Start Menu items appear in the current user's Start Menu or in all users' Start Menus, depending on the **Install Application For All Users** setting in the MSI installation parameters section. This setting is located under **MSI** in **Setup**.

4.3 CREATING FILE ASSOCIATIONS

File associations allow the appropriate viewer or editor application for a given file type to be automatically launched when the user double-clicks on a document in the Windows shell. For example, the **.doc** file extension might automatically launch a virtualized word processing application. File associations created during the snapshot process of a normal install will automatically be picked up in the configuration.

To create a file association:

- In **Setup**, Click on the **ProgIds** node and click **Add ProgId**.
- Enter a ProgId and Description in the **Create ProgId** dialog. File associations naming generally follows the convention **[Company Name].[Product Name].[Version]**.
- In the new ProgId, click **Add Extension** and enter an Extension and an optional MIME Type.
- In the new file extension, click **Add Verb** and enter a Verb, Command, and choose the Inherit behavior and Default.

Some common verbs are **open**, **edit**, **print**, and **view**. The **Verb** that is entered will be the text that is displayed when the user right-clicks on the file. When **Inherit** is checked, the behavior of the **Verb** will be controlled by the setup information in the virtual environment. When **Inherit** is unchecked, a **Target Startup File** and **Arguments** will need to be entered manually. The **Arguments** field should contain **"%1"** which is the full path to the file. When **Default** is checked, the **Verb** will be automatically executed when the file is double-clicked.

File association properties may be modified or deleted by selecting the appropriate **ProgId** in the **Setup** tree view and modifying the settings as appropriate.

5 Deploying Application Containers

This section describes several different methods for deploying applications built in Micro Focus Desktop Containers Studio.

5.1 Deploying using the Publish to USB feature

This section describes how to deploy application containers to USB storage devices using the **Publish to USB** feature.

5.1.1 Publishing applications containers to USB storage devices

The **Publish to USB** feature publishes application containers to USB storage devices. When the USB storage device is attached to a host system, the application container automatically registers the **Setup** information to the host shell environment. This information is automatically unregistered when the USB device is removed from the host system.

To deploy application containers on USB devices:

- Open an existing application container configuration.
- Attach a USB storage device to the host system.
- Click **Publish to USB**, select the USB storage device, and click **Publish**.
- After the application container is published to the USB storage device, click **OK**.

To use application containers that are published to USB storage devices:

- Attach the USB storage device to the host system.
- If prompted by AutoPlay, choose the **XUsb.exe** option. XUsb will then register the file associations and shortcuts associated with the application container.

Remove the USB storage device to unregister the application containers from the host system.

Note: If AutoPlay is disabled on the host system, open the USB storage device's contents and manually run **XUsb.exe**.

5.2 REGISTERING APPLICATION CONTAINERS IN THE WINDOWS SHELL

This section explains how to use the SpoonReg deployment tool via the Windows shell or command-line script to register and manage application containers built using Micro Focus Desktop Containers Studio.

5.2.1 Introduction to the SpoonReg registration tool

SpoonReg is a tool that provides a simple command-line interface for deploying application containers and managing the virtual desktop environment. Users and administrators can use SpoonReg to register application containers for a single user or, in the case of administrators, a group of users or devices.

After creating an application container with Micro Focus Desktop Containers Studio, it is often desirable to make the application Start Menu icons, shortcuts, and file associations available on the users' desktop. SpoonReg allows you to register Micro Focus application containers in the shell, creating all of the shell associations that would generally be created during a standard install process. Unlike performing an installation, however, registration and un-registration can be performed instantaneously.

SpoonReg also provides the ability to create, reset, and remove application sandboxes- virtual environment "bubbles" where the application containers reside. Sandbox management provides fine-grained control over application linking and intercommunication.

5.2.2 Registering application conainers using SpoonReg

SpoonReg provides a simple command-line interface for managing the virtual desktop environment. This section describes basic SpoonReg command-line syntax, including steps for registering, updating, and unregistering application containers.

5.2.3 Command-line syntax

The following naming conventions are used in this section:

Table 5-1: SpoonReg Parameters

Parameter	Description
AppSpec	The path (relative or fully qualified) to an application container executable or component built with Micro Focus Desktop Containers Studio. This may also be the URL to the XML configuration file of an application hosted on an Micro Focus Desktop Containers Server.
SandboxSpec	The name or path of a virtual sandbox.

5.2.4 Registering an application container

To register an application, use the command:

SpoonReg.exe AppSpec

This command creates all Start Menu items, desktop shortcuts, and file associations associated with the application container executable.

By default, registration will create a local cached copy of the application container executable and use the user's local profile as the sandbox location.

Note: The sandbox location specified during the application container build is ignored when registering applications using the SpoonReg tool.

5.2.5 Advanced registration options

Command-line parameters can be used to control the caching behavior and sandbox where the application container should be registered:

SpoonReg.exe[Options] AppSpec[@SandboxSpec]

Table 5-2: Advanced SpoonReg Parameters

Parameter	Description
/nocache	The application container executable will not be copied to the client machine. All shortcuts and file associations will point to the full path as given by **AppSpec**.
SandboxSpec	This parameter refers to the name and path to an existing sandbox. If this parameter is specified and a sandbox with that name exists, the application will be registered into that sandbox. (See the "Sandbox management" topic in this section for additional details.)

5.2.6 Updating registration settings

Application registration settings can be changed by re-executing the registration command with the desired options:

SpoonReg.exe[Option] AppSpec[@SandboxSpec]

Table 5-3: Updating SpoonReg Parameters

Parameter	Description
/nocache	Disable caching of the specified application (reverses the **/cache** setting).
/remote	Same as **/nocache**.
/cache [move]	Enable caching of the specified application (reverses the **/nocache** setting). The optional **move** flag deletes from the current location.
/local [move]	Same as **/cache**.
/checkforupdates	User will be prompted to update the application if updates are available (default behavior).

/noupdatecheck	User will not be prompted to update the application.
/noupdate	Application will not be updated (only valid in combination with **/noupdatecheck**).
/autoupdate	Application will be updated without prompting user (only valid in combination with **/noupdatecheck**).

5.2.7 Unregistering an application container

Unregistering an application container reverses the registration process, removing the application container, Start Menu icons, shortcuts, and file associations.

To unregister an application container, use the following command:

SpoonReg.exe /unregister AppSpec[@SandboxSpec]

It is also possible to unregister all applications with the single command:

SpoonReg.exe /unregisterall

5.2.8 Introduction to XLaunch

XLaunch is an internal application used to execute applications containers that have been registered to the desktop by SpoonReg, MFDC Server or the MFDC Console. All desktop shortcuts and file associations will reference the XLaunch application as the way to launch the application containers cached on the system.

5.2.9 Install location

Location when installed by SpoonReg:
@APPDATALOCAL@\Spoon\XLaunch\<version>\XLaunch.exe

Location when installed by MFDC Console (MFDC Server):
@APPDATALOCAL@\Spoon\Client\Components\<version>\XLaunch.exe

5.2.10 Command-line syntax

XLaunch.exe <Path to default.xclient> <RegistrationId> [<Shell Execute Info> | /XUninstall | /XUninstallQuiet]

Table 5-4: XLaunch Command-line Parameters

Parameter	Description
Path to default.xclient	This is the path to the XClient file where the app was registered.
RegistrationId	This is the uniquely identifying GUID of the application container specified on the SpoonReg command line using the /id parameter.
Shell Execute Info:	**verb**: The verb used to shell execute the application or

<verb> <path> [<additional parameters>]	document (e.g. open, edit, print). **path**: The path to the application or document to execute, may use path variables, like @PROGRAMFILES@, or not. **additional parameters**: Any additional parameters to pass to the application being executed.
/XUninstall	Uninstalls the application and informs the user when complete.
/XUninstallQuiet	Uninstalls the application with no user feedback.

5.2.11 Introduction to the XClient file

The XClient file is an XML file that provides information about applications that are registered to the user's machine.

5.2.12 File location

Applications registered with SpoonReg:
@APPDATALOCAL@\Spoon\Client\5\Default.xclient

Applications registered from the MFDC Console (MFDC Server):
@APPDATALOCAL@\Spoon\Servers\<server>\Users\[<username>|Anonymous]\Desktops\Default\Client\Default.xclient

5.2.13 XClient file format

Table 5-5: XClient File

Element	Description
ConfigReferences	List of all external configurations from which this configuration inherits.
KnownServers	List of any portals the MFDC Console has accessed. Also contains a **recentServer** element that lists the last portal the console was logged in to. The parameters for each server are: **name**: Name of the portal. **portalUrl**: Web address of the portal. **portalSecureUrl**: Web address of the portal if SSL is enabled. **syncUrl**: Web address of the synchronization service.
Sandboxes	Every time a new application is launched, a sandbox is created. This element lists the location of each sandbox for the user account.
Folders	This element contains information for the My Documents, Desktop, Music, Pictures, or Videos folders the user has synchronized to their account.
SyncSettings	Contains the settings used by the synchronization service. Sub-elements of SyncSettings are: **BandwidthManager**: Controls the **Upload Throttling** feature of synchronization. **SyncUser**: Contains the name, server, and url of the primary user's

	synchronization service.

5.3 CLIENT PROFILES

The SpoonReg command can be applied to the **Local**, **All Users** or **Roaming** Windows profiles. These profiles correspond to the user profiles available on the Windows operating system. It's recommended that the SpoonReg command is applied to the **All Users** profile whenever possible.

5.3.1 The Local profile

Each user on a Windows device has a local user profile. Any changes to the local profile affect only that user on that device.

The Local profile is the default profile used by SpoonReg if no profile is explicitly specified on the SpoonReg command line.

5.3.2 The All Users profile

Each device has a single **All Users** profile. Any changes made to the **All Users** profile affect all users on the device.

To register an application to the **All Users** profile, execute the SpoonReg command with the **/allusers** command-line flag.

You must have administrative permissions on the device to register applications to the **All Users** profile.

Note: If the **/allusers** flag is used when registering an application, it must also be used when unregistering.

5.3.3 The Roaming profile

Each user in an Active Directory environment can have a roaming profile which is mirrored to other machines according to directory policy. Typically, the roaming profile is stored on a network server and is available from all devices on a network.

To register an application to the **Roaming** profile, execute the SpoonReg command with the **/roaming** flag.

Note: There is no roaming profile for **All Users**. Therefore, the **/roaming** flag has no impact when used in conjunction with the **/allusers** flag.

Note: Because SpoonReg is applied to specific user profiles, it cannot be used for the **LocalSystem** account (has no associated profile).

5.4 SANDBOX MANAGEMENT

The SpoonReg tool allows creation and management of one or more virtual environment sandboxes.

A *sandbox* contains all of an application container's isolated data and settings as determined by the application container's isolation configuration settings. Applications registered to the same sandbox can view and modify each others' data and settings.

By default, all applications are registered into a single default sandbox named **Default**. In some cases, it may be desirable to group related application containers into a sandbox that can be treated as a single management unit. When a sandbox is reset, all of the application content and data stored in that sandbox is purged and reverts back to the default state.

5.4.1 Creating a sandbox

If no sandbox is specified during registration, the application will be registered to the default sandbox (**Default**).

To create an additional sandbox, use one of the following commands:

SpoonReg.exe [Profile] /create [SandboxName] [SandboxPath]
SpoonReg.exe [Profile] /c [SandboxName] [SandboxPath]

If no path is provided, a default path is created under the **AppData** folder under the specified profile.

5.4.2 Resetting a sandbox

Resetting a sandbox reverses all changes made to the sandbox, including any changes to data or settings made by the user. This restores all applications registered to the sandbox to their default state.

To reset a sandbox, use one of the following commands:

SpoonReg.exe [Profile] /reset [SandboxSpec]
SpoonReg.exe [Profile] /r [SandboxSpec]

If a **SandboxSpec** is not supplied, the default sandbox is reset.

