MSP Client Software Guide

MSP Client Software Guide

72E-128805-03

Revision E

September 2010

Motorola, Inc
One Motorola Plaza
Holtsville, New York 11742-1300
http:/motorola.com/enterprisemobility

Table of Contents

About This Guide

The MSP Client Software Guide document provides guidance for pre-installation planning, application set up, and the information for successfully using Mobility Services Platform software product version 3.3.1. Newer versions of this document may be available at http://support.symbol.com/support/product/softwaredownloads.do.

Related Documents

- MSP 3.3.1 Release Notes, p/n 72E-100160
- Understanding Mobility Services Platform 3.3, p/n 72E-128712
- Using Mobility Services Platform 3.3, p/n 72E-128802
- MSP 3.3.1 Software Installation Guide, p/n 72E-100159
- Administering MSP 3.3.1, p/n 72E-128775
- Using the Athena Remote Control Solution, p/n 72E-127346
- Using the Motorola Remote Control Solution, p/n 72E-127347
- Using the Athena Data Collection Solution, p/n 72E-139417
- Using the Motorola Data Collection Solution, p/n 72E-139416
- AirBEAM Package Builder Product Reference Guide, p/n 72E-55769
- AirBEAM Package Builder Version 2.X Addendum

Help text is available from the MSP Console UI via hovering the mouse pointer over certain input fields of the screen and by clicking Help in the upper-right corner of each screen (see below

Service Information

If you have a problem with your software, contact Motorola Enterprise Mobility support for your region. Contact information is available at: http://www.symbol.com/contactsupport.

When contacting Enterprise Mobility support, please have the following information available:

- Serial number of the software
- Model number or product name
- Software type and version number
- Software license information

Motorola responds to calls by email, telephone or fax within the time limits set forth in support agreements.

If you purchased your Enterprise Mobility business product from a Motorola business partner, contact that business partner for support.

Chapter 1 - Introduction

Overview

MSP Client Software is a set of software components that execute on the device and collectively provide the device-side logic required to support MSP functionality. A device cannot be Directly Managed by MSP unless the MSP Client Software is installed on the device and has been properly configured for use with MSP.

Beginning with MSP 3.3.1, there are multiple varieties of MSP Client Software to support a variety of new types of supported devices. There may also be multiple versions of some varieties of MSP Client Software to handle variations amongst devices in the same Device Class and/or to offer different levels of functionality. The varieties and versions of MSP Client Software are introduced in sections of this chapter and described in further detail in subsequent chapters of this document.

MSP Client Software for Windows Devices

What is a Windows Device?

A Windows Device is any device that is supported by MSP and that is running a supported version of Windows CE or Windows Mobile Operating System.

The Windows Device Client

All Windows Devices will be Directly Managed by MSP via a variety of the MSP Client Software called the Windows Device Client. The Windows Device Client is described in more detail in Chapter 2 – The Windows Device Client.

Notes:

Prior to MSP 3.3.1, **only** Windows Devices were supported. Beginning with MSP 3.3.1, additional types of devices, other than Windows Devices, are also supported.

The Device Class “Mobile Device”

In MSP 3.3.1, all Windows CE or Windows Mobile Operating System devices are reported to MSP as having a Device Class of “Mobile Device”. The relationship between Windows Devices, the Windows Device Client, and the Device Class “Mobile Device” is shown in Figure 1 below.

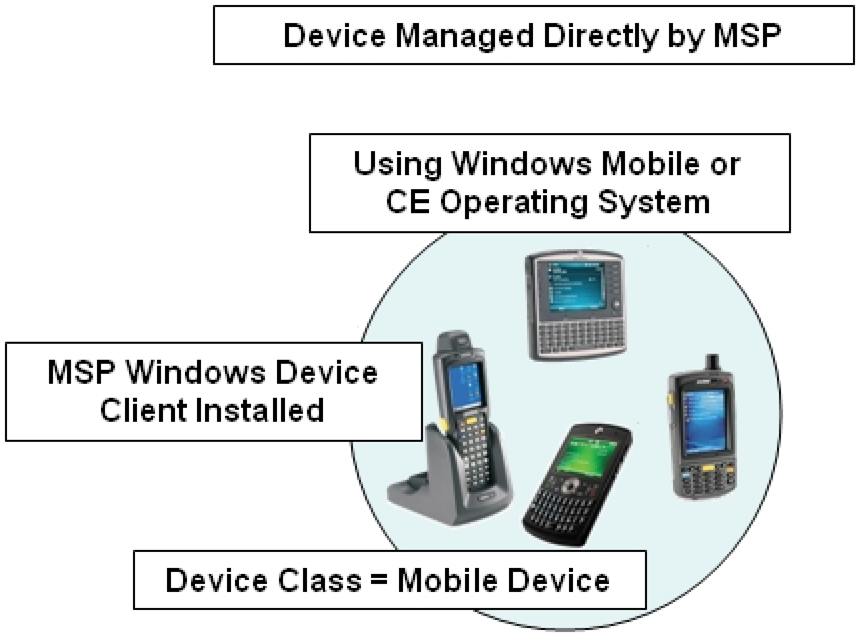

Device Managed Directly by MSP
Using Windows Mobile or CE Operating System
MSP Windows Device Client Installed
Device Class = Mobile Device

Figure 1 – Mobile Devices

Note:

Prior to MSP 3.3.1, all devices supported by MSP were Windows Devices and had the implicit Device Class “Mobile Device”. In MSP 3.3.1, the majority of devices supported by MSP are still Windows Devices and this term will be generically used to refer to them, since they are **devices** that run Windows. To avoid backward compatibility issues, however, the Device Class “Mobile Device” will continue to be used in MSP 3.3.1 for these devices.

Future versions of MSP will likely support additional types of devices that may be “mobile” but are not Windows Devices. It might seem reasonable, therefore, for such devices to be have the Device Class “Mobile Device”. The likelihood is high, however, that devices that are not running Windows CE or Windows Mobile would require different software and therefore would need to be differentiated from Windows Devices. Such devices shall therefore not use the Device Class “Mobile Device”, but instead shall use new unique Device Classes.

Important:

No devices other than Windows Devices are allowed to be reported to MSP as having a Device Class of “Mobile Device”.

All Windows Devices will generally be fairly similar to each other, in terms of how they are managed and the content that can/should be deployed to them. When necessary, distinctions amongst Windows Devices may be made based on Operating System version, device model, etc. Conversely, Windows Devices will generally be very different from other types of devices, such as Windows PCs. The mechanism used in MSP 3.3.1 to differentiate devices that are fundamentally different from each other is the Device Class.

Important:

Beginning with MSP 3.3.1, it is important to consider Device Class when managing devices, if devices of more than one Device Class are being managed.

MSP Client Software for Windows PCs

What is a Windows PC?

A Windows PC is defined to be any “Wintel X86” PC hardware that is supported by MSP that is running a supported version of a Microsoft Windows XP Operating System. This may include Desktops, Laptops, Tablets, etc.

The Windows PC Client

All Windows PCs will be Directly Managed by MSP via a variety of MSP Client Software called the Windows PC Client. The Windows PC Client is described in detail in Chapter 3 – The Windows PC Client.

The Device Class “Windows PC”

All Windows PCs shall be reported to MSP as having a Device Class of “Windows PC”. ”. The relationship between Windows PCs, the Windows PC Client, and the Device Class “Windows PC” is shown in Figure 2 below.

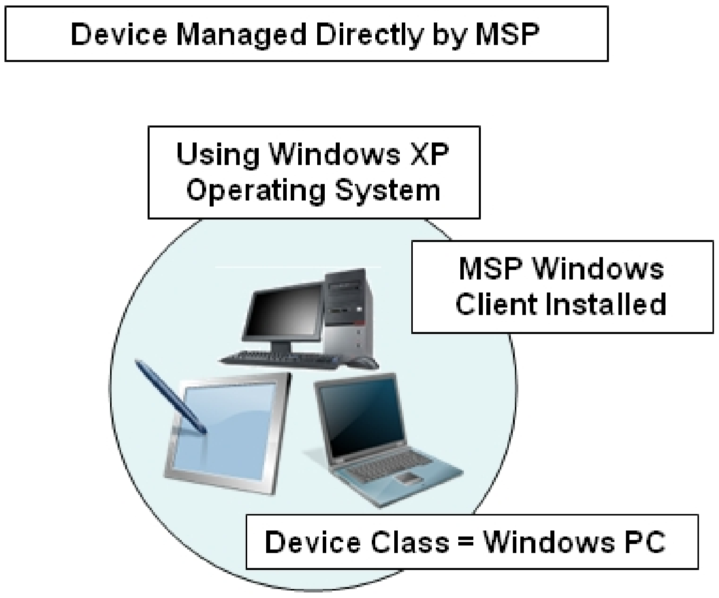

Device Managed Directly by MSP
Using Windows XP Operating System
MSP Windows Client Installed
Device Class = Windows PC

Figure 2 – Windows PCs

Notes:

The intent of the term “Windows PC” is to clearly differentiate Windows PCs, which run the Windows XP Operating System, from Windows Devices, which run the Windows CE or Windows Mobile Operating Systems.

A “mobile computer”, such as a Tablet PC running the Windows XP Operating System, will have a Device Class of “Windows PC” **not** a Device Class of “Mobile Device”. This is because the management of the device is much more dependent on its Operating System than on its physical form or appearance.

All Windows PCs will generally be very similar to each other, in terms of how they are managed and the content that can/should be sent to them. In MSP 3.3.1, no means is provided to differentiate Windows PCs from each other based on Operating System or model, but this capability might be added in the future if the need arises. Conversely, Windows PCs will generally be very different from other devices, such as Windows Devices. The mechanism used in MSP 3.3.1 to differentiate devices that are fundamentally different from each other is the Device Class.

Important:

Beginning with MSP 3.3.1, it is important to consider Device Class when managing devices, if devices of more than one Device Class are being managed.

Chapter 2 - Windows Device Client

Windows Device Client Compatibility

Windows CE

The Windows Device Client is designed to support Motorola (and Symbol) devices that are running the Microsoft Windows CE Operation System of version 4.2 or higher.

Important:

Due to the extreme variability that can exist in OEM builds of the Windows CE Operating System, the Windows Device Client is not intended to support non-Motorola (or non-Symbol) Windows CE-based devices.

Windows Mobile

The Windows Device Client is designed to support (Motorola and non-Motorola) devices that are running various versions of the Microsoft Windows Mobile Operating System (a variant of the Microsoft Windows CE Operating System). In particular, the Windows Device Client is designed to support Windows Mobile 2003 and Windows Mobile 5.0 and higher (e.g. including 6.0, 6.1, 6.5, etc.).

Important:

Due to the very significant differences between all versions of the Microsoft Windows Mobile Operating System and the Windows Phone 7 Operating System, the Windows Device Client is not intended to support devices that are running the Windows Phone 7 Operating System.

Windows Device Client Software Components

The Windows Device Client is comprised of the software components described in the following subsections:

Windows Device Rapid Deployment Client (RD Client)

The RD Client provides support for MSP Staging functionality, provides support for the MSP Legacy Staging process, and provides support for backward-compatible legacy MSP 2.x Legacy Staging functionality.

Windows Device Client RD Client Executable(s)

The Windows Device Client RD Client consists primarily of the executable file “rdclient.exe”. Depending on the characteristics of a specific Windows Device, the RD Client may be installed to various locations in the device. No matter where it may be installed to (e.g. for Persistence), the RD Client is typically copied to and executed from the \Windows folder on the device.

Windows Device Client RD Client User Interface

The installation of the Windows Device Client RD Client generally includes the placement of a shortcut into the start menu of the device. This shortcut is typically named “Rapid Deployment Client” to make it easy for the Device User to identify it when the need to initiate Staging operations arises.

The user interface of the Windows Device Client RD Client is designed to be similar on all devices and to adapt to the limitations and capabilities of those devices. In particular, since the size of the display screen varies between devices, the user interface of the RD Client is designed to be capable of being used effectively on a very small screen while also providing some enhanced usability on larger screens. As a general rule, the best user experience will be provided on devices with a quarter VGA (240x320 pixels) display. Devices with smaller screens will not display some optional information and devices with larger screens will see no additional benefit.

The primary user interface of the Windows Device Client RD Client is a menu structure consisting of a main menu and some sub-menus. Each menu consists of a title describing the menu and a list of menu options. Depending on the display screen size of a device, the number of menu items shown in the list may vary, but the list will scroll if all menu items cannot be shown on the screen at once.

Figure 3 below shows the main menu of the RD Client as rendered on two devices with different display screen sizes.

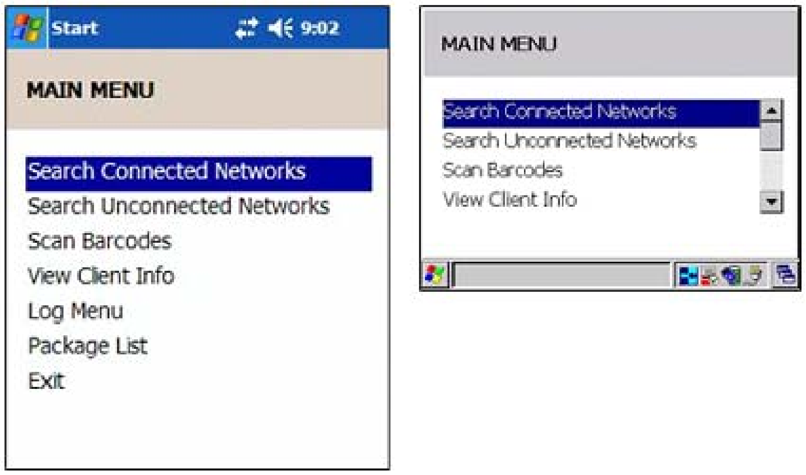


Figure 3 – RD Client UI

The various individual menu options are described below:

- Search Connected Networks

  This option causes the RD Client to search connected adapters by the following process:

  Rapid Deployment chooses to use TCP or UDP IP protocol based on the operational status of the interface. The actual interface used by Rapid Deployment will be the first available interface.

  It is possible that Rapid Deployment may select an interface that cannot communicate to an OnDemand Server because the interface routing actually occurs at a lower level (WinSock) than Rapid Deployment.

  For example, suppose a device has an established connection via Active Sync and WLAN. The WLAN connection is not connected to the same network as the OnDemand Server. Active Sync is connected to the same network as the OnDemand Server. However, the lower level may choose the WLAN interface for Rapid Deployment when the "Search Connected Networks" is selected. In another example, the lower level could also choose an Active Sync connection instead of WLAN where Active Sync is not connected to the same network as the OnDemand Server , but the WLAN is connected.

  It is highly recommended to disconnect any interfaces that are not connected to the same network as the OnDemand server to resolve situations like the examples explained above.

- Search Unconnected Networks

  This option causes the RD Client to try to establish the “well-known connections” on the WLAN adapter to see if it can locate an On-Demand Profile Server via any of them. Once the WLAN is configured, Rapid Deployment chooses to use TCP or UDP IP protocol based on the operational status of the interface . The actual interface used by Rapid Deployment will be the first available interface.

- Scan Barcodes

This option causes the RD Client to begin waiting for the Device User to scan Barcodes for a Staging Profile. Since this is the most commonly used function, the RD Client is designed to automatically select this option when the RD Client is launched. This allows the Device User to launch the RD Client and immediately begin scanning Barcodes. To exit this option and go to the main menu, use the Options button or soft key.

- View Client Info

  This option causes the RD Client to display information that identifies the version of the Windows Device Client that is running on the device, the model and OS of the device, the plug-in that will used to configure WLAN settings, and the UUID of the device.

- Log Menu

  This option causes the RD Client to display a sub-menu presenting options to configure and view the log files being created by the RD Client. The following options are provided by the sub-menu:

  - View Log

    This option causes the contents of the log to be displayed.

  - View Job Log

    This option causes the contents of the job log to be displayed.

  - Set Log Level

    This option allows the amount of logging produced into the log to be adjusted.

  - Set Job Log Level

    This option allows the amount of logging produced into the job log to be adjusted.

- Package List

  This option causes the RD Client to display a sub-menu presenting the list of Packages present on the device. Selecting any Package listed in the menu will display the name and version of the Package and will allow the Package to be deleted.

  The Package delete option can be disabled if it is desired to prevent Device Users from deleting Packages from within the RD Client user interface. This can be accomplished by launching the RD Client with a command line of –OD as described in the following section.

- Exit

  This option causes the RD Client to terminate.

Windows Device Client RD Client Programmatic Interface

There are times when it is desirable for an application or other process on a device to invoke the RD Client programmatically to perform some function. This is done by launching the executable file “\windows\rdclient.exe”, typically using the CreateProcess() API call, or equivalent.

When invoking the RD Client programmatically, various command line arguments can be provided to achieve different results, as described in Table 1.

Table 1 – Command Line Arguments and Results

Argument	Description	Example
	Start Staging by allowing the user to scan Barcodes. This is the default behavior when no command line argument is supplied and is used by the RD Client shortcut that is automatically placed onto the start menu when the Windows Device Client is installed.	\windows\rdclient.exe
`-A`	Start electronic Staging for Connected Networks	\windows\rdclient.exe -A
`-C<filename>`	Process condition .BLOB file of the specified name. Used primarily when debugging or testing condition plug-ins.	\windows\rdclient.exe –Cmy.condition.blob
`-I<filename>`	Process settings .BLOB file of the specified name. Used primarily when debugging or testing settings plug-ins that are created and separately downloaded to the device.	\windows\rdclient.exe –Imy.setting.blob
`-OB`	Launch RD Client and disable access to the Scan Barcodes option by the Device User. This can be used when launching the RD Client to control what the Device User is allowed to do. Can be combined with other –O options or with –A option.	\windows\rdclient.exe –OB

Argument	Description	Example
`-OC`	Launch RD Client and disable access to the Search Connected Networks option by the Device User. This can be used when launching the RD Client to control what the Device User is allowed to do. Can be combined with other –O options.	\windows\rdclient.exe –OC
`-OD`	Launch RD Client and disable access to the Delete Package option in the by the Device User. This can be used when launching the RD Client to control what the Device User is allowed to do. Can be combined with other –O options.	\windows\rdclient.exe –OD
`-OU`	Launch RD Client and disable access to the Search Unconnected Networks option by the Device User. This can be used when launching the RD Client to control what the Device User is allowed to do. Can be combined with other –O options.	\windows\rdclient.exe –OU
`-P<Profilename>`	Launch the RD Client and attempt to perform Automatic Staging to pull the Staging Profile of the specified name from the Relay Server. In order for this to work, the prerequisites for Automatic Staging must be met as discussed in Understanding MSP – Understanding Staging.	

Windows Device Client RD Client Registry Interface

Since both the RD Client and the MSP Agent for Windows Devices perform deployment of Bundles and Packages, they share common code. As a result, they also share many of the same Registry Entries. For example, Relay Server Settings applied to a device when using the RD Client are stored identically to Relay Server Settings applied to a device when using a Provisioning Job.

The RD Client does have some unique configuration of its own as well. The base Registry path where the RD Client stores its unique configuration is:

```
HKEY_LOCAL_MACHINE\SOFTWARE\MSP\RD
```

The unique Registry Entries supported by the RD Client are described in the following sub-section:

Additional Device Identifiers for On-Demand Staging

When performing On-Demand (Electronic) Staging, the RD Client sends to the On-Demand Profile Server one or more Device Identifier values. By default, the RD Client sends only the device unique identifier (the same value reported by the device as the Device Attribute “identity.uuid”).

When using Dynamic On-Demand Staging, it is sometimes desirable to send additional Device Identifiers to allow Dynamic Staging to map Dynamic Staging Profiles to devices based on alternate values. For example, on a Motorola EWP/TEAM device, it may be desirable to use the MAC Address of the device as the “key” to use for Dynamic Staging since that value is used by the rest of the EWP/TEAM system to indentify devices.

To accomplish this, one or more Registry values can be created under the Registry subkey “OnDemandAttributes”. Each value found under that subkey will cause the RD Client to send an additional Device Identifier to the On-Demand Profile Server. These additional Device Identifiers augment but do not replace the default Device Identifier.