5.4.3 Deleting a sandbox

Deleting a sandbox removes all applications, data, and settings from the sandbox.

To delete a sandbox, use one of the following commands:

SpoonReg.exe [Profile] /delete [SandboxSpec]
SpoonReg.exe [Profile] /d [SandboxSpec]

If a **SandboxSpec** is not supplied, the default sandbox will be reset (the default sandbox cannot be deleted). Any applications registered to the deleted sandbox will be moved to the default sandbox.

5.4.4 Moving a sandbox

You can use SpoonReg to move the sandbox location to a given path.

To move a sandbox:

SpoonReg.exe [Profile] /move [SandboxSpec] [SandboxPath]

5.5 Deploying in Active Directory environments

This section describes how an organization using Microsoft's Active Directory can leverage that infrastructure with the SpoonReg tool to deploy Micro Focus application containers to their users.

5.5.1 Active Directory

Active Directory allows the network administrator to manage users and groups within an organization. Many organizations use Active Directory to manage their network services. By combining Active Directory with SpoonReg, administrators can deploy application containers easily and reliably to one or more users in their organization.

5.5.2 SpoonReg

SpoonReg manages the virtual desktop environment for a given user by registering and unregistering application containers. A typical SpoonReg command to register an application such as Firefox would be:

\\VirtualAppServer\Tools\SpoonReg.exe \\VitualAppServer\Apps\Firefox.exe

This represents the basic functionality of SpoonReg which would copy the application container executable, create Start Menu items and desktop shortcuts and setup file associations.

5.5.3 Using SpoonReg with Active Directory

Organizations tend to manage a group of users rather than single users one at a time. By combining Active Directory with SpoonReg you can manage the virtual environment for a user, group, or organizational unit.

Note: In order to manage application containers using Active Directory, the users must have access to a shared network drive where the application container executable files exist. This can be specified by a full UNC path, or by using a mapped network drive.

5.5.4 Scenario 1: Linking an organizational unit (OU) to a group policy object (GPO)

Active Directory offers many ways to manage network services for an organization. Here we are going to look at how we can use an OU in combination with a GPO to manage the application container environment for a set of users.

5.5.5 Organizational unit (OU)

In Active Directory, OUs are containers where you can place users, groups and other OUs. Using these containers the administrator can create a structure that models the hierarchical or logical structures within the organization. By setting up the OUs we can isolate the different groups of users that will receive their own set of application containers. Some examples would be Accounting, Sales, Marketing, etc.

5.5.6 Group policy object (GPO)

GPOs are a way of applying a set of rules and features to a targeted set of users. Typically GPOs handle security, application installation, logon/logoff scripts, Internet Explorer settings and

more. A GPO generally gets applied when a user logs on to a given domain. Based on their profile, various GPOs will be applied.

5.5.7 SpoonReg and the GPO

For the purposes of application container deployment we can configure the GPO to run the necessary SpoonReg commands to register a given set of application containers. You can create and edit GPOs using the **Group Policy Object Editor**. This comes as an add-on to Microsoft's Active Directory. The simplest way to deal with the application container management is by creating a logon script that registers the application containers for a particular group in the organization. You would do this under **User Configuration > Windows Settings > Scripts** in the **Group Policy Object Editor**. You might add the following logon script for the accounting group:

```
\\VirtualAppServer\Tools\SpoonReg.exe \\VirtualAppServer\AllVirtualApps\Excel.exe

\\VirtualAppServer\Tools\SpoonReg.exe
\\VirtualAppServer\AllVirtualApps\Firefox.exe

\\VirtualAppServer\Tools\SpoonReg.exe
\\VirtualAppServer\AllVirtualApps\AcrobatReader.exe
```

Whereas you might add the following for the graphic design group:

```
\\VirtualAppServer\Tools\SpoonReg.exe
\\VirtualAppServer\AllVirtualApps\AdobeIllustrator.exe

\\VirtualAppServer\Tools\SpoonReg.exe
\\VirtualAppServer\AllVirtualApps\Firefox.exe

\\VirtualAppServer\Tools\SpoonReg.exe
\\VirtualAppServer\AllVirtualApps\AcrobatReader.exe
```

5.5.8 Linking the GPO to the OU

Once you have set up the OU and the GPO, you simply have to link them through the **Group Policy Object Editor**. After they are linked, any users that are put into that OU will have the linked GPOs applied when they logon. All of their application containers will appear when they logon to any machine on the domain where the GPO applies.

Note: SpoonReg has many other capabilities that can be applied in the GPO logon script, such as unregistering applications or registering applications to a specific sandbox.

5.5.9 Scenario 2: Using Startup Scripts with GPO's to deploy application containers

This section will describe using Startup Scripts to deploy application containers to the **All Users** profile on a host system.

5.5.10 Create a GPO with the Startup Script

For the purposes of application container deployment we can configure the GPO to run the necessary SpoonReg commands to register a given set of application containers to the **All Users** Profile. This will allow any user who logs into a host system to have access to the registered application containers.

Start by creating a new GPO with the **Group Policy Management Console (GPMC):**

- Open the **GPMC**
- Right-click on the OU that you want to link a GPO to
- Select **Create and Link a GPO Here** and give it a name
- Right-click the GPO and select **Edit.** This will open the **Group Policy Object Editor**
- Navigate to **Computer Configuration > Windows Settings > Scripts**
- Open the **Startup** item
- Click **Show Files**
- In the directory that is displayed by the **Show Files** button, create a **.bat** file. This file will serve as the **Startup Script** that will deploy your application containers to all the computers in the specified OU.

Use these samples as a guide to create the startup script:

Sample 1. Register application containers to the default sandbox in the **All Users** profile:

```
\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\Excel.exe

\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\Firefox.exe

\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\AcrobatReader.exe
```

Sample 2. Register application containers to the specified sandbox in the **All Users** profile:

```
\\VirtualAppServer\Tools\SpoonReg.exe /allusers /c sandboxname

\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\Excel.exe

\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\Firefox.exe

\\VirtualAppServer\Tools\SpoonReg.exe /allusers
\\VirtualAppServer\AllVirtualApps\AcrobatReader.exe
```

Once the Startup Script is created, add it to the GPO:

- Navigate to **Computer Configuration > Windows Settings > Scripts** in the **GPMC**
- Open the **Startup** item
- Click **Add**
- Click **Browse**
- Select the **Startup Script** that was created
- Click **Open**

- Click **OK**
- Click **OK**

5.6 Deploying using MSI setup packages

Micro Focus Desktop Containers Studio allows application containers and components to be deployed using legacy MSI setup package technology.

5.6.1 Create MSI setup packages directly within Micro Focus Desktop Containers Studio

Micro Focus Desktop Containers Studio can be used to create standalone MSI packages directly within the Micro Focus Desktop Containers Studio environment. Generated MSI setup packages can include Start Menu items, desktop shortcuts, file associations, and other custom shell integration behaviors.

Deployment using generated MSI packages is appropriate in situations where existing MSI package deployment mechanisms are in place, or for deploying applications with shell integration without the SpoonReg tool.

Application container and component shell integration settings are shared between MSI- and Virtual Desktop-based deployment, enabling easy migration between deployment models.

Refer to the section “Building MSI setup packages” for more information on the creation of MSI setup packages.

5.6.2 Import legacy MSI setup packages

Micro Focus Desktop Containers Studio supports one-click import of legacy MSI setup packages into the Micro Focus Desktop Containers environment. Following import, the application can be customized and deployed as an application container.

5.7 DEPLOYING USING MICROSOFT TERMINAL SERVICES REMOTEAPP

This section describes how to deploy Micro Focus application containers using Microsoft Windows 2008 Terminal Services RemoteApp server.

5.7.1 Terminal Services RemoteApp

Terminal Services RemoteApp is a server-side program that provides end-users remote access to applications on a terminal server. Applications configured with TS RemoteApp appear as though they are running locally on the user's machine. End-users can run RemoteApps side-by-side with local programs or other RemoteApps.

To utilize application containers created in Micro Focus Desktop Containers Studio with Microsoft TS RemoteApp:

- On the TS RemoteApp server, open the TS RemoteApp Manager and choose **Add RemoteApp Programs** by right-clicking inside the **RemoteApp Programs** list or through the **Action** drop down menu.
- Click **Next** in the **RemoteApp Wizard**.
- Click **Browse** and select the application container executable.
- After the application container is added to the list, select it and click **Properties**.
- If the application container has multiple Startup Files, configure the RemoteApp Program **Name** and **Alias**. If this is not done, the TS RemoteApp server will not distinguish between the separate applications in the suite.
- Still in the **Properties** window, select **Always use the following command-line argument**, enter in the **Trigger** for the Startup File that is to be executed, and click **OK**.
- In the **RemoteApp Programs** list, right-click the program that you added and choose **Create .rdp File**. There are no special requirements for the .rdp files.

If there are multiple Startup Files, repeat these steps for the other applications in the suite and deploy the shortcuts on the host systems.

5.8 Deploying to ZENworks Configuration Management

This section describes ZENworks Configuration Management Startup and Bundle Publishing features. All options described in this section can be found in the **ZENworks** panel.

5.8.1 ZENworks Startup

Application containers built with Micro Focus Desktop Containers Studio can be configured to require the ZENworks Configuration Management Agent to be installed on the host device executing the application container. By default, application containers built with Micro Focus Desktop Containers Studio do not require the ZENworks Configuration Management Agent.

To require the ZENworks Configuration Management Agent to be installed on the host system, check the **Require ZENworks Configuration Management Agent to be installed on workstation executing the virtual application** checkbox in the **ZENworks** panel.

When application containers built with Micro Focus Desktop Containers Studio require the ZENworks Configuration Management Agent to be installed, they can also be configured to restrict execution to a specific ZENworks Configuration Management Zone.

To restrict application container execution to a specific ZENworks Configuration Management Zone, check the **Only allow devices registered in specific zone to execute the application** checkbox in the **ZENworks** panel. When prompted, enter the **ZENworks Server Address**, click **Connect**, enter the appropriate **Username** and **Password**, and click **OK.**

When specifying the **ZENworks Server Address**, it is necessary to use the secure HTTP prefix (**https://**) in order to connect to a secure ZENworks Configuration Management Server.

When specifying the **ZENworks Server Address**, you must also use a port number suffix (e.g.**:81**) if the ZENworks Configuration Management Server has been configured to use a custom port number.

5.8.2 ZENworks Bundle Publishing

Micro Focus Desktop Containers Studio can publish application container bundles directly to a ZENworks Configuration Management Zone. Note that if an application container has been built with ZENworks Configuration Management Zone restriction, the application will not execute on devices outside of the restricted zone.

To manually publish an application container bundle directly to a ZENworks Configuration Management Zone:

- Select **MSI** or **Executable** for the **Project Type**. The MSI or Executable must be built before publication can occur.
- Click **Select Zone** under the **ZENworks Bundle Publishing** pane.
- When prompted, enter the **ZENworks Server Address** and click **Connect**.
- Log in to the ZENworks Server with the appropriate **Username** and **Password** and click **OK**.
- Assign a **Bundle Name**.

- Assign a **Bundle Folder**. The **Bundle Folder** is the location in the bundle hierarchy where the bundle will be published.
- Assign a **Bundle Group**. The **Bundle Group** is the name of the management group where the bundle will be added.
- Assign a **Destination Path**. The **Destination Path** is the location on the host system where the bundle will be installed and must be resolvable by the host system. This option will only be available if the **Project Type** is **Executable**.
- Click **Publish Now** to publish the bundle to the ZENworks Configuration Management Zone.

Additional options:

- To change the credentials used to publish the bundle, click the **Change Credentials** button and enter the new credentials (**Username** and **Password**).
- To store the credentials in the **XAPPL** file, check the **Store Zone credentials in .xappl file** checkbox. Using this option eliminates the credentials prompt when publishing the bundle after the application container has been built. This is a potential security risk as it will store the ZENworks credentials in the **XAPPL** file as plain text.
- To automatically publish future application container bundles to the selected ZENworks Configuration Management Zone after a successful build, check the **Automatically publish application as ZENworks bundle after successful build** checkbox.

Note: Application configurations that have more than one startup file will create a bundle for each startup file that has a unique Trigger value defined. The name of each bundle will be <Bundle Name> & <Trigger>.

5.9 Deploying using the Publish to MFDC Server feature

This section describes how to deploy application containers to MFDC Server using the **Publish to MFDC Server** button in Micro Focus Desktop Containers Studio.

5.9.1 Publish to MFDC Server

There are some basic requirements for using the Publish to MFDC Server feature:

a) The MFDC Server must be configured to use login authentication and there must be at least on user in the Server Administrators group.

b) A startup file must be specified in the Micro Focus Desktop Containers Studio configuration.

After completing the application build process, click the Publish to MFDC Server button located on the ribbon bar. In the login form, enter the URL for the MFDC Server and login as a server administrator. On the next screen confirm the application details and click OK. This will upload the application container to the MFDC Server and create a new application version.

The new version will not be visible to users until the "Publish" flag is set. To make the application visible, login to the MFDC Server administrator site, go to the application version detail page, check **Publish** and click **Save**.

Note: If the MFDC Server has more than one directory service, the directory prefix must be specified as part of the username. See the Micro Focus Desktop Containers Server user guide for more details on logging in.

Note: Publishing to MFDC Server may fail over a proxied connection if DNS is disabled. To solve this, make a temporary change to the Web Address of the MFDC Server and use the IP address of the machine. Be sure to change the Web Address back to the original value after publishing.

Note: If having connection issues due to SSL server configuration, see the Micro Focus Desktop Containers Server User Guide for more information.

6 WALKTHROUGHS

This section provides step-by-step instructions for using Micro Focus Desktop Containers Studio in common scenarios.

6.1 MANUALLY CONFIGURING A SIMPLE APPLICATION CONTAINER

This section provides a walkthrough of manual configuration for a simple application container in Micro Focus Desktop Containers Studio, based on the Windows Notepad application. In general, manual configuration should only be performed by experienced software developers creating internally developed software applications.

Complete the following steps to add Microsoft Windows Notepad to the application container configuration:

- Select Filesystem.
- Open the Filesystem > System Drive > System32 tree node.
- Select Add Files.
- Navigate to the System32 folder under your Microsoft Windows installation directory, usually located at C:\Windows\System32, and double-click on notepad.exe.
- Right-click on notepad.exe from the display on the right and select Set as Startup File.
- For Microsoft Windows Vista and 7: While the System32 node is still selected, choose New Folder and set the name to en-us.
- Set the isolation for the en-us folder to Merge.
- Open the en-us folder node and select Add Files to add the notepad.exe.mui file, which is required by the Notepad application. The file is located in the System32\en-us folder inside your Windows installation.

Complete the following steps to add a text file to the application container configuration:

- Using Notepad, create a file called hello.txt containing the text "Hello world" and save it to your desktop folder.
- In Micro Focus Desktop Containers Studio, open the Filesystem > Current User Directory > Desktop node and select Add Files to add the hello.txt file.

Complete the following steps to build a application container from your configuration:

- Select the Virtual Application tab to display the application container settings.
- Verify that the Startup File points to notepad.exe. The startup file indicates which file to execute when a application container is started by the user.
- Select Browse for the Output File setting, navigate to your desktop folder and Save. The output file indicates where to save the application container executable.
- Choose Build. Micro Focus Desktop Containers Studio will display a status dialog while it builds your application container. When it finishes, select OK.

To launch the newly built virtual Notepad, navigate to your desktop using Microsoft Windows Explorer and double-click on notepad.exe.

Complete the following steps to verify that you are inside the application container:

- Using Windows Explorer, delete the hello.txt file from your desktop folder.
- Launch the virtual Notepad application and navigate to File > Open.
- Navigate to the Desktop folder and verify that the hello.txt file is still present. This occurs because the virtual Notepad application is using the virtual system, which includes the hello.txt file that was added in the build process. You can open and view hello.txt exactly as if it were a real file in the physical filesystem.

6.2 BUILDING OPENOFFICE VIA SNAPSHOT PROCESS

This walkthrough describes creation of OpenOffice as an application container using the snapshot, jukebox, and setup features of Micro Focus Desktop Containers Studio. Although the examples used in this walkthrough are for OpenOffice, the processes described can be used to virtualize and configure almost any application.

6.2.1 Snapshotting

The snapshot process consists of two phases -- the "before" snapshot and the "after" snapshot. The before snapshot takes an inventory of all files and settings that are installed on the computer. The after snapshot is taken after the application being virtualized is installed. The contents of the after snapshot are compared to the before snapshot to determine all changes that were made to the host system during installation. The after snapshot also copies the new or modified files to a snapshot directory specified by the user.

Since it is required to install and capture all application dependencies during this process, it is important that snapshotting be performed on a clean Windows machine. This guarantees that all dependencies will be included in the installation and captured by the snapshot process.

To snapshot the OpenOffice installation:

- Capture the before snapshot by clicking **Capture Before** on the Micro Focus Desktop Containers Studio ribbon bar. The snapshot process may take a few minutes.
- Install OpenOffice and all of its necessary dependencies.
- Capture the after snapshot by clicking **Capture and Diff**. Micro Focus Desktop Containers Studio will prompt for the directory where the snapshot files are to be stored. This directory is usually located on an external PC or network share.

6.2.2 Saving the application container configuration

It is important to save the application container snapshot in its original state in case errors are introduced during customization and optimization. This also allows the application to be tested and modified without the need to re-snapshot after each iteration.

To save the application container configuration:

- Open the configuration menu and click **Save Configuration** or **Save Configuration As**.
- Micro Focus Desktop Containers Studio configuration files are stored in the **XAPPL**file format. The **XAPPL** file does not contain the filesystem content. The filesystem content is stored in the **Files** directory which is created during the "after" snapshot.

Note: The **XAPPL** file uses relative paths to identify snapshot files. Therefore it is required that the **Files** directory and the **XAPPL** file be located in the same directory.

6.2.3 Configuring the Application Container

Once the application snapshot has been created, the configuration can be modified or optimized depending on the desired behavior. For example, OpenOffice is a suite of applications; therefore program entry points need to be setup so each application can be run individually.

The following will need to be configured to get OpenOffice to function as an application container:

- Set the **Output File**
- Configure the **Startup Files** (Jukeboxing)
- Configure the **Setup** options
- ***(Optional)*** Remove unused installation files

6.2.4 Setting the output file name

The **Output File** is the name of the virtual executable file or **SVM** that is created by Micro Focus Desktop Containers Studio.

To set the **Output File**, click **Browse** next to the **Output File** field and assign it a file name.

6.2.5 Startup files (Jukeboxing)

The **Startup File** settings identify the specific executable files that can be started by the application container. These can be started by executing the application container, from a command prompt, or from the shortcuts.

To configure the **Startup Files**:

- Select **Virtual Application**.
- Locate the **Output** group and select the **Multiple** to open the **Startup Files** window.
- The **Start Files** window displays all entry points for the application. If an entry point is missing for your application, you can add one by:
    - Using **Start Files**, enter the **File** location on the first line and select **Enter**.
    - Using **Filesystem**, locate the target file. Right-click on the file and select **Add to Startup Files**.
- **Command Line** enables you to add parameters for each **Startup File**.
- **Trigger** sets the command-line parameter that specifies which **Startup File** to launch.
- Note: each **Startup File** must have a unique value for its **Trigger** field.
- Select **Auto Start** for the **Startup File** that should run by default if no **Trigger** is passed to the application container.

6.2.6 Setup options

When creating application containers for suites such as OpenOffice, the user may want to create a setup file. Setup files generated by Micro Focus Desktop Containers Studio are created using the MSI setup file format. Micro Focus Desktop Containers Studio setup files only install the application container and create file associations and shortcuts. No other installation functions are performed.

To create a setup file for the application container, the **Output Location**, **Product Info**, **Installation Parameters**, **Shortcuts**, and **File Associations** need to be configured.

- The **Output Location** defines where the setup file is created.
- The **Product Info** is the metadata that will be associated with the setup file. The Product Info is displayed in the **Add/ Remove Programs** window. It is recommended that this information be accurate to avoid confusion.
- The **Installation Parameters** control how the application container will be installed on the host system.
- **Shortcuts** allow the end user to launch the application directly from the Windows Start Menu or Desktop.

6.2.7 Output Location

To define the Output Location for the application container setup file:

- Click **Browse** next to the **Output Location** and assign it a file name.
- Check the **Automatically generate MSI after successful build** checkbox if the setup file should be created automatically after the build process.

6.2.8 Product Info

The following fields specify metadata associated with setup files and is visible in the Add/Remove Programs window. These fields are accessed from the MSI tree node under Setup.

- **Product Name:** The name of the application.
- **Product Version:** The version of the application.
- **Company Name:** The name of the company that made the application.

6.2.9 Installation Parameters

To install the application container for **All Users**, check the **Install application for All Users** check box. If checked, this option requires administrative privileges on the target system.

To automatically update existing versions, select the **Automatically upgrade earlier application versions** option. This option will update previous versions of this virtual executable.

To use side-by-side installation, select the **Allow side-by-side versions of the same application** option.

Note: The **Company Name\Product Name** needs to be entered in the **Application Folder** field. If neither is entered, the application container will be installed under folders named "**Company Name\Product Name**".

6.2.10 Creating Shortcuts

To create shortcuts that open specific application within the OpenOffice suite:

Click **Add Shortcut** and assign it a **Name**, assign it a **Target**, select an **Icon**, and enter any arguments that need to be passed to the specific application.

Folders that are created by the setup package can also be setup in this pane.

6.2.11 File Associations

To create File Associations**:**

Click on the **Add ProgId** button and enter the **ProgId** and a **Description** of the file association.

Click **Add Extension** and enter the file extension and **MIME Type** (if necessary) to the **ProgId**. If a verb is needed for the file extension, click **Add Verb** and enter the necessary information.

6.2.12 Removing unnecessary files from the snapshot

During the installation process for many applications, temporary installation files are created. While these files are necessary for the installation, they are not required for the application container to run.

In OpenOffice, the installation files are created in the same directory that the install executable was executed from.

To remove these files from the snapshot:

- Open the **Filesystem** pane
- Navigate to the location of the installation files, right-click the folder, and select **Delete.**

Note: For OpenOffice, removing the installation files from the snapshot may reduce the output application container binary size by up to 151 MB.

6.2.13 Build and Test

After the application container has been configured, customized, and optimized, it can now be built and tested.

To build the application container:

- Click on the **Build** or the **Build and Run** button. This will create application container binary file where the **Output File** is defined. The build process may take a few minutes.

To test the application container:

- Execute the application container binary file. The OpenOffice splash screen will display and the OpenOffice Quickstarter will open.
- Execute the application container binary file from a command prompt with the "**swriter**" Trigger. For example, run "**openoffice.exe swriter**" from a command prompt and the OpenOffice Writer application opens instead of the Quickstarter.

To test the Setup options:

- Execute the setup file on a system without OpenOffice installed. The application container, shortcuts, and file associations will be installed on the host system.
- Open the OpenOffice Writer Program from the start Menu shortcut.
- Open a file that is associated with the OpenOffice Writer program.

7 BEST PRACTICES

This section describes various best practices for making use of Micro Focus Desktop Containers Studio. Note that these methods of use are all optional.

7.1 BEST PRACTICES FOR SNAPSHOTTING

The following practices are recommended for optimal use of the snapshotting feature.