The On-Demand Profile Server will match the entire set of Device Identifiers sent by the RD Client against the Device Identifier specified in the Staging Batch that is currently being served-up. If no match is found, then Dynamic Staging cannot occur for that device and the RD Client will display an error. If a match is found, then the value supplied for the device for that Device Identifier will be compared against the values supplied for that Device Identifier in the Staging Batch. If no match is found, then Dynamic Staging cannot occur for that device and the RD Client will display an error.

Notes:

Adding additional Device Identifiers is not intended for direct use by customers. Since the only way to configure additional Device Identifiers is via the Device Registry, this feature is primarily intended to allow certain devices to be shipped with additional Device Identifiers built-in. For example, the Motorola EWP/TEAM device comes standard with the MAC Address set as an additional Device Identifier to provide a more consistent overall experience when using MSP and the EWP/TEAM system together.

An end-customer could customize a device by adding the required Registry Entries. But since those Registry Entries were not present when the device shipped, they could not be used fresh-out-of-the-box On-Demand Dynamic Staging. It is not simply a matter of adding a .REG file to the device, since each Registry Value must specify both the name of the Device Identifier and the value of that Device Identifier for that device. This generally means that code must be executed on the device to acquire the appropriate value and store it in the Registry Value. Only then can the additional Device Identifier be successfully used to select devices during Dynamic On-Demand Staging.

Windows Device AirBEAM Smart Client

The AirBEAM Smart Client provides backward-compatible legacy AirBEAM functionality and backward-compatible legacy MSP 2.x Level 2 Agent functionality.

Note:

The AirBEAM Smart Client is not supported on all devices. Because end of life has been announced for AirBEAM Smart, support for the AirBEAM Smart Client may soon be discontinued on newer devices.

Windows Device AirBEAM Smart Client Executable(s)

The AirBEAM Smart Client consists primarily of the executable file “abclient.exe”. Depending on the characteristics of a specific Windows Device, the AirBEAM Smart Client may be installed to various locations in the device. No matter where it may be installed to (e.g. for Persistence), the AirBEAM Smart Client is typically copied to and executed from the \Windows folder on the device.

Windows Device AirBEAM Smart Client User Interface

The installation of the AirBEAM Smart Client generally includes the placement of a shortcut into the start menu of the device. This shortcut is typically named “AirBEAM Client” to make it easy for the Device User to identify when the need to initiate Legacy AirBEAM operations arises.

Since the AirBEAM Smart Client is provided for legacy backward-compatibility purposes only, its user interface will not be detailed in this document. Information about the AirBEAM Smart Client user interface can be found in the document titled “AirBEAM Smart Windows CE Client Product Reference Guide (P/N 72-63060-01 Rev. B)”, which is available for download from the AirBEAM Smart sub-section of the Mobility Software section at http://support.symbol.com/support/product/softwaredownloads.do.

Windows Device AirBEAM Smart Client Programmatic Interface

Since the AirBEAM Smart Client is provided for legacy backward-compatibility purposes only, its programmatic interface will not be detailed in this document. Information about the programmatic interface of the AirBEAM Smart Client can be found in the document titled “AirBEAM Smart Windows CE Client Product Reference Guide (P/N 72-63060-01 Rev. B)”, which is available for download from the AirBEAM Smart sub-section of the Mobility Software section at:

http://support.symbol.com/support/product/softwaredownloads.do.

Windows Device AirBEAM Smart Client Registry Interface

Since the AirBEAM Smart Client is provided for legacy backward-compatibility purposes only, its Registry interface will not be detailed in this document. Information about the Registry interface of the AirBEAM Smart Client can be found in the document titled “AirBEAM Smart Windows CE Client Product Reference Guide (P/N 72-63060-01 Rev. B)”, which is available for download from the AirBEAM Smart sub-section of the Mobility Software section at http://support.symbol.com/support/product/softwaredownloads.do.

Windows Device Client MSP Agent

The Windows Device Client MSP Agent provides agent functionality for Windows Devices that are Directly Managed by MSP. This includes Provisioning functionality, when used with MSP Provision Edition or MSP Control Edition, and Data Collection functionality, when used with MSP Control Edition.

Windows Device Client MSP Agent Executable(s)

The Windows Device Client MSP Agent consists primarily of the executable file “30agent.exe”. Depending on the characteristics of a specific Windows Device, the MSP Agent may be installed to various locations in the device. No matter where it may be installed to (e.g. for Persistence), the AirBEAM Smart Client is typically copied to and executed from the \Windows folder on the device.

Windows Device Client MSP Agent User Interface

The installation of the Windows Device Client MSP Agent generally includes the placement of a shortcut into the start menu of the device. This shortcut is typically named “MSP Agent” to make it easy for the device User to identify it when the need to interact with the user interface (UI) of the MSP Agent arises.

It should be noted that the MSP Agent UI is generally not intended to be used by most Device Users. Instead, the MSP Agent UI is generally considered to be intended for monitoring and troubleshooting purposes. As such, it is most appropriate for use by operations managers or supervisors with advanced knowledge, skills, and training.

It is possible to configure the MSP Agent so its UI is disabled. This can be accomplished by configuring the MSP Agent using an Agent.30 Settings Object. Select “Configure MSP 3.2 Features” and set “Disable UI” to True. This may be useful to prevent Device Users from launching the MSP Agent UI which could distract from their primary tasks and/or permit them to perform undesirable actions (such as exiting the MSP Agent).

The MSP Agent UI is designed to be similar on all devices and to adapt to the limitations and capabilities of those devices. In particular, since the size of the display screen varies between devices, MSP Agent UI is designed to be capable of being used effectively on a very small screen while also providing some enhanced usability on larger screens. As a general rule, the best user experience will be provided on devices with a quarter VGA (240x320 pixels) display. Devices will smaller screens may not display some optional information and devices with larger screens may see no additional benefit from the extra screen size.

The primary MSP Agent UI consists of a menu structure with of a main menu and a number of sub-menus. Each menu consists of a title describing the menu and a list of menu options. Depending on the display screen size of a device, the number of menu items shown in the list may vary, but the list will scroll if all menu items cannot be shown on the screen at once.

The various individual menu options are described below:

- Monitor Processing

  This option causes the MSP Agent to display a window in which the progress of ongoing management operations can be viewed (monitored) by the Device User. This feature is useful when testing or troubleshooting the management of a device.

- Force Check-In

  This option causes the MSP Agent to execute immediately, if execution is not already in progress. It then switches to the Monitoring Processing screen, as described above, to show the progress of execution.

  Notes:

  This option may cause execution to occur before the next scheduled execution time, but it does **not** change the how the MSP Agent behaves when it executes.

  Despite the name "Force Check-In", this option may not actually cause a Check-In with the Relay Server to occur. If the Relay Server cannot be contacted, due to connectivity issues or because of limits imposed by Check-In Conditions, then no communications with the Relay Server will occur.

  Nonetheless, other processing, such as the execution of Package Check-In Commands, will be still processed as on any other execution of the MSP Agent.

- Package List

  This option causes the MSP Agent to display a sub-menu presenting the list of Packages present on the device. Selecting any Package listed in the menu will display the name and version of the Package and will allow the Package to be deleted.

  The Package delete option can be disabled if it is desired to prevent Device Users from deleting Packages from within the MSP Agent UI. This can be accomplished by configuring the MSP Agent using a Settings Object. Select "Configure MSP Features" and set "Disable delete Package" to True.

- View Client Info

  This option causes the MSP Agent to display information that identifies the version of the Windows Device Client that is running on the device, the model and OS of the device, the plug-in that will used to configure WLAN settings, and the UUID of the device.

- Log Menu

  This option causes the MSP Agent to display a sub-menu presenting options to configure and view the log files being created by the MSP Agent. The following options are provided by the sub-menu:

  - View Log

    This option causes the contents of the log to be displayed.

- View Job Log

  This option causes the contents of the job log to be displayed.

- Set Log Level

  This option allows the amount of logging produced into the log to be adjusted.

- Set Job Log Level

  This option allows the amount of logging produced into the job log to be adjusted.

- Hide UI

  This option causes the MSP Agent UI to be hidden. The MSP Agent continues to perform scheduled execution according to its current configuration, but no progress or feedback of its operation is presented to the Device User. This is generally the normal mode of production operation since the Device User should generally be unaware of and not impacted by the operation of the MSP Agent.

- Exit

  This option causes the MSP Agent to terminate its operation and stop all scheduled execution until it is launched again. While this option has value in some testing and troubleshooting scenarios, it is likely not something that would be desirable for a Device User to do during production operation.

  It is possible to allow the Device User to access the MSP Agent user interface and yet prevent the Device User from this option. This can be accomplished by configuring the MSP Agent using a Settings Object. Select “Configure MSP Features” and set “Disable exit” to True.

  Note:

  While this option causes the MSP Agent to exit, it does not prevent the MSP Agent from starting again when the device is rebooted.

Windows Device Client MSP Agent Programmatic Interface

There are times when it is desirable for an application or other process on a device to invoke the MSP Agent programmatically to perform some function. The MSP Agent has three primary Programmatic Interfaces: Command Line Execution, Package Commands, and Batch Files.

Command Line Execution

Command Line Execution is done by launching the executable file "\windows\30agent.exe", typically using the CreateProcess() API call, or equivalent. When invoking the MSP Agent programmatically, various command line arguments can be provided to achieve different results, as described in Table 2 below.

Table 2 – Windows Device Client MSP Agent Command Line Arguments

Argument	Description	Example
	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent with the MSP Agent UI hidden. The instance will continue executing in the background. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will detect that no command line arguments were present and will terminate without affecting the first instance of the MSP Agent, which will continue executing. This is the behavior exhibited when the MSP Agent is launched with no command line arguments. This is **not** the behavior when the MSP Agent is launched using the MSP Agent shortcut on the start menu (see the explanation below of the **–U** command line argument).	\windows\30agent.exe
-C	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent with the MSP Agent UI hidden. This instance will process Check-In Commands for all installed Packages. The instance will then terminate, leaving no instance of the MSP Agent executing. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to process Check-In Commands for all installed Packages. The second instance will terminate as soon as the request has been issued. The first instance may not service the request until immediately. Once the first instance has serviced the request, it will continue executing. This command line argument is used by the standard Package "StartSmartStaging" to process Check-In Commands of Packages that were deployed during Staging. This command line argument can be used on the same command line as the **–J** and/or **–R1** command line arguments to request multiple activities.	\windows\30agent.exe -C

Argument	Description	Example
-J	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing with the MSP Agent UI hidden. This instance will contact the Relay Server, if allowed, and process Jobs, as appropriate. The instance will then terminate, leaving no instance of the MSP Agent executing. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to contact the Relay Server, if allowed, and process Jobs, as appropriate. The second instance will terminate as soon as the request has been issued. The first instance may not service the request immediately. Once the first instance has serviced the request, it will continue executing. This command line argument can be used on the same command line as the **–C** and/or **–R1** command line arguments to request multiple activities.	\windows\30agent.exe -J
-M	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing with the MSP Agent UI showing the **Monitor Processing** screen. This instance will continue running in the foreground. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to show the **Main Menu** screen of the MSP Agent UI. The second instance will terminate as soon as the request has been issued. The first instance will continue running in the foreground.	\windows\30agent.exe -M
-R1	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing with the MSP Agent UI hidden. This instance will send a Discovery Document to the Relay Server, if allowed. The instance will then terminate, leaving no instance of the MSP Agent executing. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to send a Discovery Document to the Relay Server, if allowed. The second instance will terminate as soon as the request has been issued. The first instance may not service the request immediately. Once the first instance has serviced the request, it will continue executing. This command line argument can be used on the same command line as the **–C** and/or **–J** command line arguments to request multiple activities.	\windows\30agent.exe –R1

Argument	Description	Example
-T	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing with the MSP Agent UI hidden. This instance will continue running in the background. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to reload its configuration and perform an execution cycle. The second instance will terminate as soon as the request has been issued. The first instance may not service the request immediately. Once the first instance has serviced the request, it will continue executing. This is used by Packages that update the configuration of the MSP Agent and want the changes to take effect immediately without requiring a rebooting the device.	\windows\30agent.exe –T
-U	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing with the MSP Agent UI showing the **Main Menu** screen. This instance will continue running in the foreground. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to show the **Main Menu** screen of the MSP Agent UI. The second instance will terminate as soon as the request has been issued. The first instance will continue running in the foreground. This command line argument is used by the MSP Agent shortcut on the start menu.	\windows\30agent.exe –U
-X	If an instance of the MSP Agent is not already executing, this will start an instance of the MSP Agent executing. This instance will then immediately terminate, leaving no instance of the MSP Agent executing. If an instance of the MSP Agent is already executing, this will start a second instance of the MSP Agent. The second instance will send a request to the first instance to terminate. The second instance will terminate as soon as the request has been issued. The first instance may not service the request immediately. Once the first instance has serviced the request, it will terminate, leaving no instance of the MSP Agent executing.	\windows\30agent.exe –X

Package Commands

Package Command Types

Install Command

When a Package is installed via the MSP Agent, the Package can optionally contain an Install Command. When all of the files contained within the Package have been successfully written to their specified destination locations, the Install Command of the Package, if one exists, is executed.

The Install Command of a Package is most commonly used to “complete” the installation of the Package by running some program, often one delivered by the Package. For example, the Install Command for a Package that delivers an application would typically launch that application.

Uninstall Command

When a Package is uninstalled via the MSP Agent, the Package can optionally contain an Uninstall Command. Before any files by the Package are removed, the Uninstall Command of the Package, if one exists, is executed.

The Uninstall Command of a Package is most commonly used to “prepare” for the uninstallation of the Package by running some program, often one organically delivered by the Package. For example, the Uninstall Command for a Package that delivered an application would typically stop the application from running so it can be successfully removed.

Check-In Command

Once a Package has been installed via the MSP Agent, the Package can optionally contain a Check-In Command. Each time the MSP Agent executes, the Uninstall Command of the Package, if one exists, is executed.

The Check-In Command of a Package is most commonly used to “periodically” execute some program, often one delivered by the Package. For example, the Uninstall Command for a Package that delivered a Control Module would typically execute a utility delivered by the Package to perform the function of that Control Module.

Package Command Formats

The possible formats for Package Commands are shown in Table 3 below:

Table 3 – Package Command Formats

Format Title	Format Description	Format Example(s)
`Launch Executable`	Launch an executable file with or without command line parameters. Wait for the launched executable to complete, check the return code, and treat a return code of 0 as success and non-zero as failure.	\windows\solitare.exe \windows\rdclient.exe -Pmyprofile
`Launch Non-Executable`	Launch a non-executable file by launching the executable file to which the non-executable file is associated based on file extension. Wait for the launched executable to complete, check the return code, and treat a return code of 0 as success and non-zero as failure.	\temp\mysong.wma \temp\mypage.html \temp\myvideo.wmv
`Launch without waiting`	Launch an executable file or non-executable file. Do not wait for the launched executable to complete and hence proceed on the assumption that it was successful.	NOWAIT \windows\solitare.exe NOWAIT \windows\rdclient.exe –Pmyprofile NOWAIT \temp\myvideo.wmv
`Launch and ignore return code`	Launch an executable file or non-executable file as described above. Wait for the launched executable to complete, but do not check the return code, and proceed on the assumption that it was successful.	WAITIGNORE \application\myapp.exe
`Launch and invert return code`	Launch an executable file or non-executable file as described above. Wait for the launched executable to complete, check the return code, and treat a return code of 0 as failure and non-zero as success.	INVERTRESULT \application\myapp.exe

Format Title	Format Description	Format Example(s)
`Call a function in a DLL`	Call a function exported by a DLL. The function must take a fixed number (from 0 to 5) of Unicode (i.e. LPTSTR) parameters and must return a DWORD result. The function will be passed the values of quoted string parameters specified on the command line. When the function returns, a return code of 0 will be treated as success and non-zero as failure. The modifiers described above (NOWAIT, WAITIGNORE, and INVERTRESULT) can also be used to modify how the call is handled and how the return value is interpreted.	DLL \windows\MyLibrary.dll “test”
`Sleep`	Sleep (delay execution) for a specified number of seconds.	SLEEP 30
`Invoke a Batch File`	Invoke a Batch File (see the following section on Batch Files). Note that a Batch File cannot call another Batch File.	BATCH \temp\MyBatch.txt

Batch Files

Sometimes there is a need to do more than one thing in a Package Command. To make this possible, the “Invoke a Batch File” Package Format can be used. A Batch File allows multiple Package Commands to be placed into a text file, with one Package Command per line of the text file. Every line in the Batch File must have a trailing carriage return, as is standard for any text file.

Important:

If the last line in a Batch File does have a trailing carriage return, then it may not be executed.

A Batch File would typically be included into a Package and would then be referenced from a Package Command by preceding the full path and file name of the Batch File with the “BATCH” keyword.

Important:

All Package Command Formats can be used for Package Commands in a Batch File except the “Invoke a Batch File” Format. In other words, a Batch File cannot call another Batch File.

When executing a Batch File, the MSP Agent processes each line in the Batch File in the order they appear in the file. If any Package Command results in a failure (based on the rules defined for success and failure based on the Command Format used), then processing of the Batch File fails and the return code that caused the failure is returned as the return code of the Batch File. If all Package Commands in the Batch File are executed successfully, then zero (0) is returned as the return code of the Batch File.

Batch Files can be used for other purposes than Package Commands. In particular, the LockAndWipe Control Module allows a Batch File to be executed as part of a Wipe Action. Any time a Batch File is executed, it follows the same results defined above.

Windows Device Client MSP Agent Registry Interface

The Windows Device Client MSP Agent stores its configuration information in the Registry and, on some devices, stores this configuration Persistently in a Registry file in a folder in the super-Persistent File System (e.g. \Application\airbeam.reg). Since the MSP Agent leverages much of the same underlying code as the legacy AirBEAM Smart Client, the Registry interface for the MSP Agent is stored in the same location in the Registry and is Persisted in the same file in the File System.

Since both the MSP Agent and the RD Client perform deployment of Bundles and Packages, they share common code. As a result, they also share many of the same Registry Entries. For example, Relay Server Settings applied to a device when using the RD Client are stored identically to Relay Server Settings applied to a device when using a Provisioning Job.

Many of the individual Registry Entries defined by the legacy AirBEAM Smart Client continue to serve the identical function for the MSP Agent. These will not be detailed in this document but can be found in the following document titled “AirBEAM Smart Windows CE Client Product Reference Guide (P/N 72-63060-01 Rev. B)”, which is available for download from the AirBEAM Smart sub-section of the Mobility Software section at http://support.symbol.com/support/product/softwaredownloads.do. In addition, there are a number of new Registry Entries that are used only by the MSP Agent and are detailed in this section.

It is important to note that while the Registry interface of the MSP Agent can be used to configure the MSP Agent, it generally should not be used. The MSP Agent is designed to be configured using Settings Objects applied by Staging using the RD Client or by Provisioning Jobs using the MSP Agent itself. Changes applied through Settings Objects will be applied automatically to the Registry as required and Persisted automatically when appropriate. So, it should seldom be necessary or advisable to directly change the Registry interface of the MSP Agent directly.

Nonetheless, it may be useful to examine the Registry interface of the MSP Agent when troubleshooting a system and hence it is documented for those cases when it may be important to know.

The base Registry path where AirBEAM Smart Client and the MSP Agent store their configuration is:

HKEY_LOCAL_MACHINE\SOFTWARE\AIRBEAM

The new Registry values that are used only by the_MSP Agent are listed below:

ENABLE30

This Registry Value is used to indicate that the MSP Agent is enabled to function as an MSP Agent. This effectively disables other modes of client operation, including the legacy AirBEAM Smart Client and the legacy MSP 2.x Level 2 Agent.

It is important to note that all the components of the Windows Device Client are inter-related and inter-dependent. It is by design that any single device cannot be managed at any one time using more than one of the following: the legacy AirBEAM Smart Client, the legacy MSP 2.x Level 2 Agent, or the MSP Agent.

SCHEDULEMODE30

This Registry Value is used to indicate when the MSP Agent should automatically perform scheduled execution. The possible values for this Registry Value are:

- Interval

  This value indicates that the MSP Agent should schedule itself to execute if the device is active and the MSP Agent has not executed for at least the interval specified by the SCHEDULEINTERVAL30 Registry Value.

  When the MSP Agent completes a cycle of execution, it will schedule itself to execute after the specified number of minutes. If the MSP Agent is executed for some unscheduled reason (e.g. is invoked from the MSP Agent UI), then it will schedule itself to execute after the specified number of minutes from the completion of that unscheduled execution.