7.1.1 Machine configuration

- Perform snapshotting on a clean machine: Snapshotting on a clean machine assures that all dependencies will be installed by the application setup. Installing on a machine with existing components may cause dependencies to be inadvertently included in the "before" snapshot and therefore excluded from the final application container output.

- Use snapshotting in conjunction with whole-machine virtualization: Configuring a clean machine using a whole-machine virtualization tool such as Microsoft Virtual PC and saving a "before" snapshot based on this image allows many distinct application containers to be snapshotted in rapid succession by reverting the whole-machine virtual state.

- Snapshot on the earliest operating system variant you expect to target: Most applications can be successfully configured by snapshotting on the earliest (least common denominator) base operating system to be targeted. A small number of applications may require multi-platform snapshotting for successful deployment across all operating system variants.

7.1.2 Snapshot process

- Save the "before" snapshot: Saving the snapshot assures that you need only take the "before" snapshot a single time. It may be necessary to restart the target application during the installation process, in which case you will need to reload the “before” snapshot prior to capturing the “after” snapshot.

- To prevent problems with open file handles, it is recommended that processes and services that were started during installation are stopped prior to capturing the “after” snapshot. In most cases it is sufficient to exit the target application, unless the application installs services, such as Microsoft SQL Server, in which case the services should also be stopped prior to taking the “after” snapshot.

- Clean up your image: While Micro Focus Desktop Containers Studio automatically excludes many unnecessary files and registry keys, snapshotting often picks up many unnecessary items. If you have adequate technical understanding to do so, you may significantly reduce application container size by manually removing unnecessary items from the snapshot delta.

7.2 CAPTURING UPDATES TO AN APPLICATION VIA SNAPSHOT PROCESS

Application container updates can be captured within Micro Focus Desktop Containers Studio via the snapshot process.

To capture an update via snapshot process:

- Install the original native version of the application you wish to update on a clean machine.
- Click **Capture Before** to snapshot the original version of the app.
- Install the necessary updates to the native application.
- Click **Capture and Diff** to create the "after" snapshot of the app. This will capture the deltas between the original version and the updated version.
- Make sure the Project Type is set to **Component**, then click **Build** to create the **SVM** file.

This process will only capture the changes between the original EXE and the installed updates. The resultant **SVM** file can then be applied to the original virtual executable (for more information on updating **SVM** files, refer to the topic "Capture Updates to an Application Via Snapshot" in the "Advanced topics" section).

7.3 USING A PIPELINE BUILD PROCESS

Micro Focus recommends that certain steps in the application container build process be performed on multiple operating systems in order to ensure maximum compatibility and performance. Because many of these steps are time-consuming, the process can move more efficiently if the steps are performed on multiple machines in parallel. This process is referred to as “pipeline building” and is particularly useful when creating a high volume of application containers.

For each application built, Micro Focus highly recommends using three separate snapshot machines in parallel (Windows XP, Windows 7, and Windows 10), connected via local network.

The following steps should be performed on each machine in the “pipeline”:

7.3.1 Capturing the initial application container configuration via snapshot

The snapshot process is covered in detail in the “Snapshotting Applications” topic of the “Getting Started” section.

The snapshot process should be performed on each machine in the pipeline in order to create the multiple **XAPPL** files to be later merged into a single **XAPPL** file.

Each configuration should be saved in a way that denotes the operating system it was created on (E.g. **<Application><Version>Snapshot_<OS>.xappl**), and copied to a network accessible folder.

The relative paths to files in the folder will be queried when the configurations are merged into a single **XAPPL** file. This process- called “platform merge”- can be done on a single machine and is covered in detail in “Platform Merge” topic in the “Advanced Topics” section.

7.3.2 Profiling the application

The profiling process involves Micro Focus Desktop Containers Studio observing normal user behavior during short application container usage sessions, and creating transcripts- or *profiles*- of the behavior. The process is covered in detail in the “Creating application streaming models” topic in the “Advanced Topics” section.

The profiling process occurs after a merged **XAPPL** file has been created. It should be performed on each machine in the pipeline in order to create the multiple transcripts that will later be used to build a streaming model of the application container.

Each transcript should be saved in a way that denotes the operating system it was created on (E.g. **<Application><Version>Transcript_<OS>.xt**), and copied to a network accessible folder. It is recommended that multiple profiles are created for each machine in the pipeline (for more information on building streaming models using transcripts, refer to the “Creating application streaming models” topic in the “Advanced Topics” section).

7.4 SNAPSHOTTING INTERNET EXPLORER

Internet Explorer snapshot compatibility mode should be used when snapshotting Internet Explorer application setups. This mode ensures a more complete capture of the Internet Explorer portions of the registry and filesystem, whereas a normal snapshot process ignores those parts of the registry and filesystem.

To use Internet Explorer snapshot compatibility mode, click the **Options** menu above the ribbon bar in Micro Focus Desktop Containers Studio, and select **Internet Explorer snapshot compatibility mode** from the dropdown.

8 ADVANCED TOPICS

This section deals with advanced topics you may encounter while using Micro Focus Desktop Containers Studio.

8.1 Customizing the Micro Focus Desktop Containers Studio interface

This section describes Micro Focus Desktop Containers Studio interface customization options. All options described in this section can be found under the **Options** menu item.

8.1.1 Proxy settings...

Micro Focus Desktop Containers Studio uses the Internet to check for product updates and download update packages. If your computer is located behind a firewall, it may be necessary for Internet access to take place through a proxy server. By default, Micro Focus Desktop Containers Studio will use the default Internet settings configured on the host machine. In some circumstances, however, it may be necessary for you to manually configure the proxy server settings.

To manually configure proxy settings, select the **Proxy settings...** option from the **Options** menu. Provide the proxy server address, the server port and the type of authentication, if any, the proxy server uses. Select the 'Bypass proxy server for local addresses' option to bypass the proxy server when accessing resources located on the local network. Please contact your network administrator if you need assistance configuring the proxy settings.

8.1.2 Automatically detect associated runtimes and components

By default, Micro Focus Desktop Containers Studio will scan the virtual filesystem at build time and verify that the current configuration includes all available runtimes and components associated with file types contained in the virtual filesystem. This behavior is recommended to assure maximum application container reliability.

If you wish to disable this scan, uncheck the **Automatically detect associated runtimes and components** option in the **Options** menu.

8.1.3 Play sound on build completion

By default, Micro Focus Desktop Containers Studio plays a short sound to notify the user of application container build completion.

If you wish to disable this sound, uncheck the **Play sound on build completion** option in the **Options** menu.

8.1.4 Copy new files into the Files folder

When files are added into the application container configuration, they can optionally be added to the Files directory. This can be helpful for portability of the build files. However, this may not be desirable for software developers doing automated builds where each build should reference the original source location.

8.2 QUICK SNAPSHOT MODE

By default, Micro Focus Desktop Containers Studio uses a "quick" snapshotting algorithm that attempts to minimize the amount of time spent scanning the host system device state during snapshotting. In very rare cases, use of this mode may result in an improperly configured application container. Use of quick snapshot mode may also slightly increase the size of the application container configuration contents. It is strongly recommended that snapshotting be performed using the quick snapshot mode, as this is compatible with the vast majority of applications. Disabling quick snapshot mode significantly increases the amount of time required to complete the application container configuration process.

To disable quick snapshot mode, uncheck the **Quick snapshot mode** item from the **Options** menu.

Note: "Before" and "after" snapshots must be taken using the same snapshotting algorithm. Loading a saved snapshot image causes Micro Focus Desktop Containers Studio to automatically configure the snapshotting mode to be consistent with the algorithm used during the saved snapshot capture.

8.3 WELL-KNOWN ROOT FOLDER VARIABLES

The Micro Focus Desktop Containers engine dynamically remaps well-known root folders such as **My Documents** and **Program Files** to the appropriate location based on the host operating system at runtime. This assures, for example, that the virtualized **My Documents** folder will be mapped to **\User\USERNAME\Documents** when running on Windows Vista or **\Documents and Settings\USERNAME\My Documents** when running on Windows XP.

Most of the time, configurations are constructed using snapshotting or in the graphical user interface. However, if manually modifying the **XAPPL** file, the following well-known root folder variables may be used to configure virtual filesystem locations. Root folder variables are case sensitive.

The following is a complete list of root folder variables recognized by Micro Focus Desktop Containers Studio and the corresponding folder name displayed in the filesystem graphical user interface, followed by a brief description of the root folder.

- **@APPDATA@** (Current User Directory\AppData\Roaming)**:** The folder that serves as a common repository for application-specific data for the current roaming user.
- **@APPDATACOMMON@** (ProgramData)**:** The folder that serves as a common repository for application-specific data that is used by all users.
- **@APPDATALOCAL@** (Current User Directory\AppData\Local)**:** The folder that serves as a common repository for application-specific data that is used by the current, non-roaming user.
- **@APPDATALOCALLOW@** (Current User Directory\AppData\LocalLow)**:** The folder that serves as a common repository for low integrity application-specific data that is used by the current, non-roaming user.
- **@APPDIR@** (Application Directory)**:** Folder where the executing application container executable resides.
- **@DESKTOP@** (Current User Directory\Desktop)**:** The current user's Desktop folder.
- **@DESKTOPCOMMON@** (All Users Directory\Desktop)**:** The shared Desktop folder.
- **@DOCUMENTS@** (Current User Directory\My Documents)**:** The current user's Documents folder.
- **@DOCUMENTSCOMMON@** (All Users Directory\Documents)**:** The shared Documents folder.
- **@DOWNLOADS@** (Current Users Directory\Downloads)**:** The current user’s Downloads folder.
- **@FAVORITES@** (Current User Directory\Favorites)**:** The current user's Favorites folder.

- **@FAVORITESCOMMON@** (All Users Directory\Favorites)**:** The shared Favorites folder.
- **@MUSIC@** (Current User Directory\ Music)**:** The current user's Music folder.
- **@MUSICCOMMON@** (All Users Directory\Music): The shared Music folder.
- **@PICTURES@** (Current User Directory\Pictures): The current user's Pictures folder.
- **@PICTURESCOMMON@** (All Users Directory\Pictures): The shared Pictures folder.
- **@PROFILE@** (Current User Directory)**:** The folder that stores the current user's profile data.
- **@PROFILECOMMON@** (All Users Directory)**:** The folder that stores the shared profile data.
- **@PROGRAMFILES@** (Program Files)**:** The Program Files folder.
- **@PROGRAMFILESX86@** (Program Files (x86))**:** The Program Files folder for 32 bit applications on a 64 bit platform.
- **@PROGRAMFILESCOMMON@** (Program Files\Common Files)**:** The Program Files\Common Files folder.
- **@PROGRAMFILESCOMMONX86@** (Program Files (x86)\Common Files)**:** The Program Files\Common Files folder for 32 bit applications on a 64 bit platform.
- **@PROGRAMS@** (Current User Directory\Start Menu\Programs)**:** The folder that contains the user's program groups.
- **@PROGRAMSCOMMON@** (All Users Directory\Start Menu\Programs)**:** The folder for components that are shared across applications.
- **@STARTMENU@** (Current User Directory\Start Menu)**:** The folder containing the user's Start Menu contents.
- **@STARTMENUCOMMON@** (All Users Directory\Start Menu)**:** The folder containing the Start Menu contents for All Users.
- **@STARTUP@** (Current User Directory\Start Menu\Programs\Startup)**:** The folder containing the current user's startup items.
- **@STARTUPCOMMON@** (All Users Directory\Start Menu\Programs\Startup)**:** The folder containing startup items for All Users.
- **@SYSDRIVE@** (System Drive): The root folder of the drive containing the operating system installation.
- **@SYSTEM@** (Windows\System32)**:** The Windows System32 folder.