- Fixed Time

  This value indicates that the MSP Agent should schedule itself to execute at the same specified fixed time each day if the device is active.

  When the MSP Agent completes a cycle of execution, it will schedule itself to execute the next time the current time on the device matches the hour specified by the SCHEDULEHOUR30 Registry Value and the minute specified by the SCHEDULEMINUTE30 Registry Value.

- Fixed Time and Interval

  This value indicates that the MSP Agent should combine the functionality described above for Interval and Fixed Time by scheduling itself to execute at the soonest time that meets either criterion.

  When the MSP Agent completes a cycle of execution, it will schedule itself to execute at the specified fixed time or after the specified interval, whichever comes first.

Important:

Scheduled execution does not mean that the MSP Agent will always or only execute at a scheduled time. Depending on the circumstances on the device, the MSP Agent may execute sooner (e.g. the Device User issues a request from the MSP Agent UI) or the MSP Agent may not execute until much later (e.g. the device is suspended at the scheduled time).

Scheduled execution does not mean that the MSP Agent will always communicate to the Relay Server when it executes. There could be connectivity issues that prevent the MSP Agent from reaching the Relay Server. There also could be Check-In Conditions that limit when the MSP Agent is allowed to attempt to contact the Relay Server. If the MSP Agent cannot contact the Relay Server on a given execution, it may do so on a subsequent scheduled execution.

SCHEDULEINTERVAL30

This Registry Value is used only when the SCHEDULELOADMODE30 Registry Value is Interval or Fixed Time and Interval. This Registry Value specifies the number of minutes to be used as the interval for scheduling execution of the MSP Agent.

A value of zero for this Registry Value indicates that interval-based scheduled execution should not occur.

SCHEDULEHOUR30

This Registry Value is used only when the SCHEDULELOADMODE30 Registry Value is Fixed Time or Fixed Time and Interval. This Registry Value specifies the hour part of the fixed time for scheduling execution of the MSP Agent.

SCHEDULEMINUTE30

This Registry Value is used only when the SCHEDULELOADMODE30 Registry Value is Fixed Time or Fixed Time and Interval. This Registry Value specifies the minute part of the time for scheduling execution of the MSP Agent.

SCHEDULEWAKE30

This Registry Value is used to indicate whether the MSP Agent should arrange to wake up the device if the device is suspended at time the MSP Agent has scheduled itself to execute.

The possible values for this Registry Value are:

- **False** – The MSP Agent does not arrange for the device to wake up if it is suspended at the next time the MSP Agent has scheduled itself to execute.
- **True** – The MSP Agent does arrange for the device to wake up if it is suspended at the next time the MSP Agent has scheduled itself to execute.

It is important to note that this Registry Value only ensures that the device will wake up if it is suspended at the time the MSP Agent has scheduled itself to execute. It does not guarantee what the MSP Agent can or will do when it executes. See the notes for the Registry Value SCHEDULEMODE30 for examples.

FORCEREGDOC30

This Registry Value is used to indicate how often the MSP Agent should force a discovery document (regDoc) to be uploaded to the Relay Server even if the contents of that document has not changed. The MSP Server only updates the “Last Discovered” time for a device when a discovery document is received from that device. So there can be value in sending a discovery document even if no content has changed.

If this Registry Value is set to a value of 1 (as is done by the standard debug30 Package), then a discovery document is sent every time the MSP Agent Checks-In with the Relay Server, even if nothing in that discovery document has changed. This provides the best possible tracking of the “Last Discovered” time and may be useful when debugging or when performing demonstrations. But this would not generally be recommended for production use due to the potentially high impact it might have on network bandwidth. Some exceptions to this might be:

- The MSP Agent is being configured to Check-In with the Relay Server relatively infrequently (e.g. every 4 hours) and hence it is reasonable to send a discovery document every time.
- Plenty of network bandwidth is available between the devices and the Relay Server and the MSP Server and the Relay Server to handle the extra quantity of discovery documents that would be sent.

In most cases, it is better to set this Registry Value to a value larger than 1. The standard enable30 Package sets this Registry Value to a value of 32 and sets the SCHEDULEINTERVAL30 Registry Value to a value of 15. That combination causes the MSP Agent to schedule execution every 15 minutes and send a discovery document every 32 times it Checks-In with the Relay Server, whether the discovery document has changed or not. The net effect is that a discovery document is forced after 8 hours if it has not been sent sooner due to a change in content.

Important:

If your devices have Windows Device Client version 3.XX, use enable30 and debug30.

If your devices have RD Client or AirBEAM Client version 2.XX or older, use enable30legacy and debug30legacy.

It is important to note that this Registry Value only controls whether the sending of discovery documents is suppressed based on whether or not they have not changed. If the MSP Agent cannot contact the Relay Server, due to connectivity issues or due to limitations imposed by Check-In Conditions, then a discovery document may not be sent even if a sufficient number of attempts to Check-In with the Relay Server have been made. But once the MSP Agent is able to successfully Check-In with the Relay Server, a discovery document will be sent, whether it has changed or not, if the requisite number of attempts have been made since a discovery document was last sent.

DISABLEOPTIONS30

This Registry Value is used to disable various MSP Agent user interface features. This Registry Value consists of a single value which can contain one or more options by ORing together multiple “bit options”. The possible bit option values for this Registry Value are:

- Disable UI

  This bit option value indicates that the MSP Agent user interface should be completely disabled. This means that the device User will not be able to launch the will MSP Agent user interface, such as:

  - The icon for the MSP Agent will remain in the sys tray, but it will not be possible to launch the MSP Agent user interface from that icon.
  - The shortcut placed on the start menu when the Windows Device Client was installed will not be able to be used to invoke the MSP Agent user interface.
  - The –U and –M options described above in the Programmatic Interface section will not be able to be used to invoke the MSP Agent user interface.

- Disable exit

  This bit option value indicates that the Exit option will not be presented on the main menu of the MSP Agent user interface. This means that the Device User will not be able to use the MSP Agent user interface to cause the MSP Agent to stop executing.

- Disable delete Package

  This bit option value indicates that the Delete Package option should not be presented when a Package is selected from the Packages sub-menu of the MSP Agent user interface. This means that the Device User will not be able to delete Packages from the MSP Agent user interface.

CONTROLUPLOAD30

This Registry Value is used to control how often data collected by the device is uploaded to the Relay Server. When a device is configured to collect data on a defined schedule, it collects that data on the specified scheduled times so long as the device is not suspended at the time the sample is due. But collected data is only uploaded to the Relay Server when the MSP Agent Checks-In with the Relay Server.

If the collection rate and the scheduled execution rate of the MSP Agent rate are similar, then the MSP Agent could end up sending a Collection Document every time it Checks-In with the Relay Server. While this could be good from a data availability perspective, it could be undesirable from a network bandwidth perspective. Collection Documents are compressed and hence it is more efficient to store multiple collection samples in the same Collection Document before uploading it.

By setting this Registry Value to a value greater than 1, the MSP Agent will allow the Collection Document to continue to accumulate collection samples until the specified number of Check-Ins with the Relay Server times have occured. This may increase the latency of data availability to the MSP Console UI, but can have a dramatic benefit in reducing the network traffic for collecting data from devices.

LOGSIZE30

This Registry Value is used to indicate the maximum size (in KB) to which the MSP Agent is allowed to grow the log file. When the log file reaches the specified size, it is renamed to .old and then a new file is created. Only the most recent .old copy is kept, hence the maximum size the MSP Agent will use for log files is twice the size specified in this Registry Value.

The minimum log file size is 64 and the default is 200.

JOBLOGLEVEL30

This Registry Value is used to indicate the amount of detail the MSP Agent will place into the log file when performing job-related operations. The possible values for this Registry Value are:

- Critical

  This value indicates that only information about critical failures will be written to the log file. This value causes the smallest rate of growth in the log file but also provides the least information. This value should be chosen only if space or performance issues require that logging be reduced to the absolute minimum.

- Error

  This value indicates that information about non-critical errors and critical failures will be written to the log file. This value generally creates a modest rate of growth in the log file and provides information about issues that may need to be addressed. This value is a good choice for most production use situations.

- Warning

  This value indicates that information about warnings, non-critical errors, and critical failures will be written to the log file. This value generally creates a higher rate of growth in the log file but provides additional information that may be useful during the testing of new software or processes.

- Information

  This value indicates that a lot of explanatory information is written to the log file along with information about warnings, non-critical errors, and critical failures. This value generally creates a high rate of growth in the log file and provides additional information that would likely be useful only during detailed debugging scenarios.

- Verbose

  This value indicates that maximum amount of information is written to the log file along with information about warnings, non-critical errors, and critical failures. This value generally creates a very high rate of growth in the log file and should only be used when a complex issue cannot be debugged any other way.

LOGLEVEL30

This Registry Value is used to indicate the amount of detail the MSP Agent will place into the log file when performing non-job-related operations. The possible values for this Registry Value are the same as those shown above for the JOBLOGLEVEL30 Registry Value.

Windows Device Client Availability

Pre-Installation on Motorola Windows Devices

The Windows Device Client comes pre-installed on many Motorola (and Symbol) devices running the Microsoft Windows CE and Microsoft Windows Mobile Operating Systems. For information about specific devices and versions supported, consult the MSP 3.3.1 Release Notes. For devices and versions not listed in the release notes for MSP, consult the release notes for the specific device or device software version.

Devices that ship with a recent version of the Windows Device Client pre-installed are referred to as MSP-Ready. Device that ship MSP-Ready can immediately be used to perform MSP Staging functionality.

Notes:

Devices that do not come with a recent version of the Windows Device Client pre-installed will not have full Staging functionality when they are fresh-out-of-the-box. Such devices can be updated to have full Staging functionality by upgrading them to a suitable newer version of the Windows Device Client.

Updating the Windows Device Client may be performed using the standard (non-Legacy) Staging Process (albeit with some limitations) if a device has an older version of the Windows Device Client.

Updating the Windows Device Client may require using the Legacy Staging Process, as described in Using MSP 3.3.1 – Using Staging, if the device has a very old version of the RD Client that pre-dates the Windows Device Client (e.g. before version 3.0).

Updating the Windows Device Client may require using the Legacy AirBEAM Client if the device has no version of the RD Client at all.

Devices that are shipped MSP-Ready **do not** ship with the MSP Client Software scheduled to execute automatically. Most MSP Management Functionality, including Provisioning, relies on the automatic scheduled execution of the MSP Client Software. This must be configured for each device, typically by Staging appropriate Settings, either explicitly, by applying an Agent.30 Settings Object or implicitly, by deploying one of the standard Packages enable30.apf and debug30.apf.

Generic Windows Device Client .CAB Files

On a few supported Motorola devices and on all supported non-Motorola devices, the Windows Device Client components are not pre-installed. For such devices, a Generic Windows Device Client .CAB File must be used to install the Windows Device Client if the device is to be Directly Managed by MSP. Since the Windows Device Client is not yet on the device, MSP cannot be used to deploy or install the .CAB file.

Important:

Devices that do not ship with the Windows Device Client components pre-installed will only be officially supported by Motorola for use with a given version of MSP if they have been tested and certified by Motorola for use with that version of MSP.

All devices that are officially supported using the Generic Windows Device Client .CAB Files will be documented in the Release Notes for the version of MSP with which they were tested and certified. While additional devices that have not been tested and certified by Motorola might appear to work with MSP, such use will not be warranted or supported by Motorola.

Always check the Release Notes for the version of MSP with which devices that do not ship with the Windows Device Client components pre-installed will be used to determine if they are officially supported by that version of MSP. If a desired device is not list as officially supported, you should submit a request for official support of the device, via your Motorola or partner sales representative, before relying on MSP support for the device.

Obtaining Generic Windows Device Client .CAB Files

Generic Windows Device Client .CAB Files are provided via a Generic Windows Device Client .ZIP File that is separately available for download from the following link:

http://support.symbol.com/support/product/softwaredownloads.do

.ZIP File Kit Contents

A Generic Windows Device Client .ZIP File will have a name of the form:

MSP_GenericClient_<version>.ZIP

where:

<version>	indicates the version number of the software delivered in the .ZIP File.
<date>	indicates the release date of the.ZIP File (represented in YYYYDDMM format).

Each Generic Windows Device Client .ZIP File contains the following contents:

InstallMSPCert.exe

MspAgent_ddd_<version>.cab

MspAgent_dwd_<version>.cab

MSP 3.0 Client Release Notes.txt

Once a Generic Windows Device Client .CAB file has been installed on a device, it becomes MSP-Ready and is in most ways comparable to a device that shipped with the Windows Device Client components pre-installed.

Note:

The Windows Device Client is called generic because it is designed to rely on a common denominator of functionality that enables it to run on a wide variety of Windows Mobile devices. Some devices may lack certain features or have known issues, and thus may be unable to successfully support certain MSP functionality. In addition, some advanced MSP functionality is supported only on Motorola devices. Consult the Release Notes for a given version of MSP for information about any limitations in MSP functionality that apply to a particular device.

Once the Windows Device Client has been installed on a device using a Generic Windows Device Client .CAB File, scheduled execution of the MSP Client Software would generally need to be configured for the device. This is typically accomplished by Staging appropriate Settings, either explicitly, by applying an Agent.30 Settings Object or implicitly, by deploying one of the standard Packages enable30.apf and debug30.apf.

Important:

Generic Windows Device Client .CAB Files are intended **only** for the **initial** installation of the Windows Device Client on a device that does not already have **any** version of the Windows Device Client resident.

Generic Windows Device Client .CAB Files are **never** a supported mechanism to update the Windows Device Client on a device that already has **any** version of the Windows Device Client resident.

To update the Device Client on a device that already has **any** version of the Windows Device Client resident, Windows Device Client Update Packages should be used.

Generic Windows Device Client .CAB Files follow a common naming format of “MspAgent_xxx”, as shown in Table 4 below.

Table 4 – Generic Windows Device Client .CAB Files

Feature Set	Supported Devices	.CAB File Name	Recommended Usage
Persistent❶ Default Standard	All devices except “locked” Windows Mobile devices, such as the Motorola EWP/TEAM or HC700 devices.	MspAgent_ddd.<version>cab	Best for all but excluded devices where the Windows Device Client is not pre-installed. If execution of Detached Jobs is required, a Windows Device Client Update Package will need to be loaded.
Persistent❶ Windows Mobile Standard	All devices running Windows Mobile 5.0 or higher (not Windows CE) Operating Systems	MspAgent_dwd_<version>.cab	Best for “locked” Windows Mobile devices where the Windows Device Client is not pre-installed. If execution of Detached Jobs is required, a Windows Device Client Update Package will need to be loaded.

❶ Generic Windows Device Client .CAB Files are designed for installing the Windows Device Client onto devices that do not ship with the Windows Device Client pre-installed. Such devices may or may not be capable of supporting Persistence across a Restore Boot. On supported Motorola devices, Persistence will be supported if possible. On supported non-Motorola devices, Persistence generally will not be supported.

.ZIP File Installation

No specific installation process is required or provided for a Generic Windows Device Client .ZIP File. Simply extract the files from the .ZIP File and use them as appropriate.

Note:

The deployment and installation of Generic Windows Device Client .CAB files onto devices may be performed manually or using any other process that is suitable for a given target device. MSP does not provide any mechanism to assist with the deployment or installation of .CAB files onto such devices. Once a suitable Generic Windows Device Client .CAB file has been deployed and installed onto a device, then that device becomes MSP Ready and MSP can be used to Stage the device.

Installing MSP Certificates on Generic Devices

Some Windows Mobile devices are shipped "locked", by which is meant they require that all new programs loaded onto the device be authorized as certified to run on the device. Authorization for individual programs can be performed by the Device User the first time each program attempts to run on the device. Authorization can also be performed by signing programs with a certificate that is trusted to pre-certify programs for that device.

All programs provided as part of MSP are signed with a Motorola-owned certificate. If a device is locked, but has the Motorola certificate installed, then the program will be considered pre-certified and the Device User will not be asked to authorize the program. The utility "InstallMSPCert.EXE" is provided as part of the Windows Device Client Add-On Kit .ZIP file. This utility can be used to install the Motorola certificate onto any Windows Device. It can be manually copied to a device and executed or a Package can be created to deploy and execute the utility on one or more devices.

Note:

The Device User will need to explicitly authorize the InstallMSPCert.EXE utility as certified to run on a device before it can do its jobs of installing the Motorola certificate on that device.

Windows Device Client Update Packages

The Windows Device Client could be resident on a particular device as a result of pre-installation, could have been installed via a Generic Windows Device Client .CAB File, or could hgave been previously upgraded via a Windows Device Client Update Package. Whatever method was used to make the Windows Device Client resident on a device, it should **only** be updated (e.g. to fix bugs and/or add new features) by deploying a Windows Device Client Update Package to the device.

Important:

Windows Device Client Update Packages are the **only** supported mechanism to update the Windows Device Client on a device that already has the Windows Device Client resident.

A .CAB File is **never** a supported mechanism to update the Windows Device Client on a device that already has the Windows Device Client resident.

The Windows Device Client that is resident on a device includes the Rapid Deployment Client. The Rapid Deployment Client can be used to perform MSP Staging to deploy a Windows Device Client Update Package to the device.

The Windows Device Client that is resident on a device includes the MSP Agent. If scheduled execution has been configured, then the MSP Agent can be updated from MSP by deploying a Windows Device Client Update Package via an MSP Provisioning Policy or an MSP Action.

Obtaining Windows Device Client Updates

Windows Device Client Update Packages are provided as part of the MSP Server and via Windows Device Client Add-On Kits that are separately available for download from the following link:

http://support.symbol.com/support/product/softwaredownloads.do

Add-On Kit Contents

A Windows Device Client Add-On-Kit will be a .ZIP File with a name of the form:

WindowsDeviceClient_<version>.ZIP

where:

<version>	indicates the version number of the software delivered in the Add-On Kit.
<date>	indicates the release date of the Add-On Kit .ZIP File (represented in YYYYDDMM format).

Each Windows Device Client Add-On-Kit Add-On Kit .ZIP File contains the following contents:

abup_30_ddd_<version>.apf

abup_30_ddp_<version>.apf

abup_30_dwd_<version>.apf

abup_30_dwp_<version>.apf

abup_30_nwd_<version>.apf

abup_30_nwp_<version>.apf

MSP 3.0 Client Release Notes.txt

Detail on the above Windows Device Client Update Packages is provided in Table 5 below. It is generally advisable to read the table from top to bottom and select the first Windows Device Client Update Package that suits your needs.

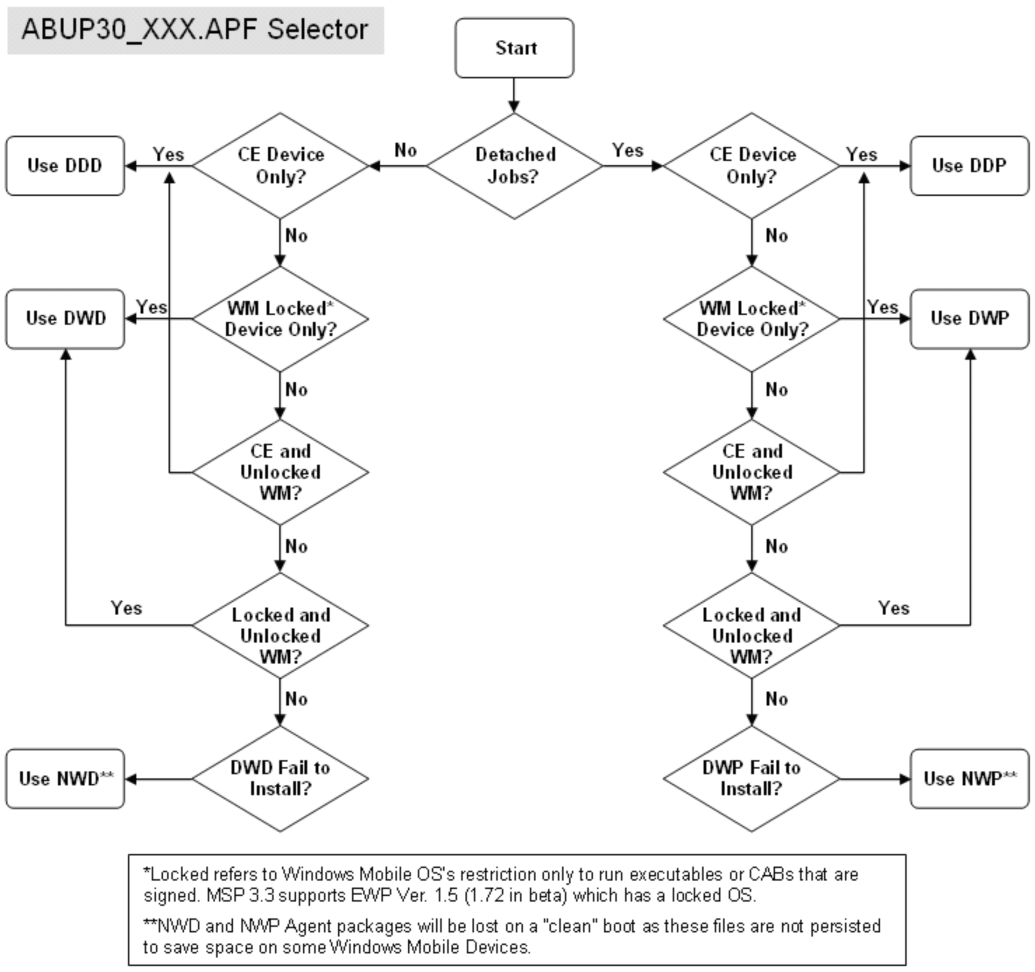
Table 5 – Windows Device Client Update Packages

Feature Set	Supported Devices	Package Name	Recommended Usage
Persistent Default Standard	All devices except "locked" Windows Mobile devices, such as the Motorola EWP/TEAM or HC700 devices.	abup30_ddd	Best for all but excluded devices if execution of Detached Jobs is not required.

Feature Set	Supported Devices	Package Name	Recommended Usage
Persistent Windows Mobile Standard	All devices running Windows Mobile 5.0 or higher (not Windows CE) Operating Systems	abup30_dwd	Best for "locked" Windows Mobile devices if execution of Detached Jobs is not required. Can be used for mixed populations of "locked" and "unlocked" devices running Windows Mobile Windows Mobile 5.0 or higher Operating Systems if execution of Detached Jobs is not required.
Persistent Default Proxy	All devices except EWP/TEAM	abup30_ddp	Best for all but excluded devices if execution of Detached Jobs is required.
Persistent Windows Mobile Proxy	All devices running Windows Mobile 5.0 or higher (not Windows CE) Operating Systems	abup30_dwp	Best for "locked" Windows Mobile devices if execution of Detached Jobs is required. Can be used for mixed populations of "locked" and "unlocked" devices running Windows Mobile Windows Mobile 5.0 or higher Operating Systems if execution of Detached Jobs is required.
Non-Persistent Windows Mobile Standard	All Symbol and Motorola devices running Windows Mobile 5.0 or higher (not Windows CE) Operating Systems and that have a persistent \Application folder	abup30_nwd	May be preferable for some Symbol and Motorola devices running Windows Mobile Windows Mobile 5.0 or higher Operating Systems if storage space in \Application is at a premium, persistence is not required, and execution of Detached Jobs is not required.

Feature Set	Supported Devices	Package Name	Recommended Usage
Non-Persistent Windows Mobile Proxy	All Symbol and Motorola devices running Windows Mobile 5.0 or higher (not Windows CE) Operating Systems and that have a persistent \Application folder	abup30_nwp	May be preferable for some Symbol and Motorola devices running Windows Mobile Windows Mobile 5.0 or higher Operating Systems if storage space in \Application is at a premium, persistence is not required, and execution of Detached Jobs is required.

To make it easier to select the proper Windows Device Client Update Package for a given device, below provides a flowchart illustrating the process.



*Locked refers to Windows Mobile OS's restriction only to run executables or CABs that are signed. MSP 3.3 supports EWP Ver. 1.5 (1.72 in beta) which has a locked OS.

**NWD and NWP Agent packages will be lost on a "clean" boot as these files are not persisted to save space on some Windows Mobile Devices.

Figure 4 – Flowchart for Selecting Windows Device Client Update Packages

Add-On Kit Installation

For Users with direct access to Windows Console on the MSP Server, an Add-On Kit .ZIP Files can be installed using the MSP Administration Program. See Add-On Kit Installation in Administering MSP 3.3.1.

Note:

The installation process, described above, does not automatically make the documents contained within the Add-On Kit available.

Windows Device Client Management Functionality

This section and the following sub-sections will provide a high-level overview of the MSP Management Functionality provided when a Windows Device is Directly Managed via the Windows Device Client.

Note:

The intent of this document is to provide an understanding of the capabilities of the MSP Client Software. This section will thus provide a high-level explanation of the functions supported by the Windows Device Client, but will not cover the exact features and functions supported on any given device, Operating System version, etc. For more information on specific devices supported by MSP, see the MSP 3.3.1 Release Notes.

Staging

Since the RD Client is one of the available components of the Windows Device Client, Staging is available for use on Windows Devices, when used with MSP Stage Edition, MSP Provision Edition, or MSP Control Edition. Staging is considered a key function provided by the Windows Device Client. Staging would typically be used to perform the initial configuration and deployment of a Windows Device and the configuration of the MSP Agent that is required to make the Windows Device Directly Manageable by MSP.

Asset Management

Since the MSP Agent is one of the available components of the Windows Device Client, Asset Management is available for use on Windows Devices when used with MSP Provision Edition or MSP Control Edition. Asset Management is considered a key function provided by the Windows Device Client.

Asset Management includes discovery of Windows Devices by MSP, the acquisition and reporting to MSP of values for a variety of Device Attributes for Windows Devices, and the tracking of the inventories of Packages deployed on Windows Devices.

The key Device Attributes supported by the Windows Device Client are listed in Table 6 below:

Table 6 – Window Device Client Key Device Attributes

Device Attribute Name	Explanation
Agent.Feature	The value of this Device Attribute indicates the level of functionality being used by the MSP Agent and hence, the type of license required to manage the device within MSP. Possible values are: Provision or Control.
Agent.SupportsCheckInConditions	The value of this Device Attribute indicates whether the MSP Agent supports the use of Check-In Conditions to limit when it can contact the Relay Server. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsCheckInPackages	The value of this Device Attribute indicates whether the MSP Agent supports the use of Check-In Commands in Packages to allow Packages to perform periodic activity. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsDisableOptions	The value of this Device Attribute indicates whether the MSP Agent supports to configure whether various MSP Agent UI Options will be accessible. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsExtendedCommands	The value of this Device Attribute indicates whether the MSP Agent supports Extended Commands (e.g. NOWAIT, WAITIGNORE, etc.) in Package Commands (e.g. Install Command, Uninstall Command, Check-In Command) in Batch Files. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsInlineContent	The value of this Device Attribute indicates whether the MSP Agent supports Inline Content, which is required to support Dynamic Provisioning. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsLimitJobs	The value of this Device Attribute indicates whether the MSP Agent supports the ability to honor the limit on the number of Jobs that can be simultaneously executed on a given Relay Server. Possible values are: 1 = Yes and 0 = No.

Device Attribute Name	Explanation
Agent.SupportsProxyPlugIns	The value of this Device Attribute indicates whether the MSP Agent supports the ability to host Proxy Plug-Ins. Possible values are: 1 = Yes and 0 = No.
Identity.ClientVersion	The value of this Device Attribute indicates the version of the MSP Client Software that is currently installed on a managed device.
Identity.DeviceClass	The value of this Device Attribute indicates the Device Class of a managed device.
Identity.DeviceOS	The value of this Device Attribute indicates the Operating System running on a managed device.
Identity.FlashSize	The value of this Device Attribute indicates the size of the flash memory, if any, that is present on a managed device.
Identity.OemVersion	The value of this Device Attribute indicates the OEM version, if any, of the software image that is present on a managed device.
Identity.PlatformID	The value of this Device Attribute indicates the Platform ID, if any, of the software image that is present on a managed device.
Identity.UUID	The value of this Device Attribute indicates the UUID value that identifies the Windows PC to MSP as a unique managed device.
Identity.WlanPlugInClass	The value of this Device Attribute indicates the Class ID of the default WLAN Settings Plug-In, if any, that is in use on a managed device.
Identity.WlanPlugInVendor	The value of this Device Attribute indicates the Vendor ID of the default WLAN Settings Plug-In, if any, that is in use on a managed device.

Provisioning

Since the MSP Agent is one of the available components of the Windows Device Client, Provisioning is available for use on Windows Devices when used with MSP Provision Edition or MSP Control Edition.

Provisioning is considered a key function provided by the Windows Device Client. Provisioning provides the ability to define Policies that automatically or manually control the installation and uninstallation of Packages to Windows Devices.

Remote Control and Troubleshooting

Since the MSP Agent is one of the available components of the Windows Device Client, Remote Control and Troubleshooting is available for use on Windows Devices when used with MSP Provision Edition or MSP Control Edition. Remote Control and Troubleshooting is considered a key function provided by the Windows Device Client.

Remote Control and Troubleshooting provides the ability to deploy a Remote Control Solution that enables a Workstation PC to perform Real-Time control of the UI of a Windows Device and to perform other troubleshooting functions, such as file management, Registry management, etc.

Actions

Since the MSP Agent is one of the available components of the Windows Device Client, Actions are available for use on Windows Devices when used with MSP Control Edition. Actions are considered a key function provided by the Windows Device Client.

Actions provide the ability to explicitly request the installation and uninstallation of Packages to Windows Devices.

Data Collection

Since the MSP Agent is one of the available components of the Windows Device Client, Data Collection is available for use on Windows Devices when used with MSP Control Edition. Data Collection is considered a key function provided by the Windows Device Client.

Data Collection provides the ability to deploy a Data Collection Solution that enables the periodic acquisition and reporting to MSP of samples of selected Collection Metrics from Windows Devices.

Real-Time Management

Since the MSP Agent is one of the available components of the Windows Device Client, Real-Time Management is a key function provided by the Windows Device Client when used with MSP Control Edition.

Real-Time Management provides the ability to deploy a Real-Time Management Solution to Windows Devices that enables a Workstation PC to perform operations on one or more Windows Devices in real-time via direct connections to those Windows Devices.

Windows Device Client Troubleshooting

Windows Device Client MSP Agent Log File

The Windows Device Client MSP Agent maintains a record during its normal processing of the activities it performs and the results of those activities. The MSP Agent Log File can only be viewed by accessing the file on the Windows Device, it is never sent to the MSP Server, and hence is never visible from, the MSP Console UI.

The Log Level can be configured as part of the MSP Agent configuration as described previously in this chapter. The Log Level controls the amount of information recorded into the MSP Agent Log File.

The MSP Agent Log File is stored by the MSP Agent into the folder “\Windows”. Within that folder, the file “Agent.log” contains the currently active (most recently recorded) log information and the file “Agent.log.old” contains the archive (least recently recorded) log information.

The MSP Agent Log File can be viewed from the Log Menu of the MSP Agent UI on the Windows Device as described earlier in this chapter. The MSP Agent Log File is a text file and can also be viewed directly on the Windows Device using any suitable text editor (e.g. Notes, Pocket Word, etc.).

The Log Size can be configured as part of the MSP Agent configuration as described previously in this chapter. The Log Size controls the maximum size the two files can be reach. When the file “Agent.log” reaches the specified size, the file “Agent.log.old” is deleted, if it exists, the file “Agent.log” is renamed to “Agent.log.old, a new empty file “Agent.log” is created, and new recording of log information is added to the new file. This means that the combined size of the two files will never total to more than twice the specified Log Size.

Windows Device Client MSP Agent Job Log File

The Windows Device Client MSP Agent maintains a record during the processing for each Job of the activities it performs and the results of those activities. The Job log information for each Job is uploaded to the MSP Server, via the Relay Server, when a Job is completed, and hence can be displayed from the MSP Console UI for any completed Job.

The Job Log Level can be configured as part of the MSP Agent configuration as described previously in this chapter. The Job Log Level controls the amount of information recorded into the MSP Agent Job Log File.

The MSP Agent Job Log File is stored by the MSP Agent into the folder “\Windows”. Within that folder, the file “Job.log” contains the log information for the most recently executed Job.

The MSP Agent Job Log File can be viewed from the Log Menu of the MSP Agent UI on the Windows Device as described earlier in this chapter. The MSP Agent Job Log File is a text file and can also be viewed directly on the Windows Device using any suitable text editor (e.g. Notes, Pocket Word, etc.).

Windows Device Client Licensing

The licensing required to use the Windows Device Client for all supported devices is included in the per-device license required to manage a device using a given MSP Edition (e.g. Stage, Provision, or Control). No additional Windows Device Client licensing is required.

Windows Device Client Best Practices

General Best Practices

- Check Device Class in Applicability Rules

  When defining Applicability Rules in Provisioning Policies, Actions, and Data Collection Requests, include a check for the Device Attribute “Identity.DeviceClass” being equal to “Mobile Device” whenever it is important that Applicability be limited to Windows Devices.

- Use an On-Going Provisioning Policy to ensure that the latest Windows Device Client is continuously kept up to date on all devices.

Over time, the Windows Device Client will be updated to add new features. While EVERY version of the Windows Device Client (from 3.0 onwards) is capable of operating, at least to some degree, with every version of MSP (from MSP 3.0 onwards), not all functionality advertised for a given version of MSP may be available if an older Windows Device Client version is present on the device.

Luckily, every version of Windows Device Client (from 3.0 onwards) is capable of updating itself to every newer version under the control of every version of MSP (from 3.0 onwards). As a result, any device with any version of Windows Device Client (from 3.0 onwards) on it can be updated using a suitable Provisioning Policy.

For fresh-out-of-the-box devices, the Staging capabilities will be limited by the capabilities of the Windows Device Client pre-loaded in the device. While Staging can, and often will, be used to update the Windows Device Client to a newer version, this must be done within the constraints of what is pre-loaded in the device. So, it may not be possible to Stage the device using the most advanced WLAN security options, for example, if the device ships with an older version of the Windows Device Client that does not support the desired option.

In such cases, it might be necessary to perform two separate Staging operations. The first Staging operation would be to load a newer version of the Windows Device Client. This would need to be done over a “Staging network” which could be a limited-access, less-secure WLAN, a cradle connection, etc. Once an appropriate Windows Device Client version is loaded, then a second Staging operation can be used to achieve the desired results.

- Use On-Going Provisioning Policies to ensure that the latest optional management software Packages (e.g., RemoteControl, LockAndWipe, AutoSiteAssign, etc.) are deployed on the device before they are needed

  While these Packages could be deployed via Staging, it is often preferable to deploy them using Provisioning. Using Provisioning allows the full power of Provisioning Policies to be used to determine which devices get which Packages. It also allows the configuration of these Packages to be adjusted differently for different devices, to meet the needs of a variety of circumstances.

  As discussed earlier, this approach also reduces the complexity for the Device User and avoids the need for double maintenance.

- Use Manual Provisioning Policies to deploy selected Control Modules (e.g., RemoteControl, LockAndWipe) onto devices when needed

  Sometimes, it is desirable to reduce the footprint impact on devices by deploying some Control Modules “Just in Time” (e.g., only when they are needed). For example, it might make sense to deploy the LockAndWipe Control Module only when the device is about to be locked down or wiped.

OS Update Best Practices

- Assess and Optimize Network Connectivity

OS Update packages can be quite large. Providing adequate bandwidth between devices and the Relay Server can thus be essential to providing reliable operation as well as an acceptable level of performance. The type of network connectivity used will often have a significant impact. For example:

- Wired vs. Wireless

  Other factors being equal, using wired connectivity (e.g. Ethernet Cradles), will often provide a superior result over using wireless connectivity (e.g. Wireless LAN). Of course, the physical connectivity and the actual bandwidth of such a wired connection would need to be assessed to determine if it is adequate to the task.

- WLAN vs. WWAN

  Other factors being equal, using local connectivity (e.g. WLAN) will often provide a superior result over using wide area connectivity (e.g. WWAN). In fact, performing OS Updates over Wireless WAN can be a slow and costly process and due to the potential unreliability of WWAN connections, could also be risky and hence should be used with caution.

- Volume of WLAN Traffic

  Because of the typically high volume of traffic generated by OS Updates, it may be necessary to optimize the Wireless LAN network used in order to achieve an acceptable balance of reliability and performance. In particular:

  - Use Multiple Access Ports and/or Access Points

    The physical capacity of the infrastructure could become a bottleneck that could limit the ability to update many devices at once. By having multiple Access Ports and/or Access Points, the load may be distributed, resulting in higher performance and better reliability.

  - Use Multiple WLANs

    While using multiple Access Ports and/or Access Points may help, if all are implementing the same WLAN, then performance and reliability may suffer. By assigning multiple SSIDs, multiple logical networks can be created. This can make the multiple Access Ports and/or Access Points able to operate more independently of each other and potentially further improve performance and reliability.

  - Use Multiple Channels

    While using multiple WLANs may help, if all are operating on the same channel, they may be competing with each other for access to that channel. By using multiple channels, collisions may be reduced and performance and reliability may be further enhanced due to use or parallel bandwidth.

  - Use Multiple Bands

    While using multiple Channels may help, using multiple bands may help even more. While it might seem that using the highest through band (e.g. 802.11a) for all WLANs might be best, this means that all devices must compete for the available bandwidth on that band. By using multiple bands in parallel, when available, the available spectrum may be used more effectively. While devices operating on slower bands may take longer to complete the update, they may nonetheless complete faster than if they had to compete for a share of the faster band that was already overused.

    - Minimize Interference, Barriers, and Distance

      WLAN networks can be sensitive to interference from other RF sources, the interposition of barriers (e.g. reflective or absorbent materials), and the simple distance between devices and the WLAN infrastructure. In many cases, eliminating or reducing interference, barriers, and distance can significantly improve the reliability and performance of OS Updates.

    - Tune WLAN Infrastructure

      In some cases, the WLAN Infrastructure can be tuned to optimize for different types of traffic patterns. For example, tuning for quick turnaround for many short transactions may severely limit the performance of downloading large files, as is done during an OS Update. It is beyond the scope of this document to describe the methods for optimizing any WLAN other than to say that it can have a significant impact on the overall performance and reliability when performing OS Updates.

- Assess and Optimize Relay Servers

  Because of the typically high volume of traffic generated by OS Updates, it may be necessary to optimize the use of Relay Servers or the connectivity to Relay Servers in order to achieve an acceptable balance of reliability and performance. In particular:

  - Network Segment Quality of Service

    Some Enterprise Networks may limit the bandwidth allocated to certain types of traffic on certain ports. In particular, some Enterprises may intentionally limit FTP traffic on the standard FTP ports (e.g. TCP Port 21). Since OS Updates are performed using FTP, such limitations can directly limit how many OS Updates might be performed over a given network segment.

    If such limits are necessary and cannot be relaxed, due to Enterprise Policies, then it may be desirable to configure MSP Relay Servers to use non-standard ports which are not subject to such limitations.

    - Network Segment Capacity

      In some cases, the physical network segment between devices and a Relay Server may become be a bottleneck. This could be the case if the Relay Server is distant from the devices. It could also be the case if a network segment simply has a low capacity and/or is heavily utilized for other traffic. In such cases, that Relay Server may not be capable of supporting OS Updates at all, or may be limited in the number of devices for it can support OS Updates at once.

      In such cases, using multiple parallel Relay Servers, over different network segments may increase the potential for performing more simultaneous OS Updates, with better performance and reliability. While each Relay Server may be unable to support many devices, providing multiple paths to multiple Relay Servers can aggregate to the ability to support a larger total number of devices.

    - Relay Server Capacity

      In some cases, an FTP Server being used as a Relay Server could have limited capacity or bandwidth. This might occur if the FTP Server were being used to support other tasks as well. For example, a wireless switch that also serves as an FTP Server might have severely limited capacity and may be able to handle only a small number of simultaneous OS Updates.

While such a Relay Server might work fine under normal conditions, it may break down under the extreme load scenario of OS Updates. In such cases, adding one or more auxiliary Relay Servers, just for OS Update, could add the capacity necessary to achieve acceptable performance and reliability.

- Measure, Calculate, and Throttle

  In many cases, the best results for performing many OS Updates in parallel may be achieved by using a process of Measure, Calculate, and Throttle.