- **@SYSWOW64@** (Windows\SysWOW64)**:** The Windows folder that manages compatibility with 32 bit applications on a 64 bit platform.
- **@TEMPLATES@** (Current User Directory\Templates)**:** The folder that serves as a common repository for the current user's document templates.
- **@TEMPLATESCOMMON@** (All Users Directory\Templates)**:** The folder that serves as a common repository for shared document templates.
- **@VIDEOS@** (Current User Directory\Videos)**:** The folder that serves as a common repository for the current user's video files.
- **@WINDIR@** (Windows)**:** The operating system install location root.

8.4 BUILDING FROM THE COMMAND LINE

Micro Focus Desktop Containers Studio can optionally be executed from the command line. This is particularly useful for building application containers as part of an automated build process.

The command line version of Micro Focus Desktop Containers Studio is called **XStudio.exe** and can be found in the Micro Focus Desktop Containers Studio installation directory. To build an application container from the command line, execute **XStudio Configuration.xappl** from the command prompt, where **Configuration.xappl** is the name of the Micro Focus Desktop Containers Studio project created using the graphical interface.

Options specified in the configuration file may be overridden with the command line arguments given in the following table. Add these arguments to the end of the build command, e.g.:

XStudio Configuration.xappl [Arguments]

Table 8-1: Command Line Build Arguments

Command	Usage	Description
<path to XAPPL configuration file>	/l <path to license file> [/o <path to output>] [/component] [/d] [/compressed] [/uncompressed] [/deletesandbox] [/v <version>] [/startupfile <virtual path to file>]	Builds the application container based on the application configuration file. /l - Path the the license file. The license file needs to be stored in Unicode format. /o - Path to the output file. This will override the setting in the XAPPL configuration file. /component - Sets the **Project Type** to ***Component*** resulting in an SVM output rather than EXE output. /d - Enables the **Generate diagnostic-mode executable** setting. /compressed - Enables the **Compress payload** setting. /uncompressed - Disables the **Compress payload** setting. /deletesandbox - Enables the **Delete sandbox on application shutdown** setting. /v - Sets the **Version** of the output exe. /startupfile - Sets the **Startup File** of the application container.

/before	/beforepath <path to where snapshot file is saved>	Performs a before snapshot and saves the snapshot to the specified folder. /beforepath - Path to the where the snapshot file is saved.
/after	/beforepath <path to where snapshot is saved> /o <path to where XAPPL configuration file is saved>	Performs an after snapshot using the specified before snapshot path. /beforepath - Path to the before snapshot file. /o - Path to where the XAPPL configuration file is saved.
/import	/i <path to the configuration file to be imported> /o <path to where XAPPL configuration file is saved>	Import MSI, AXT, or ThinApp configurations. /i - Path to the configuration file to import. /o - Path to where the XAPPL configuration file is saved. The following features are supported when importing ThinApp configurations: • File System • Registry • Startup Files • Diagnostic Tracing • Windows Services • Output File • Sandbox Path • Child Process Exceptions • MSI Metadata • MSI Shortcuts • Environment Variables • Command-line Arguments • FileList > ExcludePattern

Note: Configuration files that are generated from the command-line using the **/after** flag do not have an output file specified in the XAPPL configuration file. When using scripting to do snapshots, it may be necessary to apply changes to the generated XAPPL file, either manually or programmatically.

Note: If running XStudio displays the error, "<SandboxCollision> is missing from the string table", it is because the XStudio application cannot be run while Micro Focus Desktop Containers Studio is running. Micro Focus Desktop Containers Studio must be closed before running XStudio via the command line.

8.5 IMPORTING CONFIGURATIONS FROM EXTERNAL TOOLS

Micro Focus Desktop Containers Studio allows configurations from certain external application virtualization tools to be automatically converted into Micro Focus Desktop Containers Studio configurations. Supported external configurations currently include MSI setup packages, ThinApp configurations, and Novell AXT snapshots.

To import a configuration from an external tool:

- Click the **Start menu** button control menu (or press **Alt-F**)
- Select **Import Configuration**. This displays the configuration import wizard.
- Click **Browse** to select the configuration to be imported.
- Click **Next**.
- Follow the step-by-step instructions in the wizard to complete the import process.

Note: Some applications which depend on specialized custom actions during the MSI installation process may require additional configuration following MSI import to be fully functional. Such applications may need to be imported using the snapshot capture method.

Note: Supported features imported from ThinApp configurations include File System, Registry, Startup Files, Diagnostic Tracing, Windows Services, Output File, Sandbox Path, Child Process Exceptions, MSI Metadata, MSI Shortcuts, Environment Variables, Command-line Arguments, and the FileList > ExcludePattern.

8.6 RUNNING NATIVE APPLICATIONS IN CONTAINER ENVIRONMENTS

Micro Focus Desktop Containers Studio allows natively installed applications to launch in application container environments. This is helpful when a natively installed application can utilize resources from an application container. For example, a user creating a plugin for Microsoft Outlook would want to enable a local version of Outlook to run in the same application container as the plugin. This is accomplished in Micro Focus Desktop Containers Studio by setting the natively installed application as the startup file (or one of the startup files).

To enable a natively installed application to launch in an application container environment:

- In the **Virtual Application** tab, click on the **Multiple** button next to the **Startup File** field
- In the **File** column, enter the local path of the natively installed application.
- (Optional) Check the **Auto Start** option to have your natively installed application automatically run when the application container is launched
- Click **OK**

This will enable your application container and natively installed application to interact with each other in the same environment.

A sample startup file path for Microsoft Word would look like this:

@PROGRAMFILES@\Microsoft Office\Office14\WINWORD.exe

If **Auto Start** has been enabled, Microsoft Word will launch with the application container, in the same virtual environment.

8.7 MODIFYING CONTAINER BEHAVIOR AT RUN-TIME

Most application container behavior is specified during design using the Micro Focus Desktop Containers Studio interface. However, in some cases, it is useful to override application container behaviors at application run-time.

Micro Focus Desktop Containers Studio allows the following settings to be specified on the application container command line. Settings specified on the command line supersede design-time settings.

Table 8-2: Run-Time Virtualization Arguments

Argument	Behavior
/XEnv=Variable Name=Value	Specifies additional environment variables. Multiple /XEnv arguments can be used to add additional environment variables.
/XLayerPath=Layer Path	Adds the given **SVM** into the virtual environment. Multiple /XLayerPath arguments can be used to add additional virtual components.
/XSandboxPath=Sandbox Path	Specifies the path to be used for the application sandbox.
/XShellEx=Command	Specifies a shell execute command to be launched from within the application container environment. This option overrides any startup files specified in the configuration. Only one /XShellEx argument can be specified.
/XShellExVerb=Command Verb	Specifies the verb to be used in conjunction with the XShellEx command. The default verb is OPEN.
/XLogPath=Log Path	Specifies the destination path for generated log files (only applies to executables built in diagnostic-mode). The path must be created prior to executing the application container. This path can include a custom file name pattern, e.g. **c:\logs\mylog*.log**
/XSpawnVmExceptions=Process Exceptions	Accepts a semi-colon delimited list of processes to be added to the child process exception list (refer to the Child processes section for more information). E.g. **/XSpawnVmExceptions=notepad.exe**
/XEnable and **/XDisable**	Enables or disables specific process configuration options. These options include: **Diagnostics, SpawnVm**, **ReadOnly**, **ExeOptimization**, **SpawnComServers**, **IsolateWindowClasses**,

	IndicateVirtualization, **DeleteSandbox**, **NotifyProcStarts**, **ReadShare**, **DRMCompat**, **ChildProcAsUser, ShutdownProcTree, DEPCompat**, and **IndicateElevated**; all of which correspond to specific options in the Process Configuration tab, e.g. **/XEnable=SpawnVm;DEPCompat**. In addition, **SuppressLogging** can be specified to enable or disable diagnostic mode.

8.8 Capture Updates to an Application Via Snapshot

Application container updates can be captured within Micro Focus Desktop Containers Studio via snapshots.

Complete the following steps to capture an update via snapshots:

- Install the native version of the application on a clean machine.
- Select **Capture Before**.
- Install necessary updates to the native application.
- Select **Capture and Diff** to create the after snapshot. This captures the deltas between the original and updated versions.
- Set the **Project Type** to **Component**, then select **Build** to create the **SVM**.

This process only captures changes between the original executable and installed updates. You can then apply the resulting **SVM** to the original application container.

8.9 SPECIFYING ADDITIONAL SVM LAYERS FOR AN APPLICATION CONTAINER

Micro Focus Desktop Containers Studio users may want to specify additional **SVM** layers for applications, in the case of updates or patches. Micro Focus Desktop Containers Studio allows three mechanisms for doing this.

The first mechanism is via the command line using the **/XLayerPath=** syntax. This syntax takes a path with optional wildcards to additional **SVM**s to load.

An example of a specified **SVM** path using a wildcard:

virtual-app.exe /XLayerPath=@APPDIR@\patches*.svm

An example of specifying **SVM**s from multiple locations:

virtual-app.exe /XLayerPath=@APPDIR@\patches*.svm
/XLayerPath=@APPDIR@\officepatches*.svm

An example of specifying**SVM**s on a network share:

virtual-app.exe /XLayerPath=\\network\share\patches*.svm

An example using Microsoft Office:

MSOffice.exe /XLayerPath=c:\Patches\MSOffice_*.svm.

This would do a wildcard match finding any files such as **MSOffice_001.svm** in the **c:\Patches** directory.

Note: The **SVM**s are applied in reverse-alphabetical priority. This means that items in **MSOffice_002.svm** have higher priority than items in **MSOffice_001.svm**.

The second mechanism is a **XAPPL** file specified way to load additional **SVM** files. It is via the <**XLayers**> portion of the **XAPPL** file and has the following elements:

Table 8-3: XAPPL XLayer Elements

Attribute	Description
XLayerSearchPattern	Attribute to provide the default search pattern, similar to what would be passed to /XLayerPath.
<RequiredXLayername="@APPDIR@\APP.svm">	Sub-elements specifying which **SVM** must be loaded, or else an error is reported.

A Sample **XAPPL** configuration is as follows:

<XLayers XLayerSearchPattern="@APPDIR@\Dependencies.svm">

<RequiredXLayer name=" Dependencies.svm" />

</XLayers>

The third mechanism is available in the Micro Focus Desktop Containers Studio interface. To access this method:

- Click on the Settings button
- Click on the Process Configuration tab
- Click on the **SVM** button

The first field is the **SVM Search** field. Here users can enter the complete path to where multiple **SVM**s are located using a wildcard. An example of using a wildcard in the search field is **@APPDIR@\patches*.svm**. This is similar to what is passed to **/XLayerPath** using the command line approach.

In the second field users can also specify required **SVM**s. In this case, the wildcard is removed and a specific file is referenced. An example of this format is **@APPDIR@\Dependencies.svm**. If the file is not found during application launch, an error will be reported.

All methods allow the use of the **@VARIABLE@** format.

Multiple **SVM**s may be specified after the **XLayerSearchPattern** attribute in a semi-colon delimited list. **SVM**s specified first in the list will take precedence over **SVM**s specified later in the list. If multiple **SVM**s are specified in one search pattern through the use of the '*****' wildcard, the **SVM**s are applied in reverse-alphabetical priority. For example, items in **MSOffice_002.svm** would have higher priority than items in **MSOffice_001.svm**.

8.10 PLATFORM MERGE

The Merge Platforms feature allows application container configurations snapshotted on multiple operating system variants (ie. Windows XP, Windows 10, etc.) to be combined into a single configuration. At runtime, the Micro Focus Desktop Containers engine dynamically applies the configuration options appropriate for the operating system variant used for execution.

Tip: The most common platform merge scenario is a merge of snapshots taken on Windows XP and Windows 7. This is because some newer applications use operating system features specific to Windows 7 and later platforms.

To merge configurations from multiple platforms:

- From the **Advanced** tab, click the **Merge Platforms** button.
- Click **Browse** and open the appropriate configuration for each applicable operating system variant.
- For operating systems without a configuration, choose which configuration it should use by using the Inherit option.
- When all configurations have been selected or set to **Inherit**, click **Browse** in the **Merge Settings** area, choose where to save the merged configuration, and click **Merge**.