  - Measure

    By measure is meant running actual tests on the actual infrastructure and the actual Relay Servers, under real-world load conditions, to determine the actual behavior. In particular, it is important to determine the number of simultaneous OS Updates that can be performed to a given Relay Server over a given network segment and the total time taken to complete an OS Update.

    Notes:

    On devices running Windows Mobile 5.0 and higher, the OS Update process typically proceeds in two distinct phases: a download phase followed by an update phase. Once a device has completed the download phase and has begun the update phase, the device will no longer communicate on the network and a subsequent device could safely start the OS Update process.

    On other devices, the OS Update process may repeatedly cycle between downloading a partition and updating that partition. In such cases, it may not be safe to start OS Update process on a subsequent device until a device has completed the downloading of the last partition and this may be difficult to determine. In such cases, it may be necessary wait until the OS Update is fully complete on a device before starting the OS Update process on a subsequent device.

    The OS Update process varies by device and Operating System. Before performing measurements, it is generally advisable to become familiar with the OS Update process for a given device and to be sure you know when a device has completed all of its download activities as part of an OS Update.

  - Calculate

    Once the maximum number of OS Updates that can safely be performed is known, and once the total [download] time for a single OS Update is known, it is an simple matter to compute the number of OS Updates that can be accomplished within a desired duration. Assuming that all practical optimizations have been performed, it would be necessary to add more Relay Servers and/or more network capacity in order to expand beyond that limit.

- Throttle

  Once the practical limit has been determined for a given combination of Relay Server and network segment, reliable operation generally requires staying within that limit. This would generally consist of two parts. First, each device must perform its OS Update from a specifically chosen combination of Relay Server and network segment. Second, the number of devices that simultaneously perform OS Updates from the same Relay Server over the same network segment must be limited. The exact mechanism(s) used to accomplish this depends on how OS Updates will be initiated.

  - Staging

    When using Staging to initiate OS Updates, the throttling process would need to be a manual one, performed at each device. If we assume that devices are being Staged fresh out-of-the-box, then the Staging Profile applied to each device would typically define everything of importance, in particular the network segment used (e.g. WLAN Settings) and the Relay Server used.

    One simple way to handle throttling when initiating OS Updates via Staging is to have one Staging User handle the Staging for all devices that use a given combination of Relay Server and network segment. Each such Staging User could be assigned their own unique Staging Profile which they could use to initiate Staging on devices one after another until, the identified limit is reached. Then, the Staging User can Stage one more device each time a device completes.

  - Provisioning or Actions

    When using Provisioning or Actions to initiate OS Updates, the throttling process **might** be made be fully automatic. MSP permits a Relay Server Job limit to be established for each Relay Server. This limit controls the maximum number of Jobs that can simultaneously be executed from a single Relay Server at any one time. This limit is cooperatively enforced by all devices executing Jobs from the same Relay Server.

    The Relay Server Job limit would provide full automation of throttling if the bottleneck is due to the Relay Server itself or if there is a one-to-one correspondence between Relay Servers and network segments. In such cases, a single Policy or Action could be applied to a large set of devices and throttling would be completely automatic, based on the Relay Server Job limit of the Relay Server for each device.

    But if a network segment is the bottleneck and more than one Relay Server can be reached over the same network segment, then limiting Jobs based on Relay Servers will not be sufficient. Instead, a manual Job throttling approach would need to be used. Manual Job throttling typically requires multiple Policies or Actions to be created, such that no single Policy or Action is Applicable to devices that should not be allowed to execute Jobs at the same time.

    To handle this kind of situation, the Applicability Rules for each Policy or Action would typically be defined such that the devices that reach different Relay Servers over the same network segment would not be Applicable to the same Policy or Action. Then, by manually sequencing which Polices are enabled and/or which Actions are executed, the appropriate throttling could be achieved. Jobs will not be sent to devices that should not execute them in parallel, and hence the necessary limits will be honored.

Note:

As described above, initiating OS Updates via Provisioning or Actions **might** require manual action to be taken from the MSP Console UI. But manual action required at the MSP Console UI is likely **much** less costly than manual action required on devices, as would be required if Staging were used.

Windows Device Anomalies

This section explains differences in how the Windows Device Client operates on devices that have some different or unusual nature or behavior.

WT4070 and WT4090

Some models of the WT4070 and WT4090 "wearable" devices do not have a touch panel and hence must be navigated and control entirely using the keyboard. Because of the special nature of the keyboard on the WT4070 and WT4090 devices, they do not use the same key combinations to accomplish certain functions. In particular:

- When the User selects Exit to shut down the MSP Agent, the UI asks if you are sure you want to exit and the **No** button is highlighted. A User would expect that pressing the Enter button would select "No" and pressing the TAB or Right Arrow key would move to the **Yes** button, but this does not work on the WT4070 and WT4090 devices. Instead, the P1 and P2 keys are used instead. **P1** is used for **No** and **P2** is used for **Yes**.

Windows Device Client Error Codes

The Windows Device Client components (the Rapid Deployment Client, AirBEAM Smart Client, and the MSP Agent) can return error codes in the -8xx range and the -10xx range. The meanings of the error codes used by the Windows Device Client are shown in Table 7.

Table 7 – Windows Device Client Error Codes

Error Code	Description
-2	The RD Client was unable to enable the selected barcode scanner. This error occurs if the barcode scanner selected for use by the RD Client could not be successfully enabled. This could occur if the selected barcode scanner is not connected or is unavailable for use due to a conflict with some other device. To correct this issue, select a different barcode scanner or make the selected barcode scanner is available for use (e.g. connect the required barcode scanner, stop using the conflicting device, etc.). Then retry the operation.

Error Code	Description
-8	The RD Client was unable to request a barcode from the barcode scanner. This error occurs if another application is using the same barcode scanner and has requested a barcode of one of the same types used by the RD Client. The barcode scanner will not allow multiple applications to request the same barcode type at the same time, since it would not how which application to send the barcode to. To correct this issue, stop the other application that is using the barcode scanner, change the other application to not use any barcode types used by the RD Client, or select a different barcode scanner for use by the RD Client. Then retry the operation.
-801	A file on the device could not be created. This error occurs when a file could not be created on the device. This could be due to write-protection issues (e.g. of an existing file to be overwritten or of the destination folder). It could also be caused by insufficient space to hold the file in the device. To correct this issue, find and correct the reason why the file could not be created (e.g. remove write-protection, free up space, etc.). Then retry the operation.
-802	A required Package file does not exist on an FTP Server. This error occurs when a Package file (.APD file) to be downloaded to the device could not be found in the configured folder on an FTP Server. This should typically not occur unless the Content of the FTP Server has been altered in an out-of-band manner. To correct this issue, restore the missing Package file to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-803	A required Package file has an invalid format. This error occurs when a Package downloaded from an FTP Server had an invalid format. This should typically not occur unless the Content of the FTP Server has been altered in an out-of-band manner. To correct this issue, restore a valid copy of the Package file to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.

Error Code	Description
-805	A Package could not be downloaded to the device due to Insufficient space. This error occurs when the size defined for a Package is determined to be too large to fit on the device. This could occur if there is really insufficient space in the device to hold the Package. To correct the issue, free up some space on the device. Then retry the operation. This error could also occur if the Package incorrectly describes the space required on the device (e.g. the Package places files in a non-standard location on the device). Consult the AirBEAM Package Builder Guide for information on how to construct a Package that properly describes the space required on the device. Then retry the operation.
-806	An FTP user name was invalid. This error occurs if the user name specified for an FTP Server in a Relay Server Object was not accepted for login to the FTP Server whose IP Address or network name was configured for that Relay Server. To correct this issue, change the Relay Server Object to reference a valid user name or change the FTP Server to add the user name as a valid user. Then retry the operation.
-807	An FTP password was invalid. This error occurs if the password specified for an FTP Server in a Relay Server Object is not valid for logging into the account specified by the configured user name on the FTP Server whose IP Address or network name was configured for that Relay Server. To correct this issue, change the Relay Server Object to configure the correct password, change the user name to an account that uses the specified password, or change the password of the account on the FTP Server. Then retry the operation.
-808	An FTP account was invalid. This error occurs if the account selected by the user name specified for an FTP Server in a Relay Server Object was not a valid account for login to the FTP Server whose IP Address or network name was configured for that Relay Server. To correct this issue, change the Relay Server Object to reference a valid account or change the FTP Server to make the requested account valid. Then retry the operation.

Error Code	Description
-810	Binary mode could not be set for a file transfer to or from an FTP Server. This error occurs when an attempt to place an FTP Server into binary file transfer mode failed. This would generally indicate that the FTP Server cannot be used with MSP since it does not support the required commands and hence does not comply with the required RFCs. To correct this issue, change the Relay Server Object to reference an FTP Server that complies with the required RFCs. Then retry the operation.
-811	An FTP Server could not be contacted. This error occurs when an FTP Server could not be contacted at the IP Address or network name configured in a Relay Server Object. This could occur if the FTP Server is down, if network connectivity to the FTP Server is unavailable, or if the IP Address of network name configured for a Relay Server was incorrect. To correct this issue, restore contactability to the FTP Server or change the Relay Server configuration to specify a contactable FTP Server. Then retry the operation.
-812	An FTP Server name could not be resolved to an IP Address. This error occurs when the network name specified for an FTP Server in a Relay Server Objects cannot be resolved to an IP Address. This could occur if the network name was incorrect, if the network name was not mapped to an IP Address, or if name resolution is not supported. To correct this issue, change the Relay Server to specify a correct network name, ensure that the network name is mapped to an IP Address, add support for name resolution, or change the Relay Server to specify an IP Address instead of a network name. Then retry the operation.

Error Code	Description
-813	A file could not be successfully downloaded from an FTP Server to the device. This error could occur if a file to be downloaded from an FTP Server to the device is not present and hence cannot be downloaded. This error most commonly occurs due to a communications error during the download of a file. The configured number of retries will automatically be attempted to recover from this error. Retrying will frequently allow continuation following intermittent connectivity errors. If the file cannot be downloaded within the configured retries, the operation will fail. To correct this issue, retry the operation when connectivity is more reliable. If a Job is initiated by a Provisioning Policy or an Action and the Job is set to allow Pause/Resume, then instead of failing, the Job will be paused if the number of configured retries is exceeded without success. The Job will then be automatically resumed later when connectivity might be better and the Job can be successfully completed. This error could also occur if the Content on an FTP Server has been modified in some out-of-band manner. To correct this issue, restore the missing file to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-815	The available space on the device could not be determined. This error should not typically occur unless there is a serious problem with the Device File System.
-816	A path specified for a file to be stored on the device was invalid. This error occurs when the path in a Package that determines where a file is to be stored in the device does not define a valid location in the File System of the device. To correct this issue, change the Package to specify a valid destination path in the device. Then retry the operation.

Error Code	Description
-817	A folder in the device could not be created. This error occurs when an attempt to create a folder in the device failed. This could occur if some part of the path specified for a file in a Package that does not exist and could not be created. This would typically indicate a permissions issue on the device (e.g. creation of folders is not allowed within some folder). To correct this issue, correct the permissions issue on the device that is preventing the folder from being created or change the Package to specific a different path that can be successfully created. Then retry the operation.
-822	The temporary Package folder on the device does not exist. This error occurs when the temporary Package folder (\application\airbeam_pt) on the device does not exist. This folder is required to hold files temporarily when they cannot be immediately overwritten. If this folder does not exist, it will disrupt operation of the MSP Client Software. This error generally indicates a serious problem occurred during the installation of the MSP Client Software or that a change has been made to the MSP Client Software installation on the device in an out-of-band manner. To correct this issue, the missing folder must be created on the device. This can be done simply by rebooting the device since the MSP Agent will automatically re-create this folder when the device is rebooted, if it is found to be missing. Then retry the operation.
-823	The Package folder on the device does not exist. This error occurs when the Package folder (\application\airbeam\pkg) on the device does not exist. This folder is required to hold Package files that are stored on the device. If this folder does not exist, it will disrupt operation of the MSP Client Software. This error generally indicates a serious problem occurred during the installation of the MSP Client Software or that a change has been made to the MSP Client Software installation on the device in an out-of-band manner. To correct this issue, the missing folder must be created on the device. This can be done simply by rebooting the device since the MSP Agent will automatically re-create this folder when the device is rebooted, if it is found to be missing. Then retry the operation.
-829	This is a low-level error that indicates the MSP agent configuration could not be read from the registry. This error happens most frequently because of a registry corruption issue. The last, correct registry entries are kept in the Persistence folder in storage or RAM (depending on the device).

Error Code	Description
-830	This is a low-level error that indicates the MSP agent configuration could not be saved to \Application\Airbeam.reg. This error happens most frequently when there is corruption in the file system. Check to be sure there is enough disk space on the App folder.
-836	A file could not be successfully uploaded from the device to an FTP Server. This error occurs when a file, such as a Job Log, a Discovery Document, or a Collection Document, cannot be uploaded to an FTP Server. This error generally indicates that permissions on the FTP Server account or folder have not been set correctly to support MSP operation. To correct this issue, ensure that the permissions of the account or folder on the FTP Server are set correctly to support MSP operation or change the Relay Server to reference a different FTP Server. Then retry the operation.
-837	The directory listing of a folder on an FTP Server could not be obtained. This error occurs when an attempt to read the directory of a folder on an FTP Server fails. This could occur if the folder does not exist on the FTP Server or the permissions for the folder do not allow a directory listing to be produced. This error generally indicates that permissions on the FTP Server account or folder have not been set correctly to support MSP operation. To correct this issue, ensure that the permissions of the account or folder on the FTP Server are set correctly to support MSP operation or change the Relay Server to reference a different FTP Server. Then retry the operation.
-841	Execution of an Install Command Line failed. This error occurs when the command line specified as the Install Command of a Package resulted in an error. This can occur for a variety of reasons, such as a typo in the command line, a missing file, etc. that prevents a program from being successfully launched as required by the command line. It can also occur because a program run was successfully launched as required by the command line but the program returned a non-zero error code. To correct this issue, change the command line if it is incorrect, correct any missing files if necessary, change the program to return a zero instead of a non-zero, or change the command line to ignore the error returned by the program. Then retry the operation.

Error Code	Description
-842	Execution of an Uninstall Command Line failed. This error occurs when the command line specified as the Uninstall Command of a Package resulted in an error. This can occur for a variety of reasons, such as a typo in the command line, a missing file, etc. that prevents a program from being successfully launched as required by the command line. It can also occur because a program run was successfully launched as required by the command line but the program returned a non-zero error code. To correct this issue, change the command line if it is incorrect, correct any missing files if necessary, change the program to return a zero instead of a non-zero, or change the command line to ignore the error returned by the program. Then retry the operation.
-847	An AirBEAM Smart Client license is required. This error occurs when an attempt is made to load an unlicensed Package using an unlicensed AirBEAM Smart Client. The AirBEAM Smart licensing model requires that either the Package or the AirBEAM Smart Client be licensed. It is not permitted to load unlicensed Packages using an unlicensed AirBEAM Smart Client. To correct this issue, use a pre-licensed Package (e.g. one provided by Motorola), install an AirBEAM license file onto the device to cause the AirBEAM Smart Client to become licensed, or switch to using MSP which does not require AirBEAM Smart licensing.
-848	This error is basically the same as error -803 and should be handled the same.
-849	A file could not be successfully renamed on an FTP Server. This error occurs when a file, such as a Job file, cannot be renamed on an FTP Server. This error generally indicates that permissions on the FTP Server account or folder have not been set correctly to support MSP operation. To correct this issue, ensure that the permissions of the account or folder on the FTP Server are set correctly to support MSP operation or change the Relay Server to reference a different FTP Server. Then retry the operation.
-850	This is a low-level error that indicates that an FTP error occurred while the MSP agent was trying to restart the download of a file. This error happens most frequently when the FTP Server being used does not support Restart. This would generally indicate that the FTP Server cannot be used with MSP since it does not support the required commands and hence does not comply with the required RFCs.

Error Code	Description
-851	This is a low-level error that indicates a resource problem while trying to determine the inventory of installed packages on the device. This error happens most frequently when there is insufficient program memory space to create a temporary inventory list. Clear some program memory and retry.
-854	Processing is already in progress. This error occurs when an attempt is made to start some processing when that same processing is already being performed by another process. This could occur if the RD Client is used to start Settings Processing or Bundle Processing when the MSP Agent is already performing similar processing in the background. This could occur if the AirBEAM Smart Client is used to start Package Processing when the MSP Agent is already performing similar processing in the background. This could occur if the MSP Agent attempts to perform its scheduled execution in the background when the RD Client or AirBEAM Smart Client is performing similar processing in the foreground. If this error occurs when the MSP Agent attempts to perform its scheduled execution, then it is non-fatal and essentially self-correcting. The MSP Agent will try again when the time comes for its next scheduled execution. To correct this issue if it occurs when using the RD Client or AirBEAM Smart Client, try the requested operation again at a later time when the MSP Agent is not performing its scheduled execution.
-857	This is a low-level error that indicates one, or more of the required registry entries that contain file paths are missing. This error indicates an OS build issue. Check the OS build for errors.
-859	An end-user has attempted to invoke the AirBEAM client on a device that does not support AirBEAM.
-1001	A Job could not be completed and hence was marked as an orphan. This error occurs if the Install Command of a Package processed as part of a Job causes a reboot. In such a case, the Job state is lost, preventing the Job from being continued following the reboot. Consequently, the Job cannot be completed and is marked as an orphan. To prevent this issue, do not reboot the device from within a Package install command. Instead, use a Reboot Step in the Bundle to initiate the reboot. This will allow the MSP Agent to save the state of the Job and continue processing the Job following the reboot.

Error Code	Description
-1002	A Job was cancelled by MSP Server request. This error indicates that a Job was cancelled due to a request from the MSP Server. A Job can be cancelled by deactivating the Provisioning Policy that sent the Job or by cancelling the Action that sent the Job. This error does not indicate a failure but rather indicates that the requested cancellation has been successfully accomplished.
-1003	The MSP Agent log file could not be opened. This error should not typically occur unless there is a serious problem with the Device File System.
-1004	The MSP Agent log file could not be read. This error should not typically occur unless there is a serious problem with the Device File System.
-1007	A required Plug-In is improperly registered. This error occurs when an attempt was made to use a Plug-In that is improperly registered. This most commonly occurs if a faulty custom Plug-In is being used or if the registration of a Plug-In has been changed in an out-of-band manner. To correct this issue, obtain and install a corrected custom Plug-In or reinstall the Plug-In for which the registration has been damaged. Then retry the operation.
-1008	A value cannot be read from the Device Registry. This error should never occur unless there is a serious problem with the Device Registry.
-1009	A required Plug-In is not available. This error occurs when an attempt was made to use a Plug-In that is not registered on the device. This most commonly occurs when a Settings Object (e.g. Clock.DateAndTime) or Condition Object (e.g. AdapterTime) is deployed to a device when the required Package to support that Object has not been installed on the device. To correct this issue, deploy the required Package. Then retry the operation.

Error Code	Description
-1010	A required Bundle is not on an FTP Server. This error occurs when a Bundle referenced in a Staging Profile was not found on an FTP Server. This most commonly occurs when a Staging Profile is used before the Content required by the Staging Profile has been delivered. To correct this issue, wait until all the required Content has been delivered. Then retry the operation. This error should not occur as a result of a Provisioning Policy or Action because a Job is not sent until the referenced Bundle has been sent. This error could occur if the Content of the FTP Server has been altered in an out-of-band manner. To correct this issue, the missing Content must be restored to the Relay Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-1011	A connection to an FTPS Server could not be established. This error occurs when there is a trust issue with the Server Certificate of the FTP Server or when the VerifyServer option is enabled for a Relay Server and the FTPS Server IP Address or network name does not match the Server Certificate. To correct this issue, establish the necessary trust of the Server Certificate (e.g. by installing the issuing CA Certificate into the Root Certificate Store of the device) or change the configuration of the Relay Server so the Server IP Address or network name used to access the FTPS Server matches the Server Certificate of the FTPS Server. Then retry the operation.
-1012	A Bundle is invalid or not supported. This error occurs when a Bundle is invalid or uses a format that is not supported. This typically occurs when newer features, such as Dynamic Deployment, are used with an older version of the MSP Client Software. To correct this issue, update the version of the MSP Client Software to a version that supports the required features or limit the features used to those supported by the installed version of the MSP Client Software. Then retry the operation.
-1013	A Job is invalid or not supported. This error occurs when a Job is invalid or uses a format that is not supported. This typically occurs when newer features, such as Dynamic Deployment, are used with an older version of the MSP Client Software. To correct this issue, update the version of the MSP Client Software to a version that supports the required features or limit the features used to those supported by the installed version of the MSP Client Software. Then retry the operation.