To display or edit a specific operating system from a merged configuration:

- Open the merged configuration.
- From the **Advanced** tab, click on the **Display** drop down menu.
- Select the operating system that you want to display or edit.

The **Filesystem** and **Registry** panels will only display settings specific to the selected operating system. Note that you cannot edit configurations which are inherited from other platforms; to edit inherited configurations, you must select and edit the master configuration.

To change the inheritance of an operating system in a merged configuration:

- Open a merged configuration.
- From the **Advanced** tab, click the **Display** drop down menu.
- Select the operating system that you want to modify.
- Select the platform from which to inherit using the **Inherits** drop down menu.

8.11 CREATING APPLICATION STREAMING MODELS

The **Streaming** section of the **Advanced** tab allows you to profile and build streaming models for an application container.

The **Profile** feature generates transcripts, or *profiles*, which are then used to create a streaming model for the application container. Clicking the **Profile** button launches the application and creates a single transcript file based on observed user behavior during that run. It is recommended that multiple transcripts are created before creating a streaming model. Using multiple transcripts allows the streaming system to take into consideration different use cases for the application. It is also recommended that at least one transcript be created for each operating system.

Note: Only uncompressed application containers can be profiled and streamed. Compression is automatically disabled during the model build process.

To profile application containers:

- Build the application container.
- Click the **Profile** button on the **Advanced** tab.
- Select the output location for transcripts and click **OK**.
- After the application container launches, use the application for approximately one minute, as if you were a typical end-user.
- Close the application. Once the application terminates, the transcript will be created in the selected output location.
- Create additional transcripts as needed.

Once the necessary profiles have been created, the streaming model is ready to be built. The model build process uses the transcripts and **Connection Speed** parameter to compute a model of execution. After the model build process is complete, the streaming files are written to the selected output folder. The **Connection Speed** setting is used to optimize delivery of application content to the end-user.

To create a streaming model:

- Select the desired **Connection Speed**. The 1.5Mbps connection speed setting is recommended for most scenarios.
- Click the **Build Model** button.
- Select the folder where the transcripts are located and click **OK**.
- Select the folder where the streaming model will be created and click **OK**.

The resulting model is made up of an .xm file and a collection of .xs files.

8.12 LAUNCHING STREAMING MODELS USING SPOONPLAY

The SpoonPlay tool can be used to launch the streaming models created using Micro Focus Desktop Containers Studio (refer to the section "Creating application streaming models" for additional information). SpoonPlay can be used to launch streaming models which exist on your local machine, a file server, or a web server which has been configured to stream applications. In addition to launching streaming models, SpoonPlay can also be used to cache applications on your local machine.

8.12.1 Launching a streaming model

Launching a streaming model requires the use of the **/model** and **/xvm** flags. The **/model** flag specifies the location to the model files and the **/xvm** flag specifies the location of the XVM file that will be used to launch the application. The example below demonstrates the command line usage for launching a streaming model from a file server:

Spoonplay.exe /model=\\server1\apps\myapp\1-2-3-4__0\xstream\0\model.xm /xvm=\\server1\tools\xvm.exe

8.12.2 Launching an application from a web server

SpoonPlay can also launch a streaming model from a web server which has been configured to stream applications. SpoonPlay requires the **/config** flag, which specifies the location of the application configuration file that exists on the web server. The example below demonstrates the command line usage for launching a streaming model from a web server:

Spoonplay.exe /config=http://mysite.com/configs/app.xml

8.12.3 Caching an application

After launching a streaming model, SpoonPlay has a feature which allows the application to be cached as an **SVM** file on your local machine. This will eliminate the need to re-stream the model for this application during future launches. The example below demonstrates the command line usage for caching an application after launching a streaming model:

Spoonplay.exe /model=\\server1\apps\myapp\1-2-3-4__0\xstream\0\model.xm /xvm=\\server1\tools\xvm.exe /cache=c:\apps\myapp

8.12.4 Command line flag overview

The following table provides a summary of all command line arguments that can be used with SpoonPlay.

Table 8-4: SpoonPlay Arguments

Argument	Description
/model	Specifies the location of the steaming model (points specifically to the .xm file). Requires the **/xsandboxpath** argument.
/xsandboxpath	Specifies the location of the application

	sandbox. This is required when using the **/xmodel** argument.
/xvm	Specifies the location of the XVM file to be used to launch the application (XVM.exe). If this is not included, SpoonPlay will look in the same folder as it is located for a valid xvm.exe.
/config	Specifies the location (as a URL) to an application configuration file.
/cache	Specifies the location to store the cached **SVM** file after the application has launched.

8.13 Application Expiration

This section describes the **Expiration** feature. With the **Expiration** feature, application containers can be set to expire after a certain number of days or after a certain date.

To set an application container to expire after a specific number of days:

- Click on the **Expiration** button.
- Check the **Disallow execution after number of days** box.
- Select the number of days after the application is first executed on a system it will take to expire.
- Choose the **Time Source** the application container will use to validate the date.

To set an application container to expire after a certain date:

- Click on the **Expiration** button.
- Check the **Disallow execution after date** checkbox.
- Select the date the application container will expire.
- Choose the **Time Source** the application container will use to validate the date.

For all expiration modes, the **System clock** setting will use the host system's clock to validate the date. The **Web server clock** setting will validate the date against an HTTPS-based web server. Check the **Disallow execution if web server is unreachable** checkbox to prevent the application from being executed offline.

Tip: The **Web server clock** setting is more secure than the **System clock** setting since it prevents the expiration mechanism from being circumvented by modifying the system clock. However, this setting will prevent applications from executing on devices which cannot connect to the time server source.

Optionally, an **Expiration Warning** can be set to warn the user when the application container is about to expire. The message will be displayed each time the application container is executed when it is within the specified threshold.

8.14 NETWORK CONFIGURATION

This section describes the network configuration options available in Micro Focus Desktop Containers Studio.

To configure the network proxy settings to be used inside the application container:

- Click on the **Network** button.
- Click on the **Proxy** tab.
- Check **Proxy TCP connections** or **Proxy UDP connections** depending on which protocols are required.
- Set the **Proxy Server** and **Port** fields to point to the proxy server.
- Set the proxy server communication protocol to **SOCKS 5**, **SOCKS 4**, **HTTPS**, or **HTTP**.
- If the proxy server requires basic authentication, set the username and password in the **Authentication** section.

To configure DNS mappings used inside the application container:

- Click on the **Network** button.
- Click on the **DNS** tab.
- Click on the **Add…** button.
- Click in the new **Hostname or IP Address** field and enter a value. This is the hostname or IP address which will be redirected if encountered.
- Click in the new **Redirect** field and enter a value. This is the hostname or IP address which we get redirected to if the previous value was encountered. Setting a value of 0.0.0.0 here will make the previous value inaccessible.

8.15 APPLYING THE APPLICATION CONTAINER CONFIGURATION TO THE HOST DEVICE

Micro Focus Desktop Containers Studio allows the application container configuration to be applied to the host system. Applying the configuration to the host system is helpful when creating **SVM** updates for application containers.

To apply the configuration to the host system:

- Click the **Apply Configuration** button in the **Start menu** of the Micro Focus Desktop Containers Studio application.
- Enter the path of the sandbox to be merged into the current configuration.
- Click **Yes** to acknowledge that the **Apply Configuration** process cannot be undone.

Note: The **Apply Configuration** feature is not intended for use as an installation process for application containers.

For example, to create an **SVM** update to the Firefox application container template:

- Use the **Configuration Wizard** to create a Firefox application container.
- With the Firefox configuration loaded, run the **Apply Configuration** process as above.
- Open a new application container configuration.
- Capture a before snapshot.
- Open Firefox, select **Help>Check for Updates**, and apply any updates.
- Capture an after snapshot.
- Build the captured updates as an **SVM**.
- Execute the built **SVM** on top of the original virtual Firefox browser and notice that the updates have been applied.

8.16 ENABLING SHARED OBJECT ISOLATION

Micro Focus Desktop Containers Studio provides the ability to isolate shared objects in memory. Certain applications will refer to objects in memory by a specific name, which can cause runtime errors if a named object in an application container's memory collides with other objects of the same name in the memory of a natively installed version of the application on the same device. Shared object isolation creates unique names for memory objects at runtime, in order to allow an application container and a natively installed version of the same application to run side by side without conflict.

Shared object isolation can only be enabled by manually editing the **XAPPL** file for an application container. A global setting to enables/disables named object isolation. The global setting controls whether named objects are renamed by default. There are exceptions to the global setting which take regular expression values. Any shared object having a name matching an exception is given the opposite behavior of the default. The regular expressions can also be replaced by a replacement value. The first match of the regular expression in a named object will be replaced by the replacement value.

The following example, OBJECTN (of the form OBJECT1, OBJECT2, ... OBJECT99), includes named objects used by an application container that conflict with identically-named objects used by a natively installed application. Common named objects include mutexes and named pipes. The second example, Test Value.* (of the form "Test Value 1612"), will be changed to simply "Test Value", but also appended with a unique signature for the application.

Complete the following steps to enable shared object isolation for OBJECTN in an application container:

- Open the **XAPPL** file of the application container you are working with in a text editor
- Replace the <**NamedObjectIsolation ..**/> element with the example below:

```
<NamedObjectIsolation enabled="False">
<Exception regex=""OBJECT\d+" />
<Exception regex="Test Value.*" replacement="Test Value" />
</NamedObjectIsolation>
```

- Reload the **XAPPL** file in Micro Focus Desktop Containers Studio and build the application

The resulting application container will have shared object isolation enabled. Note that multiple objects in memory can be isolated simultaneously.

8.17 Specifying a Virtual Mapped Drive

A virtual mapped drive can be added to your XAPPL file in the FileSystem section. This is not available in the Micro Focus Desktop Containers Studio UI, but can be added manually in a text editor. Below is an example XML snippet for adding the virtual file *X:\AppData\Settings.txt* to the configuration. This would be inserted before the closing tag of the FileSystem section of the XAPPL file:

```
<Directory name="@DRIVE_X@" isolation="Merge">
  <Directory name="AppData" hide="False" readOnly="False" isolation="Full">
    <File name="Settings.txt" hide="False" readOnly="False"
source=".\Files\X_Drive\AppData\Settings.txt" />
  </Directory>
</Directory>
```

In this example the drive is **X**, but any drive letter can be specified in the **@DRIVE_X@** variable where the letter after the underscore represents the drive letter. The virtual mapped drive will not be visible in Windows Explorer or through File browse dialogs, but it will be visible via a command window or if the application does a file lookup using standard Windows APIs.

8.18 XAPPL FILE FORMAT

A **XAPPL** file is an XML representation of all the application container configuration settings.

All paths in the **XAPPL** file are relative to where the **XAPPL** file resides. For example, the **source** attribute of a File element will begin with "**.\Files**". The ".\" directory is the path where the **XAPPL** file should reside in order for Micro Focus Desktop Containers Studio to locate the physical source files during the build process.

The **XAPPL** file must adhere to all XML syntax rules. If there are syntax errors in the **XAPPL** file, Micro Focus Desktop Containers Studio will not load the file.

In this section, attribute values are shown in parenthesis after their description and default values are shown in **bold**.

8.18.1 XAppl Configuration Elements and Attributes

OutputLocation

- The **outputlocation** attribute is the path to the folder where the application container executable will be created. This can be a local path, a UNC path, or a mapped drive.

OutputFile

- The **outputfile** attribute is the file name of the application container executable.

ProjectType

- The **project type** attribute denotes whether this configuration is for an **EXE** file (Application) or an **SVM** file (Component).

Licensing

- The **licensing** attribute contains information about the license that was used to build the application container.