Error Code	Description
-1014	A file could not be deleted from an FTP Server. This error occurs when a file could not be deleted from an FTP Server. This error typically occurs when the permissions on the FTP Server account or folder have not been set correctly to support MSP operation. To correct this issue, ensure that the permissions of the account or folder on the FTP Server are set correctly to support MSP operation or change the Relay Server to reference a different FTP Server. Then retry the operation.
-1015	A Bundle file could not be created. This error occurs during Legacy Staging when a Legacy Staging Profile needs to be converted into a Bundle and the Bundle file cannot be created. This error typically indicates that inadequate space is available in the File System of the device. To correct this issue free up some space in the File System of the device. Then retry the operation.
-1016	A Bundle file could not be written. This error occurs during Legacy Staging when a Legacy Staging Profile needs to be converted into a Bundle and the Bundle file cannot be written. This error typically indicates that inadequate space is available in the File System of the device. To correct this issue free up some space in the File System of the device. Then retry the operation.
-1017	Invalid Message Set format. This error occurs when the Message Set information in a Bundle downloaded to the device had an invalid format. This should typically not occur unless the Bundle file on the FTP Server has been altered in an out-of-band manner. To correct this issue, restore the corrupted Bundle file to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.

Error Code	Description
-1018	Invalid BLOB format. This error occurs when the header information in any BLOB downloaded to the device had an invalid format. This should typically not occur unless a file on the FTP Server has been altered in an out-of-band manner. To correct this issue, restore the corrupted file to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-1019	A valid UUID could not be obtained for the device. This error occurs when the attempt to obtain the UUID of the device fails. Since a device cannot be managed by MSP without a valid UUID, this is a fatal error. This error typically results from a faulty device that does not properly implement the required APIs to return the UUID and hence is not compliant with MSP. To correct this issue it is generally necessary load new firmware into the device to make it MSP compliant. Then retry the operation.
-1022	A Job was cancelled by MSP Agent shutdown. This error indicates that a Job was cancelled because it was in progress when the MSP Agent on the device was shut down. The MSP Agent could be shut down by explicit request from the Device User via the MSP Agent UI or via the use of the MSP Agent Programmatic Interface. To avoid cancellation of Jobs by the Device User, it may be desirable to disable access to the MSP Agent UI or to just the Exit function of the MSP Agent UI. The MSP Agent Programmatic Interface should be used with caution to avoid cancelling Jobs unnecessarily.
-1023	A value cannot be written into the Device Registry. This error should never occur unless there is a serious problem with the Device Registry.

Error Code	Description
-1024	A Condition Plug-In returned an error. This error occurs when a Condition Object is being evaluated on the device and the associated Condition Plug-In returned an error. This usually occurs because the data passed to the Plug-In (in the BLOB generated based on the data entered for the Condition Object) is invalid or if a faulty custom Condition Plug-In is being used. To correct this issue, change the Condition Object to correct the invalid data or obtain and install a corrected custom Condition Plug-In. Then retry the operation.
-1025	The current working directory could not be changed on an FTP Server. This error occurs when an attempt to change the current working directory on an FTP Server fails. This would typically occur because of a permission problem on the FTP Server or because a change was made to remove some folder from the FTP Server in an out-of-band manner. To correct this issue, adjust the permissions or recreate the missing required folder on the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-1026	A Relay Server Information File could not be processed. This error indicates that the Relay Server Information File downloaded from an FTP Server was invalid and could not be successfully processed. This error should only occur if changes are made to the Relay Server Information File in an out-of-band manner. To correct this issue, the correct Relay Server Information File must be restored to the FTP Server. Then retry the operation. The MSP Relay Server Optimization feature, if enabled, may assist with the correction of this issue. For more information on Relay Server Optimization, see Using MSP 3.3.1 – Administrative Setup.
-1028	A required Plug-In is missing a required entry point. This error occurs when an attempt was made to use a Plug-In that is registered but where the registered DLL is missing a required entry point. This most commonly occurs if a faulty custom Plug-In is being used. To correct this issue, obtain and install a corrected custom Plug-In. Then retry the operation.

Error Code	Description
-1029	A required Plug-In DLL is missing. This error occurs when an attempt was made to use a Plug-In that is registered but where the registered DLL is missing. This most commonly occurs if a faulty custom Plug-In is being used or if the DLL registered for the Plug-In has been removed in an out-of-band manner. To correct this issue, obtain and install a corrected custom Plug-In or reinstall the Plug-In for which the registered DLL has been deleted. Then retry the operation.
-1030	A Settings Plug-In returned an error. This error occurs when a Settings Object is being applied on the device and the associated Settings Plug-In returned an error. This usually occurs because the data passed to the Plug-In (in the BLOB generated based on the data entered for the Settings Object) is invalid or if a faulty custom Settings Plug-In is being used. To correct this issue, change the Settings Object to correct the invalid data or obtain and install a corrected custom Settings Plug-In. Then retry the operation.
-1031	A Job could not be started because it would exceed “Max Jobs”. This error occurs when starting a Job would exceed the limit on the number of concurrent Jobs allowed to be executed at once for a Relay Server. This error is non-fatal and will not cause the Job to fail. The Job will remain pending and will be started automatically at some later time when it would not cause the limit to be exceeded.
-1034	An invalid type of reboot was requested. This error occurs when reboot was requested and the device cannot support that type of reboot. For example, a Clean Boot might have been requested on a device that cannot support a Clean Boot. To correct this issue, request a type of reboot that is supported by the device. Then retry the operation.

Chapter 3 - Windows PC Client

Windows Device Client Compatibility

Windows XP

Prior to MSP 3.3.1, only Windows Devices were supported. Beginning with MSP 3.3.1, Windows PCs are supported as well, via the Windows PC Client. The Windows PC Client is designed to support Windows PCs that are running the following Microsoft Windows Operating Systems:

- Windows XP 32 Bit with Service Pack 3 (SP3)
- Windows XP 64 Bit with Service Pack 2 (SP2)

Windows PC Client Software Components

The Windows PC Client is comprised of the software components described in the following subsections:

Windows PC Client Rapid Deployment Client. Due to the nature of Windows PCs, however, certain features may not work or may not make sense on a Windows PC.

Where common functionality is provided by both RD Clients, it will not be described again in this chapter. Where common interfaces are supported by both RD Clients, it will not be described again in this chapter. This chapter will focus solely on the differences between the two.

Windows PC Client RD Client Executable(s)

The Windows PC Client RD Client consists primarily of the executable file “rdclient.exe”. The RD Client is always installed into the folder “C:\MspAgent” on the Windows PC.

Windows PC Client RD Client User Interface

While the Windows PC Client RD Client implements the same User Interface (UI) as the Windows Device Client RD Client, that UI was designed to be used on small device screens rather than on large PC screens. Since it does not provide an efficient UI solution on Windows PCs, it has not been subjected to the extensive testing and certification that was performed for the Windows Device Client RD Client UI.. The Programmatic Interface isthe **only** recommended and formally supported use of the RD Client on the Windows PC.

Windows PC Client RD Client Registry Interface

While the Windows PC Client RD Client implements the same Registry Interface as the Windows Device Client RD Client, it has not been subjected to the extensive testing and certification that was performed for the Windows Device Client RD Client Registry Interface.. Nonetheless, it is neither recommended nor formally supported for use on Windows PCs.

Windows PC Client MSP Agent

The Windows PC Client MSP Agent provides agent functionality for Windows PCs that are Directly Managed by MSP. This includes Provisioning functionality, when used with MSP Provision Edition or MSP Control Edition, and Data Collection functionality, when used with MSP Control Edition.

Windows PC Client MSP Agent Executable(s)

The Windows PC Client MSP Agent consists primarily of the executable file “MPAService.exe”. The MSP Agent is always installed into the folder “C:\MspAgent” on the Windows PC.

Windows PC Client MSP Agent User Interface

The Windows PC Client MSP Agent on the Windows PC has no available User Interface.

Windows PC Client MSP Agent Programmatic Interface

There are times when it is desirable for a PC User, application, or other process to interact with Windows PC Client MSP Agent programmatically to perform some function. This is done by launching the Windows PC Client Service executable file “C:\MspAgent\ MPAService.exe”, typically using the CreateProcess() API call, or equivalent.

When invoking the Windows PC Client MSP Agent programmatically, various command line arguments can be provided to achieve different results, as described in Table 8 below.

Table 8 – Windows PC Client Service Command Line Arguments

Argument	Description	Example
`-checkin`	Command the Windows PC Client Service of the Windows PC Client MSP Agent to perform a single execution cycle as soon as possible.	C:\MspAgent\MPAService.exe –checkin

Argument	Description	Example
`–install`	Install the Windows PC Client Service of the Windows PC Client MSP Agent. This is to set the Windows PC Client Service so it starts automatically when the Windows PC is booted.	C:\MspAgent\MPAService.exe –install
`–remove`	Remove the Windows PC Client Service of the Windows PC Client MSP Agent. This is used to reset the Windows PC Client Service so it does not start automatically when the Windows PC is booted.	C:\MspAgent\MPAService.exe –remove
`–start`	Start the Windows PC Client Service of the Windows PC Client MSP Agent. This is used typically used during installation of the Windows PC Client. It can also be used to restart the Windows PC Client Service if it has been stopped for some reason (e.g. during maintenance, etc.).	C:\MspAgent\MPAService.exe –start
`–stop`	Stop the Windows PC Client Service of the Windows PC Client MSP Agent. This is used typically used before uninstalling the Windows PC Client. It can also be used to stop the Windows PC Client Service if it becomes necessary for some reason (e.g. to perform maintenance, etc.).	C:\MspAgent\MPAService.exe –stop

Windows PC Client MSP Agent Registry Interface

The Windows PC Client MSP Agent on the Windows PC has the same Registry Interface as described for the Windows Device Client MSP Agent in .

Windows PC Client Availability

Windows PC Client Initial Installation

Pre-requisites

A Windows PC must meet the following pre-requisites to support the use of the Windows PC Client:

- The Windows PC must be running one of the Operating Systems listed in the first bullet under the heading Windows XP on Page 63
- The User performing the installation of the Windows PC Client must have the following rights on the Windows PC:
  - o “Read/write/create/delete files” rights in the folder “C:\MspAgent”.
  - o “Read/write/create/delete registry entries” rights to the Registry Path “HKEY_LOCAL_MACHINE\ Software\AirBEAM”.
  - o “Read/write/create/delete registry entries” rights to the Registry Path “HKEY_LOCAL_MACHINE\Software\MSP”.
  - o “Log on as a service” right.

    Note:

    This right can be set by launching the Local Policies control applet as shown below.

    1. From the Start Menu on the Windows PC, launch **Control Panel > Administrative Tools > Local Security Policy**.
    2. Select **Local Policies**
    3. Select **User Rights Assignment**
    4. Select **Log on as service**

    This right can also be set by using the Microsoft NTRights.exe utility using a command line of the following format:

    ntrights –u <username> +r SeServiceLogonRight
    e.g. ntrights –u administrator +r SeServiceLogonRight

- The User performing the installation of the Windows PC Client can be a dedicated user or any available User, so long as the rights listed above are available to that User.
- The Windows PC on which the Windows PC Client is installed must have TCP/IP connectivity to an MSP Relay Server through which it will communicate with the MSP Server.

- Since the Windows PC Client installs unconditionally to the folder “C:\MspAgent folder” on the Windows PC, sufficient hard disk must be available on the C: drive.

Note:

The available disk space required on the C: drive of the Windows PC will vary depending on what the Windows PC Client will be called upon to do on a given Windows PC. If any Control Modules or Proxy Plug-Ins will be loaded, then additional space would be required.

Manual Initial Installation Process

Windows PCs generally will not ship with the Windows PC Client pre-installed. Consequently, the Windows PC Client must be installed on a Windows PC before it can be Directly Managed by MSP.

Important:

Windows PCs do not provide a universal standard method to obtain a unique hardware identifier value. Since MSP requires that every managed device have a Unique Unit Identifier (UUID), the Windows PC Client generates a software-based UUID for the Windows PC when it is first installed. The generated UUID is stored in the Registry of the Windows PC and is guaranteed to be unique and never duplicated by any Windows PC on which the Windows PC Client may be installed.

The generated UUID is **not** removed from the Registry of the Windows PC when the Windows PC Client is uninstalled using the process defined later in this document. If the Windows PC Client is later installed again onto the same Windows PC, the Windows PC Client will **not** generate a new UUID and hence the Windows PC will be treated by MSP as being the same managed device.

If a Windows PC was completely re-initialized or if the generated UUID stored in the Registry of the Windows PC was otherwise removed, and then the Windows PC Client was installed onto the Windows PC, then the Windows PC Client would generate a new UUID and hence the Windows PC will be treated by MSP as being a **different** managed device. In such a case, the old managed device should be deleted via the MSP Console to avoid tying up a license for the old UUID.

The Initial Installation Process could be performed manually by a User logged into the Windows PC or could be automated. Automation might be performed in a variety of ways, including:

- Through the local or remote execution of scripts or batch files.
- Through the use of some other Windows PC management system (e.g. Microsoft SMS< SCCM, etc.).

Note:

While automation of the Initial Installation Process is possible, it is beyond the scope of this document to define exact methods to accomplish it using any specific method or tool.

The Initial Installation Process for the Windows PC Client is accomplished by performing the following steps:

1. Select a Windows PC that meets the pre-requisites defined for the Windows PC Client as defined earlier in this chapter.

2. Extract the file “MspAgentInstall.msi” from the Add-On Kit .ZIP file and place it into an appropriate temporary location on the Windows PC.

3. Log into the MSP Console UI and export a .BLOB file containing the Relay Server Object for the Server via which the Windows PC Client will communicate to the MSP Server. Place the .BLOB file into the same temporary location on the Windows PC.

4. Launch a Command Window on the Windows PC and change the working directory of the Command Window to the temporary folder into which the above files were placed.

5. Install the Windows PC Client software onto the Windows PC using a command of the form:

   ```
   Msiexec.exe /quiet /i <path>\MspAgentInstall.msi
   ```

   where: ***<path>*** represents the path to the folder on the Windows PC where the file was placed.

6. Configure the Windows PC Client Service on the Windows PC to automatically start on subsequent reboots of the Windows PC by using a command of the form:

   ```
   C:\MspAgent\MPAService.exe –install <user> <password>
   ```

   where: ***<user>*** represents the user the service will login as. The user must be specified in domain format (e.g. \domain\user) if the user is part of a domain or must be specified in non-domain format (e.g. \user) if the user is not part of a domain.

   <password> represents the password that will be used to authentic the specified user when the service logs on as that user.

   Note:

   If you are using the Windows PC Client with certain Proxy Plug-Ins, it may be necessary to run the Windows PC Client Service under the Local System account. This can be accomplished by leaving the <user> and <password> blank in the above command. Consult the documentation for the Proxy Plug-In you intend to use to determine if this is a requirement.

7. [Optional] Determine and record the UUID assigned to the Windows PC by using a command line of the form:

   ```
   C:\MspAgent\rdclient.exe
   ```

   When the RD Client UI appears, select **Options > View Client Info** and record the UUID displayed.

8. Configure the Windows PC Client to access the appropriate Relay Server by using a command of the form:

   ```
   C:\MspAgent\rdclient.exe -I<path>\<file.blob>
   ```

   where: ***<path>*** represents the path to the folder on the Windows PC where the .BLOB file was placed.

   <file.blob> represents the file name of the .BLOB file into which the appropriate Relay Server Object was exported.

9. Start the Windows PC Client Service executing on the Windows PC by using a command of the form:

   ```
   C:\MspAgent\MPAService.exe –start <user> <password>
   ```

   where: ***<user>*** represents the user the service will login as. The user must be specified in domain format (e.g. \domain\user) if the user is part of a domain or must be specified in non-domain format (e.g. \user) if the user is not part of a domain.

   <password> represents the password that will be used to authentic the specified user when the service logs on as that user.

10. From the Start Menu on the Windows PC, launch **Control Panel > Administrative Tools > Services** and verify that the Windows PC Client Service is running. The actual service name that should shown is "MSP Proxy Agent Service".

11. [Optional] Verify that the Windows PC shows up properly in MSP:

    a. Log into the MSP Console UI and invoke the **Device Search** function.

    b. Select Device ID as the Device Attribute to search on.

    c. Enter the UUID recorded in Optional step 7 above.

    d. Verify that exactly one device is found that matches the entered UUID.

    e. Click the link for the displayed device to get to its **Device Detail** page.

    f. Verify the information shown for the Windows PC, especially that the Relay Server is the one via which the Windows PC was just Staged.

Automation of the Initial Installation Process

As previously mentioned, it is possible to automate the Initial Installation of the Windows PC Client on a Windows PC, but it is beyond the scope of this document to describe an exact process for doing so via any specific mechanism or tool. Nonetheless, this section will provide some high-level guidelines that would likely need to be followed as part of any automation of the process.

- Identify one or more Windows PCs on which the Windows PC Client will be installed and which can be remotely managed in some manner. For each such Windows PC, the following steps would generally need to be done:

  1. Extract the necessary files from the Add-On Kit .ZIP file, as described in the Manual Process and deliver them to a temporary folder on the Windows PC.

  2. Obtain a .BLOB file, as described in the Manual Process, containing the Relay Server Object for the Relay Server via which the Windows PC will communicate to the MSP Server and deliver it to a temporary folder on the Windows PC.

  3. Execute the command to install the Windows PC Client software onto the Windows PC, as described in the Manual Process.

  4. Execute the command to configure the Windows PC Client Service on the Windows PC to automatically start on subsequent reboots, as described in the Manual Process.

  5. Execute the command to configure the Windows PC Client to access the appropriate Relay Server, as described in the Manual Process.

6. Execute the command to start the Windows PC Client Service executing on the Windows PC, as described in the Manual Process.
7. Log into the MSP Console UI and verify that the Windows PC shows up properly in MSP.

- One or more of the above steps might be accomplished for multiple Windows PCs at once. For example:
    - Multiple .BLOB files for multiple Relay Server Objects might be exported from the MSP Console UI in a single operation. The right .BLOB file would need to be identified and delivered to each Windows PC.
    - The UUID associated to a Windows PC can be extracted from the Registry on that Windows PC from "HKEY_LOCAL_MACHINE\Software\MSP" in the value name "UUID".
    - Verification that a Windows PC shows up as managed device might be done for multiple Windows PCs at once.

Windows PC Client Update Packages

The Windows PC Client on a given Windows PC could have been installed via the Windows PC Client Initial Installation Process or could have been previously upgraded via a Windows PC Client Update Package. Whatever method was used to make the Windows PC Client resident on a Windows PC, it should **only** be updated (e.g. to fix bugs and/or add new features) by deploying a Windows PC Client Update Package to the Windows PC.

Important:

A Windows PC Client Update Package is the **only** supported mechanism to update the Windows PC Client on a Windows that already has the Windows PC Client resident.

An .MSI File is **never** a supported mechanism to update the Windows PC Client on a device that already has the Windows PC Client resident.

The Windows PC Client that is resident on a device includes the Rapid Deployment Client. The Rapid Deployment Client can be used to perform MSP Staging to deploy a Windows PC Client Update Package to the device.

The Windows PC Client that is resident on a device includes the MSP Agent. If scheduled execution has been configured, then the MSP Agent can be updated from MSP by deploying a Windows PC Client Update Package via an MSP Provisioning Policy or an MSP Action.

Obtaining Windows PC Client Updates

Windows PC Client Update Packages are provided as part of the MSP Server and via Windows PC Client Add-On Kits that are separately available for download from the following link:

http://support.symbol.com/support/product/softwaredownloads.do

Add-On Kit Contents

- A Windows PC Client Add-On-Kit will be a .ZIP File with a name of the form:
    - MSP_PC_Client.zip_<version>.ZIP

where:

<version>	indicates the version number of the software delivered in the Add-On Kit.
<date>	indicates the release date of the Add-On Kit .ZIP File (represented in YYYYDDMM format).

- Each Windows PC Client Add-On-Kit Add-On Kit .ZIP File contains the following contents:
  - MspAgentInstall_<version>.msi
  - MPAServiceUpdate_<version>.apf
  - MSP 3.0 Client Release Notes.txt

Add-On Kit Installation

For Users with direct access to Windows Console on the MSP Server, an Add-On Kit .ZIP Files can be installed using the MSP Administration Program. See Add-On Kit Installation in Administering MSP 3.3.1.

Windows PC Client Uninstallation

In some situations, it may be desirable to permanently remove the Windows PC Client from a Windows PC. This might make sense, for example, if the Windows PC is being re-purposed and there is no longer a need for that Windows PC to be managed by MSP.

While the Windows PC Client can be updated using a Windows PC Client Update Package, and while a Windows PC Client Update Package can be uninstalled from the Windows PC, the uninstallation of such a Package will **not** remove the Windows PC Client from the Windows PC. The only way to remove the Windows PC Client from a Windows PC is to following the Uninstallation Process defined below.

Important:

Once the Windows PC Client has been uninstalled from a Windows PC by following the Uninstallation Process, the Initial Installation Process would need to be repeated in its entirety if it ever became necessary to install the Windows PC Client again onto that Windows PC.

To completely uninstall the Windows PC Client from a Windows PC, perform the following steps:

1. Launch a Command Window on the Windows PC and change the working directory of the Command Window to the temporary folder into which the above files were placed.
2. Stop the Windows PC Client Service from executing on the Windows PC by using a command of the form:

   ```
   C:\MspAgent\MPAService.exe –stop
   ```

3. Configure the Windows PC Client Service on the Windows PC to no longer automatically start on subsequent reboots of the Windows PC by using a command of the form:

   ```
   C:\MspAgent\MPAService.exe –remove
   ```

4. Uninstall the Windows PC Client software from the Windows PC using the following command:

```
Msiexec.exe /quiet /x {462147C5-B13F-43AF-A02A-E9DFE2A23905}
```

Note:

The GUID listed above (“**`{462147C5-B13F-43AF-A02A-E9DFE2A23905}`**”) identifies the Windows PC Client and should not be changed. The command line should be used exactly as presented.

Windows PC Client Proxy Plug-In Support

In addition to enabling the Windows PC on which it is running to be Directly Managed by MSP, the Windows PC Client can host Proxy Plug-Ins which can enable the Windows PC to act as a Proxy for other devices, thus allowing them to be Indirectly Managed by MSP.

Note:

For more information on Directly Managed and Indirectly Managed devices, refer to Understanding MSP 3.3.1 – MSP and Your Enterprise.

Once the Windows PC Client is installed on a Windows PC and the Windows PC is shown in the MSP Console UI as a managed device, the Windows PC is ready to host one or more Proxy Plug-Ins.

Proxy Plug-In Package Installation

Proxy Plug-Ins are delivered as Packages and are deployed to a Windows PC just like any other Package. When the Package for a Proxy Plug-In is installed by the Windows PC Client, it begins its task of enabling devices of one or more Device Classes to be Indirectly Managed by MSP.

Note:

For more information on Proxies, refer to Understanding MSP 3.3.1 – Understanding Device Classes and Proxies.

Proxy Plug-In Package Uninstallation

When the Packages for one or more Proxy Plug-Ins are installed onto a given Windows PC, they can be uninstalled at any time just like any other Packages. When the Package for a given Proxy Plug-In is uninstalled from a given Windows PC, then that Windows PC will cease to act as a Proxy for devices of the Device Classes supported by that Proxy Plug-In.

Proxy Plug-In Package Updates

When the Packages for one or more Proxy Plug-Ins are installed onto a given Windows PC, they can be updated at any time just like any other Packages. If a version of the Package for a given Proxy Plug-In is deployed to a Windows PC that already has a different version of the Package for that Proxy Plug-In, then the old Package will be uninstalled and the new Package will be installed. This means that the Windows PC will temporarily cease to act as a Proxy for devices of the Device Classes supported by that Proxy Plug-In for duration of the update.

Proxy Plug-Ins and Windows PC Client Updates

When the Packages for one or more Proxy Plug-Ins are installed onto a given Windows PC, the

Windows PC Client can be updated at any time. All Proxy Plug-Ins that were installed before the Windows PC Client is updated will still be installed afterwards. The Windows PC will temporarily cease to act as a Proxy for devices of the Device Classes supported by all Proxy Plug-Ins for duration of the update.

Windows PC Client Management Functionality

This section and the following sub-sections will provide a high-level overview of the MSP Management Functionality provided when a Windows PC is Directly Managed via the Windows PC Client.

Note:

The intent of this document is to provide an understanding of the capabilities of the MSP Client Software. This section will thus provide a high-level explanation of the functions supported by the Windows PC Client, but will not cover the exact features and functions supported on any given Windows PC, Operating System version, etc. For more information on specific Windows PCs or Operating Systems supported by MSP, see the MSP 3.3.1 Release Notes.

Staging

Since the RD Client is one of the available components of the Windows PC Client, Staging is available for use on Windows PCs when used with MSP Stage Edition, MSP Provision Edition, or MSP Control Edition. Staging is not, however, considered a key function provided by the Windows PC Client.

Staging would typically only be used on Windows PCs to perform the initial configuration of the MSP Agent required to make the Windows PC Directly Manageable by MSP.

Asset Management

Since the MSP Agent is one of the available components of the Windows PC Client, Asset Management is available for use on Windows PCs when used with MSP Provision Edition or MSP Control Edition. Asset Management is considered a key function provided by the Windows PC Client.

Asset Management includes discovery of Windows PCs by MSP, the acquisition and reporting to MSP of values for a variety of Device Attributes for Windows PCs, and the tracking of the inventories of Packages deployed on Windows PCs.

The key Device Attributes supported by the Windows PC Client are listed in Table 9 below:

Table 9 – Window PC Client Key Device Attributes

Device Attribute Name	Explanation
Agent.Feature	The value of this Device Attribute indicates the level of functionality being used by the MSP Agent and hence, the type of license required to manage the device within MSP. Possible values are: Provision or Control.
Agent.SupportsCheckInConditions	The value of this Device Attribute indicates whether the MSP Agent supports the use of Check-In Conditions to limit when it can contact the Relay Server. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsCheckInPackages	The value of this Device Attribute indicates whether the MSP Agent supports the use of Check-In Commands in Packages to allow Packages to perform periodic activity. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsDisableOptions	The value of this Device Attribute indicates whether the MSP Agent supports to configure whether various MSP Agent UI Options will be accessible. Possible values are: 1 = Yes and 0 = No.
Agent.SupportsLimitJobs	The value of this Device Attribute indicates whether the MSP Agent supports the ability to honor the limit on the number of Jobs that can be simultaneously executed on a given Relay Server. Possible values are: 1 = Yes and 0 = No.
Identity.ClientVersion	The value of this Device Attribute indicates the version of the MSP Client Software that is currently installed.
Identity.UUID	The value of this Device Attribute indicates the UUID value that identifies the Windows PC to MSP as a unique managed device.
Identity.DnsDomain	The value of this Device Attribute indicates the DNS domain, if any, to which the managed device is assigned on the network.
Identity.DnsName	The value of this Device Attribute indicates the DNS name, if any, assigned to the managed device on the network.

Provisioning

Since the MSP Agent is one of the available components of the Windows PC Client, Provisioning is available for use on Windows PCs when used with MSP Provision Edition or MSP Control Edition. Provisioning is considered a key function provided by the Windows PC Client.

Provisioning provides the ability to define Policies that automatically or manually control the installation and uninstallation of Packages to Windows PCs.

Remote Control and Troubleshooting

As of MSP 3.3.1, Remote Control and Troubleshooting is **not yet available** for use on Windows PCs.

Actions

Since the MSP Agent is one of the available components of the Windows PC Client, Actions are available for use on Windows PCs when used with MSP Control Edition. Actions are considered a key function provided by the Windows PC Client.

Actions provide the ability to explicitly request the installation and uninstallation of Packages to Windows PCs.

Data Collection

Since the MSP Agent is one of the available components of the Windows PC Client, Data Collection is available for use on Windows PCs when used with MSP Control Edition. Data Collection is considered a key function provided by the Windows PC Client.

Data Collection provides the ability to deploy a Data Collection Solution that enables the periodic acquisition and reporting to MSP of samples of selected Collection Metrics from Windows PCs.

Real-Time Management

Since the MSP Agent is one of the available components of the Windows PC Client, Real-Time Management is available for use on Windows PCs when used with MSP Control Edition. Real-Time Management is considered a key function provided by the Windows PC Client.

Real-Time Management provides the ability to deploy a Real-Time Management Solution to devices that enables a Workstation PC to perform operations on one or more Windows PCs in real-time via direct connections to those Windows PCs.

Windows PC Client Troubleshooting

Windows PC Client MSP Agent Log File

The MSP Agent maintains a record during its normal processing of the activities it performs and the results of those activities. The MSP Agent Log File can only be viewed by accessing the file on the Windows PC, it is never sent to the MSP Server, and hence is never visible from, the MSP Console UI.

The Log Level can be configured as part of the MSP Agent configuration as described previously in this chapter. The Log Level controls the amount of information recorded into the MSP Agent Log File.

The MSP Agent Log File is stored by the MSP Agent into the folder “C:\MspAgent”. Within that folder, the file “Agent.log” contains the currently active (most recently recorded) log information and the file “Agent.log.old” contains the archive (least recently recorded) log information.

The MSP Agent Log File is a text file and can be viewed on the Windows PC using any suitable text editor (e.g. Notepad, Wordpad, etc.).

The Log Size can be configured as part of the MSP Agent configuration as described previously in this chapter. The Log Size controls the maximum size the two files can be reach. When the file “Agent.log” reaches the specified size, the file “Agent.log.old” is deleted, if it exists, the file “Agent.log” is renamed to “Agent.log.old, a new empty file “Agent.log” is created, and new recording of log information is added to the new file. This means that the combined size of the two files will never total to more than twice the specified Log Size.

Windows PC Client MSP Agent Job Log File

The MSP Agent maintains a record during the processing for each Job of the activities it performs and the results of those activities. The Job log information for each Job is uploaded to the MSP Server, via the Relay Server, when a Job is completed, and hence can be displayed from the MSP Console UI for any completed Job.

The Job Log Level can be configured as part of the MSP Agent configuration as described previously in this chapter. The Job Log Level controls the amount of information recorded into the MSP Agent Job Log File. The MSP Agent Job Log File is stored by the MSP Agent into the folder “C:\MspAgent”. Within that folder, the file “Job.log” contains the log information for the most recently executed Job.

The MSP Agent Job Log File is a text file and can be viewed on the Windows PC using any suitable text editor (e.g. Notepad, Wordpad, etc.).

Windows Event Log

Errors during installation, starting, stopping, or removal of the Windows PC Client Service will be recorded into the Windows Event Log of the Windows PC. To view this log, launch the Windows Event Viewer on the Windows PC (click **Start > Administrative Tools > Event Viewer** and select the Application item).

Note:

Many different applications and services that are running on a Windows PC may record information into the Windows Event Log. Consequently, not all messages seen in the Windows Event Log will necessarily be related to the Windows PC Client Service.

Windows PC Client Licensing

The licensing required to use the Windows PC Client for all supported devices is included in the per-device license required to manage a Windows PC using a given MSP Edition (e.g. Stage, Provision, or Control). No additional Windows PC Client licensing is required.

If a Windows PC is used with a Proxy Plug-In to provide the ability for additional devices to be Indirectly Managed by MSP, then each additional Indirectly Managed device will also require a suitable license.

Windows PC Client Best Practices

- Check Device Class in Applicability Rules

  When defining Applicability Rules in Provisioning Policies, Actions, and Data Collection Requests, include a check for the Device Attribute “Identity.DeviceClass” being equal to “Windows PC” whenever it is important that Applicability be limited to Windows PCs.

- Use an On-Going Provisioning Policy to ensure that the latest Windows PC Client is continuously kept up to date on all devices.

  Over time, the Windows PC Client will be updated to add new features. While EVERY version of the Windows PC Client is capable of operating, at least to some degree, with every version of MSP (from MSP 3.3.1 onwards), not all functionality advertised for a given version of MSP may be available if an older Windows PC Client version is present on the Windows PC.

  Luckily, every version of Windows PC Client is capable of updating itself to every newer version under the control of every version of MSP (from 3.3.1 onwards). As a result, any Windows PC with any version of Windows Device Client on it can be updated using a suitable Provisioning Policy.

- Don’t expect MSP to be the “be all and end all” of management systems for a Windows PC.

  MSP and the Windows PC Client were collectively designed and intended to provide a comparable level of management functionality for Windows PCs to what is offered for Windows Devices. It should be understood, however, that the Windows XP Operating System is must more sophisticated than any version of Windows CE Operating System or Windows Mobile Operating System.

  As a result, mechanisms provided by MSP that are perfectly adequate for managing Windows Devices are likely to fall short of providing a complete management solution for Windows PCs, especially generic desktop and laptop PCs. MSP is not intended to replace or compete with existing Enterprise Windows PC management systems.

  When considering MSP as a management system for Windows PCs, it should be seen as providing a “common denominator” of management functionality across a variety of managed devices, including Windows Devices and Windows PC. Within a context where similar management functions need to be performed across disparate populations of managed devices, MSP can offer a consistency of experience that may be very attractive.

- Leverage Pre-built Installation Packages.

  In many cases, content to be deployed to a Windows PC may already be available as a pre-built installation packages. For example, a third-party application may be provided as a .MSI file or an OS Patch may be provided as a self-installing .EXE file.

  Such installation packages can easily be included into MSP Package(s) for deployment to one or more Windows PCs that are being managed by MSP. The install command of an MSP Package can simply execute the installation package(s) to cause them to be installed on the Windows PC.

- Leverage Existing Windows PC Installation Tools

  Many tools exist for building installation packages for Windows PCs. Using such tools can greatly increase the ease and effectiveness of deploying content to Windows PCs using MSP. Once such an installation has been built, it can be handled as described about for pre-built installation packages.

- Use User Interfaces with Discretion

  Unlike a Windows Device, which is generally considered to have a Device User, a Windows PC may be seldom, if ever, attended by a Windows PC User. Be cautious about using any management functionality that requires interaction with a Windows PC User for successful operation.

Chapter 4 - Device Reboots, Shutdowns, and Persistence

Reboots and Shutdowns

Depending on the type of a device and the type and version of Operating System running on a device, a variety of different ways may exist to reboot and/or shutdown the device.

The terminology used to refer to a particular type of reboot or shutdown may vary depending on the device and/or Operating System, even if the basic operation of a reboot or shutdown is basically the same. To establish some consistency for the purposes of discussion, the terms most commonly used for Motorola Windows Mobile devices will be used. Where those terms are not commonly used for other devices, an explanation will be provided for how that type of reboot or shutdown does or does not apply.

The term reboot is generally used to refer to any operation that causes the Operating System on the device to be restarted. A reboot might be performed to recover from of a problem, such as a locked-up application, or might be performed to apply some configuration change that will take effect only when the Operating System starts.

The term shutdown is generally used to refer any operation that stop a device from operating for some duration. A shutdown might be done to save power, to permit the safe swapping of batteries or peripherals, or to prepare a device for transportation, long-term storage, servicing, etc.

Suspend and Resume

A Suspend is a type of shutdown that temporarily stops the operation of a device by transitioning it into a Suspend state. Suspend is typically performed to save power when the device will not be used for some period of time. A Resume is the reverse of a Suspend and restores the operation of the device to the state it was in just before it was Suspended.

Key Requirements

The key requirements for Suspend and Resume are:

- The device can relatively quickly enter the Suspend state by doing a Suspend.
- The device can be relatively quickly restored to operational status from the Suspend state by doing a Resume.
- The contents of RAM will be preserved across a Suspend followed by a Resume.
- The contents of any RAM Disk may or may not be lost as a result of a Warm Boot.
- The contents of all File Systems will be preserved across a Suspend followed by a Resume.
- The Device Registry will be preserved across a Suspend followed by a Resume.
- The state of the Operating System and running applications will be preserved across a Suspend followed by a Resume.
- The date and time maintained by the Operating System will automatically be updated to properly reflect any time elapsed between the Suspend and Resume.

Common Usage

Suspend is generally used when it is important to preserve the entire operational state of the device.

Suspend might be initiated manually by a Device User when the device is not expected to be used for a significant duration. Whether or not manual initiation of Suspend is supported may vary by device.

Suspend might be initiated programmatically on a device based on expected non-use, detection of inactivity, battery level, etc. Whether or not programmatic initiation of Suspend is supported may vary by device.

Since the Operating System may not be running between and Suspend and Resume, many devices rely on a Real Time Clock circuit to allow the date and time of the Operating System to be reinitialized during a Resume. To achieve the requirements for Suspend, it is generally mandatory that the Real Time Clock circuit **not** be reset on a Suspend followed by a Resume.

Device Support

Windows Devices

Windows Devices (i.e. devices that run any version of the Windows CE or Windows Mobile Operating Systems) generally support Suspend and Resume, although the exact mechanism used to implement Suspend and Resume may vary. In some cases, a Suspend might be as simple as halting the CPU until an interrupt occurs. In other cases, complex power management logic might be performed as part of Suspend and Resume to maximize power savings.

Windows PCs

Some Windows PCs (i.e. devices that run Windows XP), may support Suspend and Resume. This is most common in laptop or tablet PCs. In Windows XP, Suspend might be implemented using a low power mode of the device, similar to that described above for Windows Devices. Alternately, Suspend might be implemented via some form of hibernation approach, such as by saving RAM and CPU state to disk and then later restoring them.

Persistence Implications

Because its requirements demand essentially no loss of content or state, Persistence or content and state across Suspend and Resume is generally assumed for all devices that support Suspend and Resume.

Warm Boot

A Warm Boot is a type of reboot that restarts the Operating System with the minimum possible additional impact on the device. In particular, the impact on device hardware is often limited to just the loss of the contents of RAM.

Key Requirements

The key requirements for Warm Boot are:

- It may take a significant amount of time to prepare for and perform a Warm Boot.
- It may take a significant amount of time to start the Operating System and become operational after a Warm Boot.
- The contents of RAM will typically be lost across a Warm Boot.
- The state of the Operating System and all running applications will typically be lost across a Warm Boot.
- Any configuration changes made to the Operating System (e.g. via the Device Registry) will be reflected in its operation following a Warm Boot.
- The contents of any RAM Disk may or may not be lost as a result of a Warm Boot.
- The contents of other File Systems will be preserved across a Warm Boot.
- The Device Registry will be preserved across a Warm Boot.
- The date and time maintained by the Operating System will properly reflect any time elapsed during a Warm Boot.

Common Usage

Warm Boot is generally used when it is important to restart the Operating System while preserving the content of the device as much as possible.

Warm Boot might be initiated manually by a Device User to recover from some abnormal state, such as an application lockup. Warm Boot might also be initiated manually to cause some configuration change to take effect. Whether or not manual initiation of Warm Boot is supported may vary by device.

Warm Boot might be initiated programmatically on a device following re-configuration or the installation of new software in order to make the changes take effect. Whether or not programmatic initiation of Warm Boot is supported may vary by device. When a device supports a standard method to programmatically initiate a Warm Boot, MSP generally should support initiating a Warm Boot on that device.

Since a restart of the Operating System is an integral part of a Warm Boot, many devices rely on a Real Time Clock circuit to allow the date and time of the Operating System to be reinitialized following a Warm Boot. To achieve the requirements for Warm Boot, it is therefore generally mandatory that the Real Time Clock circuit **not** be reset on a Warm Boot.

Device Support

Windows Devices

Windows Devices (i.e. devices that run any version of the Windows CE or Windows Mobile Operating Systems) commonly support Warm Boot. Warm Boot of a Windows Device typically causes the contents of RAM to be lost as the Operating System reloads itself into RAM and re-initializes RAM during its startup.

Windows PCs

Windows PCs (i.e. devices that run Windows XP) commonly support Warm Boot. A Warm Boot on a Windows PC is also commonly referred to as a Restart (e.g. Start->Shut Down->Restart). Warm Boot of a Windows PC typically causes the contents of RAM to be lost as the Operating System reloads itself into RAM and re-initializes RAM during its startup.