Output

- The **diagnosticMode** attribute denotes when the application output should log diagnostic information (**True**) or not (**False**). If true, the application container will create diagnostic logs in the directory where it was executed from.
- The **sourcePackage** attribute is not used.

MSI

All sub-elements contain settings pertaining to the configuration of the MSI setup file.

- The **outputMsiPath** attribute indicates the location where the setup MSI will be built.
- The **title** attribute indicates the value of the MSI title property.

- The **subject** attribute indicates the value of the MSI subject property.
- The **keywords** attribute indicates the value of the MSI keywords property.
- The **productName** attribute indicates the value of the MSI product name property.
- The **productVersion** attribute indicates the value of the MSI product version property.
- The **manufacturer** attribute indicates the value of the MSI manufacturer property.
- The **productLanguage** attribute indicates the value of the MSI product language property.
- The **author** attribute indicates the value of the MSI author property.
- The **description** attribute indicates the value of the MSI description property.
- The **manufacturerUrl** attribute indicates the value of the MSI manufacturer URL property.
- The **autoBuild** attribute denotes whether the MSI should build when the application container build completes successfully (**True**) or not (**False**).
- The **isolatePerUser** attribute denotes whether the MSI setup should be installed on a per-user basis (**True**) or installed for all users (**False**). When installing per-user, the install root path is Application Data. When installing for all users, the install root path is **Program Files**.
- The **applicationFolder**attribute indicates the subfolders into which the application container should be installed (**Company Name\Product Name**).
- The **upgradePreviousVersion** attribute denotes whether the setup should maintain the same Upgrade code when it builds (**True**) or change the Upgrade code for each build (**False**). This allows the setup to upgrade previous versions when it is installed, or to exist side by side.
- The **productCode** attribute indicates the value of MSI product code property.
- The **upgradeCode** attribute indicates the value of MSI upgrade code property.
- The **componentId** attribute indicates the value of the MSI component id property.

Packages

All sub-elements contain settings pertaining to the configuration of the packages included in the application container.

CLR

The .NET *CLR* runtime element and all sub-elements contain settings pertaining to the configuration of the virtual .NET Framework runtime.

Direct X

The DirectX element and all sub-elements contain settings pertaining to the configuration of the virtual DirectX runtime.

Java

All sub-elements contain settings pertaining to the configuration of the virtual java runtime.

RunTime

- The **name** attribute indicates the name of the java runtime (Java).
- The **platform** attribute indicates the platform that the java runtime is designed for (x86).
- The **version** attribute indicates the version of the java runtime.

Settings

- The **startupType** attribute denotes whether to use the jar file (JAR) or class path (Class) command line parameters for java.exe to launch the application.
- The **startup** attribute indicates the jar file path or class name depending on the StartupType.
- The **classpath** attribute indicates the path to the class files of the Java runtime.
- The **options** attribute denotes any additional command line parameter.

Package

- The **name** attribute indicates the name of the component or runtime.
- The **platform** attribute indicates the platforms that the component or runtime is supported on. The following are the only available values:

Any platform (Any)

x86 platform (x86)

- The **version** attribute indicates the version of the component or runtime.

Virtualization Settings

All sub-elements contain settings pertaining to the configuration of the virtual operating system.

- The **suppressBranding** attribute controls the branding pop-up that is displayed (**False**), or not displayed (**True**) in the lower right-hand corner during application startup.
- The **isolateWindowClasses** attribute is used to isolate windows classes, as registered via the **Windows ::RegisterClass** or **::RegisterClassEx** APIs. For example, this allows a virtualized Firefox instance to run while a non-virtualized instance is running.
- The **readOnlyVirtualization** attribute denotes whether the application container has the ability to modify virtual files and registry settings (**False**) or not (**True**). Setting this attribute to **True** will prevent modification to the virtual filesystem and virtual registry.
- The **disableXenocodeCommandLine** attribute controls the ability to use (**False**) any of the /X[var] command line switches to modify the runtime behavior.

- The **subsystem** attribute indicates the application output type. It can be inherited from the startup file (**Inherit**) or set explicitly to be a Windows application (**GUI**) or console application (**Console**). If **Inherit** is set, but the startup file is either not in the virtual filesystem or not an executable, then the output will be a Windows application.

- The **ie6Emulation** attribute denotes a special mode required for the Internet Explorer 6 template (**True**). For all other apps, this should be disabled (**False**).

- The **sandboxPath** attribute indicates the base path of the application sandbox **(@APPDATALOCAL@\Micro Focus\Sandbox\@TITLE@\@VERSION@).**

- The **workingDirectory** attribute defines what directory the application will run in.

- The **compressPayload** attribute controls whether the output executable will be compressed (**True**) or not (**False**).

- The **trimUACManifest** attribute removes items from the application container manifest file that may require elevation and trigger UAC prompts (**True**).

- The **enableDRMCompatibility** attribute ensures compatibility (**True**) with applications protected by software formerly known as "Armadillo" and other DRM software.

- The **deleteSandbox** attribute will cause the sandbox to be reset automatically when the application container is shutdown (**True**).

- The **shutdownProcessTree** attribute will cause the all child processes spawned within the virtual environment to be shutdown when the root process exits. By default, the root process is specified by setting the startup file.

- The **exeOptimization** attribute will attempt to launch the startup executable with the initial application container process, preventing the creation of a separate application process (**True**).

- The **enhancedDEPCompatibility** attribute provides compatibility for systems with Data Execution Protection enabled (**True**). This setting is used primarily for applications containers running on Windows 2003.

- The **notifyProcessStarts** attribute causes a notification to be sent as a debugging output string whenever a new process is started within the virtual environment (**True**).

- The **forceReadShareFiles** attribute forces any file opened by any process within the virtual environment to do so with the READ_SHARE flag set (**True**).

- The **launchChildProcsAsUser** attribute causes all child processes to be provided with the same level of privileges as the virtual machine root process (**True**).

- The **forceIndicateRunningElevated** attribute forces the application to run as if it has elevated security privileges (**True**).

- The **suppressPopups** attribute will prevent an error dialog popup if the application container encounters a fatal startup error, and will cause the application to exit silently (**True**).

ChildProcessVirtualization

- The **spawnExternalComServers** attribute controls whether the application container launches ComServers in the virtual environment (**True**) or the external environment (**False**).
- The **spawnVm** attribute denotes whether the spawned external applications are spawned inside the virtual environment (**True**) or outside the virtual environment (**False**).

ChildProcessException

- The **name** attribute indicates the name of the executable file (extension included) to except from the effects of the **spawnVm** attribute.

CustomMetadata

All sub-elements contain settings pertaining to the configuration of the individual custom metadata items.

CustomMetadataItem

- The **property** attribute indicates the name of the custom metadata item.
- The **value** attribute indicates the value of the custom metadata item.

StandardMetadata

All sub-elements contain settings pertaining to the configuration of the individual standard metadata items.

StandardMetadataItem

- The **property** attribute indicates the name of the standard metadata item. The following are the available standard metadata:

  Product Title (**Title**)

  Publisher (**Publisher**)

  Description (**Description**)

  Website (**Website**)

  Product Version (**Version**)

SplashImage

- The **path** attribute indicates the source path to the splash image displayed at application startup.
- The **transparency** attribute indicates the color in the splash image that should be made transparent when the image is displayed (E.g. **Magenta**).

StartupFiles

All sub-elements contain configuration pertaining to the individual startup files.

StartupFile

- The **node** attribute indicates the path of the startup file.
- The **tag** attribute indicates the command line trigger used to specify this entry as the startup to use.
- The **commandLine** attribute indicates the command line arguments to pass to the startup file.
- The **default** attribute denotes whether this entry is executed automatically when no tag is specified (**True**) or not (**False**).

StartupShims

A startup shim is a virtualized binary that is invoked prior to the startup file. Startup shims are used to perform customized licensing checks or other initialization tasks.

- The **path** attribute indicates the path to the virtual shim DLL implementation.
- The **param** attribute indicates a string to be passed to the shim **OnInitialize** function.
- The startup shim signature is **typedef BOOL (__stdcall *FnOnInititialize) LPCWSTR pwcsInitilizationToken)**. The return value indicates whether virtual machine execution should proceed.

ShutdownShims

A shutdown shim is a virtualized binary that is invoked prior to the application shutting down. Shutdown shims are used to perform custom actions before application shutdown.

- The **path** attribute indicates the path to the virtual shim DLL implementation.
- The **param** attribute indicates a string to be passed to the shim **OnShutdown** function.
- The startup shim signature is **typedef VOID (__stdcall *FnOnShutdown) LPCWSTR pwcsShutdownToken)**.

Layers

All sub-elements are individual virtual layers.

Layer

The **Layer** element and all sub-elements contain settings pertaining to the configuration of this layer of the virtual operating system.

- The **name** attribute indicates the name of the layer. The default layer (**Default**) is the only layer for whom the name matters. All other layer names are purely informational.

Condition

- The **variable** attribute indicates the host system setting that will be evaluated. The operating system version (**OS**) is the only available option.
- The **operator** attribute indicates the Boolean operation that will be used to evaluate the host system. The available Boolean operations are:

  greater than or equal to (**GREATEREQUAL**)

  greater than (**GREATER**)

  equal to (**EQUAL**)

  not equal to (**NOTEQUAL**)

  less than (**LESS**)

  less than or equal to (**LESSEQUAL**)
- The **value** attribute indicates the value against which the host system will be evaluated, using the Boolean operation. The available values in ascending order are:

  Windows 2000 (**Win2k**)

  Windows XP (**WinXP**)

  Windows 2003 (**Win2k3**)

  Windows Vista/2008 (**Vista**)

  Windows 7/2008r2 (**Win7**)

  Windows 8/2012 (**Win8**)

  Windows 8.1/2012r2 (**Win81**)

  Windows 10 (**Win10**)

Filesystem

All sub-elements contain settings pertaining to the configuration of the virtual filesystem.

Directory

All sub-elements contain settings pertaining to the configuration of this directory of the virtual filesystem.

- The **isolation** attribute indicates the isolation setting of the virtual folder. The available values are:

  Full isolation (**Full**)

WriteCopy isolation (**WriteCopy**)

Merge isolation (**Merge**)

- The **name** attribute indicates the name of the virtual directory.
- The **hide** attribute denotes whether the directory is marked as hidden (**True**) or visible (**False**).

File

- The **name** attribute indicates the name of the file.
- The **hide** attribute denotes whether the file is marked as hidden (**True**) or visible (**False**).
- The **source** attribute indicates the source path to the file.

Registry

All sub-elements contain settings pertaining to the configuration of the virtual registry.

Key

All sub-elements contain settings pertaining to the configuration of this key of the virtual filesystem.

- The **name** attribute indicates the name of the key.
- The **namePathInformationTuples** indicates that there is a path in the name or value of the registry item. There are 3 comma delimited integers for each path found in the name/value.

  1. Flags that indicate the state of the path (valid combinations: 0x0, 0x1, 0x2, 0x4, 0x5, 0x6)

  0x1 – All Uppercase

  0x2 – All Lowercase

  0x4 – Uses Short Path Names

  2. Start index of the path

  3. Length of the path

- The **isolation** attribute indicates the isolation setting of the virtual folder. The available values are:

  Full isolation (**Full**)

  Merge isolation (**Merge**)

Value

- The **name** attribute indicates the name of the value.
- The **type** attribute indicates the type of the value. The available values are:

  REG_SZ and REG_EXPAND_SZ (**String**)

  REG_DWORD (**DWORD**)

  REG_QWORD (**QWORD**)

  REB_BINARY (**Binary**)

  REG_MULTI_STRING (**StringArray**)

- The **namePathInformationTuples** indicates that there is a path in the name or value of the registry item. There are 3 comma delimited integers for each path found in the name/value.

  1. Flags that indicate the state of the path (valid combinations: 0x0, 0x1, 0x2, 0x4, 0x5, 0x6)

  0x1 – All Uppercase

  0x2 – All Lowercase

  0x4 – Uses Short Path Names

  2. Start index of the path

  3. Length of the path

- The **value** attribute indicates the value of the value. This is true for all types, except **StringArray**, which contains the String sub-element.

Environment Variables

- The **name** attribute indicates the name of the environment variable.
- The **value** attribute indicates the value of the environment variable.

Services

- The **name** attribute indicates the name of the windows service.
- The **autoStart** attribute denotes whether the windows service starts when the application container starts (**True**) or not (**False**).
- The **commandLine** attribute indicates the startup command line of the windows service.

- The **friendlyName** attribute indicates the friendly name of the windows service.
- The **description** attribute indicates the description of the windows service.
- The **objectName** attribute indicates the account under which the windows service ran when not virtualized.
- The **keepAlive** attribute denotes whether the windows service should continue running after the startup application has closed (**True**) or not (**False**).
- The **start** attribute indicates the value of the **Start DWORD** value in the Windows Services registry key.
- The **type** attribute indicates the value of the **Type DWORD** value in the Windows Services registry key.
- The **errorControl** attribute indicates the value of the **ErrorControl DWORD** value in the Services registry key.

Shortcuts

All sub-elements contain settings pertaining to the configuration of the MSI shortcuts.

Folder

All sub-elements contain settings pertaining to the configuration of the MSI shortcuts in this folder.

- The **name** attribute indicates the name of the folder. The two top level folders represent the Desktop (**Desktop**) and the Programs menu on the Start menu (**Programs Menu**).

Shortcut

- The **name** attribute indicates the name of the shortcut.
- The **targetPath** attribute indicates the path of the StartupFile that is the target of the shortcut.
- The **targetParameter** attribute indicates the Trigger or Tag of the StartupFile that is the target of the shortcut.
- The **arguments** attribute indicates the arguments passed to the target of the shortcut at runtime.
- The **showCmd** attribute denotes whether the application should start in a maximized(**3**), minimized(**7**) or regular(**1**) window state.
- The **description** attribute indicates the description of the shortcut.

IconResource

- The **IconResource** sub-element contains an identifier of the icon that is used for the Shortcut.

ProgIds

All sub-elements contain settings pertaining to the configuration of the ProgId.

- The **name** attribute indicates the name of the ProgId.
- The **description** attribute indicates the description of the ProgId.

IconResource

- The **IconResource** sub-element contains an identifier of the icon that is used for the file association.

Extension

All sub-elements contain settings pertaining to the configuration of the file extensions for the ProgId.

- The **extension** attribute indicates the file extension that is associated with the ProgId.
- The **mimeType** attribute indicates the MIME type of all files with the extension.

Verb

All sub-elements contain settings pertaining to the configuration of the Verb for the file extension.

- The **title** attribute indicates the title of the verb.
- The **verb** attribute indicates the verb value.
- The **arguments** attribute indicates the arguments passed to the target of the verb at runtime.
- The **default** attribute denotes whether this verb is the default verb (**True**) or not (**False**).

8.19 Example Startup Shim and Header

Below is an example of a startup shim that would be put in place to verify if an external service or application was already running before the application container could be launched.

Note: This example is a 32 bit DLL. If the application is a 64 bit application, the shim should be compiled as 64 bit.

8.19.1 Setup

Use Visual Studio 2010 (or install Visual C++ 2010 Express for free)

8.19.2 Create Project

Create a new project and select **Visual C++ > General > Empty Project** and give it a name (MFDCShim)

8.19.3 Edit project properties

- Right click on the Project and select **Properties**
- Set Configurations to **All Configurations**
- Go to **Configuration Properties > General > Project Defaults > Configuration Type** and select **Dynamic Library (.dll)**
- Go to **Configuration Properties > C/C++ > Code Generation > Runtime Library** and select **Multi-threaded (/MT)**. If you need to debug, set this to **Multi-threaded Debug (/MTd)** for the Debug configuration.
- Click OK

8.19.4 Create Header Files

- Right Click on **Header Files** and click **Add > New Item**
- Select **Header File (.h)** and give it a name (MFDCShim)

```
// MFDCShim.h

#ifdef MFDCSHIM_EXPORTS
#define MFDCSHIM_API __declspec(dllexport)
#else
#define MFDCSHIM_API __declspec(dllimport)
#endif BOOL __stdcall OnInitialize(LPCWSTR pwcsInitializationToken);
```

- Right Click on **Header Files** and click **Add > New Item**
- Select **Header File (.h)** and name it **stdafx**

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but are changed infrequently

 #pragma once
```

8.19.5 Create a Module Definition File

- Right Click on **Header Files** and click **Add > New Item**
- Select any file type, but name the file **Exports.def**
- Note that the DLL name is in the code after **LIBRARY** this should correspond to the name of the DLL (MFDCShim)

```
LIBRARY "MFDCShim"
EXPORTS
OnInitialize
```

8.19.6 Edit project properties (again)

- Right click on the Project and select **Properties**
- Set Configurations to **All Configurations**
- Go to **Configuration Properties > Linker > Input > Module Definition File** and type in **Exports.def**
- Click OK

8.19.7 Create main C++ file

- Right click **Source Files** and click **Add > New Item**
- Select **C++ File (.cpp)** and give it a name (MFDCShim)

```
// This is the main DLL file.

#include "stdafx.h"
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include "MFDCShim.h"
#include <ShellAPI.h>

/// Main dll entry point
BOOL APIENTRY DllMain( HMODULE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved)
{

    switch (ul_reason_for_call)
    {

    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }

    return TRUE;
}

/// Function called when the virtual app is being initialized

BOOL __stdcall OnInitialize(LPCWSTR pwcsInitializationToken)
{
    BOOL fLaunchApp = FALSE;

    /// perform operation to check if app should launch
    if (!fLaunchApp)

    /// let the user know that the application is unable to launch
    MessageBox(NULL, "Unable to launch the application.  Please check with
```

```
        your administrator.", NULL, NULL);


    /// return whether to launch the app
    return fLaunchApp;
}
```

8.19.8 Build and test

Now you can build the DLL and test it by selecting the DLL in Micro Focus Desktop Containers Studio under **Settings > Startup Settings > Startup shim DLL**.

8.20 WORKING WITH TURBO.NET APPLICATION CONTAINERS

Turbo.net offers tools and a large library of application containers which can be used to improve the experience of using Micro Focus Desktop Containers Studio. Micro Focus Desktop Containers Studio can be used to import, configure, and build any application container image which has been pulled from the **Turbo.net Hub**. Visit http://turbo.net for a list of available application containers and complete documentation on tooling and configuration.

8.20.1 Import Turbo.net application container images

To import an application container image from your local Turbo.net repository:

- Click on the Micro Focus Desktop Containers Studio **Start Menu** button.
- Click on the **Import Configuration** menu item.
- Select the **Local Repository** source configuration type.
- Select the application container image from the list. Use the search box if there are many options available. If the application container image that you are looking for is not present then it has not been pulled to your local repository. Use the **turbo pull** command from a Windows command prompt (see https://turbo.net/docs/reference for complete documentation on **turbo** commands).
- Select a destination folder which will receive the application container configuration file.
- After the application container image has been imported, the new configuration file will be loaded in Micro Focus Desktop Containers Studio. At this point it can be edited, saved, and built like any other application container in Micro Focus Desktop Containers Studio.

8.20.2 Configuring application containers with Turbo.net

Micro Focus Desktop Containers Studio can use the power of the Turbo.net tools to efficiently configure application containers by allowing the application container designer to make changes inside the application container directly and apply those changes directly to the configuration file.

- Click on the **Run and Merge** button in the **Build** group of the **Virtual Application** ribbon menu. You will need to login with your turbo.net account to use this feature.
- The application container will be built and then launched in the Turbo.net application container environment.
- Once running, make any configuration changes that you want (ie. set options, create/modify files, etc). When complete, close the application. If the Micro Focus Desktop Containers Studio dialog which says **VM session is running** does not close shortly afterward, press the **Force Exit** button.

- At this point you will be prompted to either accept or discard the changes that you made inside the application container. If you accept the changes, they will be committed to a new application container configuration file.

8.20.3 Publishing to the local repository

Micro Focus Desktop Containers Studio can publish any application container to the Turbo.net local repository to be used in the Turbo.net environment.

- Click the **Publish to Local Repository** button in the **Publish** group of the **Virtual Application** ribbon menu.
- You will be prompted to save and build the application container.
- Fill in the image **namespace**, **name**, and **release** information. Namespace and release are optional. These values will be used to create the image id in the form "namespace/name:release".
- You can now access the image with the id you gave it from the Turbo.net environment. See https://turbo.net/docs/ for complete documentation on the Turbo.net system.

9 TROUBLESHOOTING

This section describes the most common configuration errors that occur when using Micro Focus Desktop Containers Studio.

If you encounter a problem with an application container, please carefully read this section or query the online knowledge base before using other support options. It is very likely that the issue you have encountered is addressed in one of these places.

9.1 PROBLEMS ACCESSING INTERNET-BASED RESOURCES

Several Micro Focus Desktop Containers Studio features require access to Internet-based resources in order to function properly. These features may be unavailable if Micro Focus Desktop Containers Studio is unable to connect to the Internet.

In many corporate environments, access to the Internet is filtered through a firewall or proxy server. In these cases, Micro Focus Desktop Containers Studio will attempt to automatically configure itself for Internet access based on the system Internet settings. In some cases, however, it may be necessary to manually configure the proxy server settings.

To configure proxy server settings:

- Click the **Proxy Settings...** option on the **Options** menu. This displays the **Proxy Settings** dialog.
- Enter appropriate proxy server settings in the dialog. It may be necessary for you to consult your system administrator to obtain your organization's proxy server settings.

For users not on a corporate network, it may be necessary to ensure that Windows Firewall is configured to allow an exception for Micro Focus Desktop Containers Studio so the program can access resources.

To add an exception to Windows Firewall (Windows Vista and later):

- Open the **Control Panel**.
- Select **Windows Firewall**.
- Click the **Allow a program or feature through Windows Firewall** link.
- In the popup window, find **Micro Focus Desktop Containers Studio** and ensure the proper boxes are checked.
- Click **OK** and close the window.

9.2 GENERATING DIAGNOSTIC-MODE APPLICATION CONTAINERS

Occasionally, errors during application container configuration result in an executable which fails to run properly; that is, to exactly emulate the behavior of the original source application. Usually this is a result of an error in the configuration, such as a missing file or registry entry.

To assist in diagnosis of these problems, Micro Focus Desktop Containers Studio offers the option of enabling *diagnostic-mode*. Diagnostic-mode executables generate logging data during execution that can assist in diagnosis of problems related to virtualization.

All executables can be run in diagnostic-mode by passing in a command-line argument, */XEnable=Diagnostics*, or by using an environment variable, *__VMDIAGNOSTICS=t*. Alternatively, to generate an executable that will run in diagnostic-mode by default, select the **Generate diagnostic-mode executable** option on the **Output** section of the **Virtual Application** ribbon bar. Then click **Build** to generate the instrumented executable.

Execution of the instrumented executable will generate an **xclog_<id>.txt** file in the application startup directory that contains detailed diagnostic data gathered during execution. Inspection of this file, particularly of entries labeled **WARNING** or **ERROR**, often allows diagnosis of virtualization errors. If you require assistance from Micro Focus technical support to resolve your problem, we strongly encourage you to submit this information along with your support request to facilitate resolution of your issue.

Note: Because diagnostic-mode executables run significantly slower than standard executables and generate very large log files, diagnostic-mode executables should not be distributed to your end-users.

Thank you for using Micro Focus Desktop Containers Studio!

We hope you enjoy using Micro Focus Desktop Containers Studio. Please let us know any way we can improve your experience.

- *The Micro Focus Desktop Containers Studio Team*