Persistence Implications

Because its requirements demand minimal loss of content, Persistence of content across Warm Boot is generally assumed for all devices that support Warm Boot.

Applications may be configured to automatically start when the Operating System. This will result in new instances of such applications that are distinct from any instances that were running before the Warm Boot. Such new instances have no direct access to the state of those former instances. Some applications might be designed to save and restore selected state information (e.g. in a File System or in the Device Registry), but saving and restoring application state information is **not** an inherent part of a Warm Boot.

Safe Shutdown and Power-On

A Safe Shutdown is a type of shutdown that causes a device to enter a “very low power state” in a safe and orderly manner that ensures that no unexpected loss of information occurs. The reverse of a Safe Shutdown is a Power-On, which starts the operation of the device again.

Key Requirements

The key requirements for Safe Shutdown are:

- It may take a significant amount of time to prepare for and perform a Safe Shutdown.
- It may take a significant amount of time to start the Operating System and become operational after a Power-On.

- The state of the Operating System and all running applications will typically be lost as a result of a Safe Shutdown.
- The contents of RAM will typically be lost as a result of a Safe Shutdown.
- Any configuration changes made to the Operating System (e.g. via the Device Registry) will be reflected in its operation as a result of a Power-On after a Safe Shutdown.
- The contents of any RAM Disk will typically be lost as a result of a Safe Shutdown.
- The contents of other File Systems will be preserved between a Safe Shutdown and subsequent Power-On.
- The Device Registry will be preserved between a Safe Shutdown and subsequent Power-On.
- The date and time maintained by the Operating System will properly reflect any time elapsed between a Safe Shutdown and subsequent Power-On.
- A device will typically enter its lowest power state as a result of a Safe Shutdown and will remain in that state until a future Power-On.

Common Usage

Safe Shutdown is generally used when it is important to place the device into a state where it can safely remain shutdown for a long duration. This might be done to prepare a device for transportation, long-term storage, servicing, etc.

Safe Shutdown might be initiated manually by a Device User to cause a device to enter its lowest power state before an extended period of non-use or before changing the battery. Whether or not manual initiation of Safe Shutdown is supported may vary by device.

Safe Shutdown might be initiated programmatically on a device in reaction to some critical condition, such as a very low battery or detection that the battery is being removed. Whether or not programmatic initiation of Warm Boot is supported may vary by device.

Device Support

Windows Devices

Windows Devices (i.e. devices that run any version of the Windows CE or Windows Mobile Operating Systems) may or may not support Safe Shutdown and Power-In. On some devices, a Safe Shutdown followed by a Power-On will produce a result similar to that produced by a Warm Boot. On other devices, a Safe Shutdown followed by a Power-In may be more severe since due to the duration the device is not running and is in a very low power state.

Windows PCs

Windows PCs (i.e. devices that run Windows XP) commonly support Safe Shutdown. A Safe Shutdown on a Windows PC is also commonly referred to as a Shut Down (e.g. Start->Shut Down->Shut Down).

Persistence Implications

Because its requirements demand minimal loss of content, Persistence of content across Safe Shutdown and Power-On is generally assumed for all devices that support Safe Shutdown and Power-On. Safe Shutdown typically relies on some form of non-volatile storage (e.g. hard disk, flash memory, etc.) to retain content until a subsequent Power-On.

Since a restart of the Operating System is an integral part of a Power-On, many devices rely on a Real Time Clock circuit to allow the date and time of the Operating System to be reinitialized during a Power-On. To achieve the requirements for Safe Shutdown, it is generally mandatory that the Real Time Clock circuit **not** be reset on a Safe Shutdown.

Since a Safe Shutdown may be performed as a prelude to servicing device hardware, it is possible that the servicing process could unavoidably impact or even require replacement of some hardware component that is crucial to providing Persistence. In such cases, the Safe Shutdown never really results in a Power-On and the end result is more like a Cold Boot or Clean Boot, as described later in this chapter.

Cold Boot

A Cold Boot is a type of reboot that restarts the Operating System with a greater degree of impact on the device than would occur in a Warm Boot. In particular, a Cold Boot often includes more extensive resetting of device hardware and generally does not follow any orderly shutdown process.

Warning:

Cold Boot should be **avoided** except when **absolutely necessary** (e.g. when another option, such as a Warm Boot, will not suffice).

Since a Cold Boot does not follow any orderly shutdown process, it can result in **loss of data**. Such loss of data could negatively impact the operation of MSP with respect to the device and/or the operation of any applications on the device.

For more information on the potential hazards and impacts of performing a Cold Boot, see the details in the following subsections.

Key Requirements

The key requirements for Cold Boot are:

- It may take a significant amount of time to prepare for and perform a Cold Boot, but often a Cold Boot is faster than a Warm Boot because no orderly shutdown process is followed.
- It may take a significant amount of time to start the Operating System and become operational following a Cold Boot.
- The state of the Operating System and all running applications will typically be lost as a result of a Cold Boot.
- Any configuration changes made to the Operating System (e.g. via the Device Registry) might or might not be reflected in its operation following a Cold Boot.
- The contents of RAM will typically be lost as a result of a Cold Boot.
- The contents of any RAM Disk will typically be lost as a result of a Cold Boot.
- The contents of various File Systems may or may not be preserved across a Cold Boot.

- The Device Registry may or may not be preserved across a Cold Boot.
- Recent cached changes that have not yet been committed to non-volatile storage might be permanently lost as a result of a Cold Boot.
- The date and time maintained by the Operating System might not reflect the proper time following a Cold Boot. In some cases, the date and time maintained by the Operating System might be reset to some arbitrary value as a result of a Cold Boot.

Common Usage

Cold Boot is generally used when a Warm Boot is not available or does not accomplish the required goal (e.g. the device is locked-up too severely to perform a Warm Boot). Cold Boot might also be performed in a deliberate attempt to reset device hardware, either to correct a problem or to intentionally destroy data (e.g. wipe a stolen device).

Cold Boot might be initiated manually by a Device User to recover from some abnormal state, such as an application lockup when a Warm Boot could not be performed. Whether or not manual initiation of Cold Boot is supported may vary by device. When a Cold Boot is manually initiated, it generally does **not** follow an orderly shutdown process. For example, a Cold Boot generally would not “flush” cached changes to non-volatile storage.

Cold Boot might be initiated programmatically on a device to cause certain highly impactful changes to take effect. Whether or not programmatic initiation of Cold Boot is supported may vary by device. When a device supports a standard method to programmatically initiate a Cold Boot, MSP generally should support initiating a Cold Boot on that device. When MSP initiates a Cold Boot programmatically, it will first attempt to programmatically initiate a “flush” to commit cached changes, when possible, to minimize loss of content and state.

Device Support

Windows Devices

Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System commonly support Cold Boot. A Cold Boot on such devices typically performs hardware reset that causes the Operating System to restart without following an orderly shutdown process. Since the Operating System on such devices stores File Systems and the Device Registry in RAM, the end result is more like a Clean Boot, as described later in this chapter.

Windows Devices that run the Windows Mobile Operating System version 5.0 or higher commonly support Cold Boot. A Cold Boot on such devices typically performs a hardware reset, which causes the Operating System to restart without first performing any form of orderly shutdown process. Since the Operating System on these devices stores File Systems and the Device Registry in Flash Memory, a Cold Boot **may** be less impactful on these devices. But since a Cold Boot may prevent cached changes from being “flushed” to non-volatile storage, a Cold Boot may result in some undesirable loss of content or state.

Windows PCs

Windows PCs (i.e. devices that run Windows XP) commonly support Cold Boot. A Cold Boot on a Windows PC is also commonly referred to as a Hard Reset (e.g. Press a hardware reset button). Cold Boot of a Windows PC typically results in a loss of cached changes that are not “flushed” to non-volatile storage.

Persistence Implications

Because its requirements permit significant loss of content and/or state, Persistence across Cold Boot generally **should not** be assumed for devices that support Cold Boot.

Since a restart of the Operating System is an integral part of a Cold Boot, many devices rely on a Real Time Clock circuit to allow the date and time of the Operating System to be reinitialized following a Cold Boot. To allow the data and time to be preserved, it is highly desirable that the Real Time Clock circuit **not** be reset on a Cold Boot. Some devices **may** reset the Real Time Clock circuit on a Cold Boot, however, resulting in the inability to preserve the date and time across a Cold Boot.

Depending on the device, a Cold Boot may cause loss of content and state. For example, RAM Disk contents, cached changes to non-volatile storage, and date and time may commonly be lost. Some devices may also lose other important content and/or state information. Cold Boot should thus typically be considered a risky operation to perform on any device.

Important:

Whenever possible, a less impactful reboot or shutdown should be used in preference to a Cold Boot. For example, a Warm Boot, if available, should be used to reboot the device unless the increased impact of a Cold Boot is actually desired (e.g. loss of content and/or state is intended) or unavoidable (e.g. Warm Boot is not available or did not work).

If a device does **not** support a means to manually initiate a Warm Boot but **does** support a means to manually initiate a Cold Boot, then there may be no choice but to Cold Boot if a reboot is required. In such cases, performing a Suspend or a Safe Shutdown **before** performing a Cold Boot may reduce the chance of unintended loss of content and/or state.

Clean Boot

A Clean Boot is a type of reboot that restarts the Operating System with a greater degree of impact on the device than would occur in a Cold Boot. In most devices, a Clean Boot is the most extreme form of reboot available and is typically performed with the expression intention of causing loss or content and state.

Key Requirements

The key requirements for Clean Boot are:

- It may take a significant amount of time to prepare for and perform a Clean Boot.
- It may take a significant amount of time to start the Operating System and become operational following a Clean Boot.
- The state of the Operating System and all running applications will be lost as a result of a Clean Boot.
- The configuration of the Operating System will typically be returned to pre-defined defaults as a result of a Clean Boot.
- The contents of RAM will typically be lost as a result of a Clean Boot.
- The contents of any RAM Disk will typically be lost as a result of a Clean Boot.
- The contents of some or all File Systems will typically be lost as a result of a Clean Boot.
- The Device Registry will typically be lost as a result of a Clean Boot.

- Recent cached changes that have not yet been committed to non-volatile storage will typically be permanently lost as a result of a Clean Boot.
- The date and time maintained by the Operating System will typically be reset to some arbitrary value as a result of a Clean Boot.

Common Usage

Clean Boot is generally used when there is a deliberate need to return a device partially or completely back to pre-defined defaults (which might be factory-defined or customer-defined).

Clean Boot might be initiated manually by a Device User to intentionally return a device to a pre-defined default state. Whether or not manual initiation of Clean Boot is supported may vary by device. When a Clean Boot is manually initiated, loss of content and/or state should be expected.

Clean Boot might be initiated programmatically on a device to intentionally return a device to a pre-defined default state. This might be done as part of a Wipe operation on a device that is suspected to be lost or stolen. When a device supports a standard method to programmatically initiate a Clean Boot, MSP generally should support initiating a Clean Boot on that device. When a Clean Boot is programmatically initiated, loss of content and/or state should be expected.

Device Support

Windows Devices

Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System do not support Clean Boot. On such devices a Cold Boot already accomplishes the same function.

Windows Devices that run the Windows Mobile Operating System version 5.0 or higher commonly support Clean Boot because a Cold Boot is less impactful. A Clean Boot on such devices typically performs a hardware reset, similar to that performed by a Cold Boot. But in addition, a Clean Boot also typically performs a deliberate clearing of some File Systems and the Device Registry. The end result is a deliberate loss of data that would not otherwise have occurred as a result of a Cold Boot.

Windows PCs

Windows PCs (i.e. devices that run Windows XP) do not support Clean Boot.

Persistence Implications

Because its requirements permit, and ever dictate, significant loss of content and/or state, Persistence across Clean Boot generally **should not** be assumed for devices that support Clean Boot.

Restore Boot

A Restore Boot is a term used to refer to the most impactful reboot available for a given device. The term Restore Boot is chosen because such a reboot is generally used to restore a device back to pre-defined defaults (which might be factory-defined or customer-defined).

Key Requirements

The key requirements for Restore Boot are:

- It may take a significant amount of time to prepare for and perform a Restore Boot.
- It may take a significant amount of time to start the Operating System and become operational following a Restore Boot.
- The state of the Operating System and all running applications will be lost as a result of a Restore Boot.
- The configuration of the Operating System will typically be returned to pre-defined defaults as a result of a Restore Boot.
- The contents of RAM will typically be lost as a result of a Restore Boot.
- The contents of any RAM Disk will typically be lost as a result of a Restore Boot.
- The contents of some or all File Systems will typically be lost as a result of a Restore Boot.
- The Device Registry will typically be lost as a result of a Restore Boot.
- Recent cached changes that have not yet been committed to non-volatile storage will typically be permanently lost as a result of a Restore Boot.
- The date and time maintained by the Operating System will typically be reset to some arbitrary value as a result of a Restore Boot.

Key Requirements

The key requirements for Restore Boot are:

- It may take a significant amount of time to perform a Restore Boot.
- It may take a significant amount of time to start the Operating System and become operational following a Restore Boot.
- The state of the Operating System and all running applications will generally always be lost as a result of a Restore Boot.
- The configuration of the Operating System will typically be returned to pre-defined defaults as a result of a Restore Boot.
- The contents of RAM will typically be lost as a result of a Restore Boot.
- The contents of any RAM Disk will typically be lost as a result of a Restore Boot.
- The contents of some or all File Systems will typically be lost as a result of a Restore Boot.
- The Device Registry will typically be lost as a result of a Restore Boot.
- Recent cached changes that have not yet been committed to non-volatile storage will typically be permanently lost as a result of a Restore Boot.
- The date and time maintained by the Operating System will typically be reset to some arbitrary value as a result of a Restore Boot.

Common Usage

O is generally used when there is a deliberate need to return a device partially or completely back to pre-defined defaults (which might be factory-defined or customer-defined).

Restore Boot might be initiated manually by a Device User to intentionally return a device to a pre-defined default state. Whether or not manual initiation of Restore Boot is supported may vary by device. When a Restore Boot is manually initiated, loss of content and/or state should be expected.

Restore Boot might be initiated programmatically on a device to intentionally return a device to a pre-defined default state. This might be done as part of a Wipe operation on a device that is suspected to be lost or stolen. When a device supports a standard method to programmatically initiate a Restore Boot, MSP generally should support initiating a Restore Boot on that device. When a Restore Boot is programmatically initiated, loss of content and/or state should be expected.

Device Support

Windows Devices

For Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System, a Restore Boot is generally a Cold Boot, as described earlier in this chapter.

For Windows Devices that run the Windows Mobile Operating System version 5.0 or higher, a Restore Boot is generally a Clean Boot, as described earlier in this chapter..

Windows PCs

For Windows PCs (i.e. devices that run Windows XP), a Restore Boot is generally a Cold Boot.

Persistence Implications

Some devices may allow the pre-defined defaults, to which a device will be restored by a Restore Boot, to be customer-defined. In such cases, there generally must be some special memory or storage area that can Persist across a Restore Boot. Customer-defined defaults would need to be stored into that special memory or storage area and would need to be retrieved as part of the Restore Boot and used to configure the device.

The exact capabilities for configuring customer-defined defaults may vary by device. In some cases, such capabilities may provide significant capability for Persisting customer-defined content across a Restore Boot. For more information on Persisting across a Restore Boot, see the section Enhanced Persistence below.

OS Update

When the Operating System of a device must be updated, it usually has a very significant impact on device content and generally requires the equivalent of a Restore Boot at the end to make sure that the newly updated Operating System is started cleanly. The fact that the Operating System has changed means that a loss of content and/or state is typically desirable and/or required.

Key Requirements

The key requirements for OS Update are:

- It may take a significant amount of time to perform an OS Update.
- It may take a significant amount of time to start the Operating System and become operational following an OS Update.
- The state of the Operating System and all running applications will generally always be lost as a result of an OS Update.
- The configuration of the Operating System will typically be returned to pre-defined defaults as a result of an OS Update.
- The contents of RAM will typically be lost as a result of an OS Update.
- The contents of any RAM Disk will typically be lost as a result of an OS Update.
- The contents of some or all File Systems will typically be lost as a result of an OS Update.
- The Device Registry will typically be lost as a result of an OS Update.
- Recent cached changes that have not yet been committed to non-volatile storage will typically be permanently lost as a result of an OS Update.
- The date and time maintained by the Operating System will typically be reset to some arbitrary value as a result of an OS Update.

Common Usage

An OS Update is usually performed when there are bug fixes, new features, etc. that are valuable or necessary to support the intended use of a device. In many cases, a new Operating System may also include a newer version of MSP Client Software.

Device Support

Windows Devices

Windows Devices (i.e. devices that run any version of the Windows CE or Windows Mobile Operating Systems) may or may not support OS Update. Many, but not all, Motorola and Symbol mobile devices do support OS Update. Most non-Motorola devices do not support OS Update or, if they do, do not support OS Update initiated from MSP.

Windows PCs

Windows PCs (i.e. devices that run Windows XP) typically do not support OS Update.

Persistence Implications

In some cases and on some devices, it may be possible to provide Persistence across an OS Update. This is generally not possible for devices that do not also support Persistence across a Restore Boot because an OS Update generally includes a Restore Boot as an integral step. Depending on the device, it may be possible to support Persistence across an OS Update between certain OS versions and not between others due to the changes implemented between one OS version and another.

What is Persistence?

Persistence is a term used to describe the lifecycle of information stored in a device across various kinds of reboots and shutdowns. Various types of devices support different types of reboots and shutdowns, each of which has its own degree of Persistence. Some devices may also provide Enhanced Persistence Mechanisms that can be used to increase the Persistence of certain types of information across various types of reboots and shutdowns.

Persistence of Common Items

The previous section explained the various types of reboots and shutdowns that a device might support. This section will identify key items in a device that might or might not be Persistent and discuss the degree of Persistence these items might have across various types of reboots and shutdowns on various devices.

Date and Time

Devices generally need to maintain the date and time, both while they are running and across periods when they are not. To accomplish this, most devices rely on some form of Real-Time Clock circuit to maintain the date and time independently of the Operating System.

The date and time maintained by the Operating System typically cannot Persist most reboots or shutdowns of the Operating System. In the event of a reboot or shutdown, the date and time maintained by the Real-Time Clock circuit is commonly used to reinitialize the date and time maintained by the Operating System.

The degree to which a Real-Time Clock circuit can be used to enable the Persistence of the date and time may vary based on the type of reboot or shutdown and based on the device. In particular, certain types of reboots or shutdowns on certain types of devices might reset the Real-Time Clock circuit. If the Real-Time Clock circuit is reset, it cannot be used to reinitialize the date and time maintained by the Operating System and hence the date and time of the device will not be Persistent across that type of reboot or shutdown.

Windows Devices

For Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System, the Real-Time Clock circuit is generally **not** reset on a Warm Boot and generally **is** reset on a Cold Boot.

For Windows Devices that run the Windows Mobile Operating System version 5.0 or higher, the Real-Time Clock circuit is generally **not** reset on a Warm Boot, may or may not be reset on a Cold Boot, and generally **is** reset on a Clean Boot.

Windows PCs

For Windows PCs (i.e. devices that run Windows XP), the Real-Time Clock circuit is generally reset **only** when power has been completely removed from the device for an extended period of time.

RAM

Devices generally utilize RAM to load and run the Operating System, load and run programs, hold transient program data, etc. RAM typically is fast to read, is fast to write, and is volatile. RAM is therefore generally **not** Persistent across any type of reboot and may or may not be Persistent across certain types of shutdown.

The Operating System and most applications on most devices are designed to be loaded into RAM from a File System that provides at least some Persistence across reboots and shutdowns. This is necessary if a device is to be rebooted or shut down without a major loss of functionality. Similarly, important application data may be stored to and restored from a File System that provides some Persistence across reboots and shutdowns.

RAM Disk

Devices may allocate some amount of RAM to create and maintain a volatile File System. Such a File System is commonly called a RAM Disk or Cache Disk. Because RAM is allocated for a RAM Disk, any data stored on the RAM Disk typically will **not** be Persistent across **many** types of reboots and shutdowns.

On some devices, specific enhancements may permit a RAM Disk to survive across various types of reboots or shutdowns. On devices that support the ability to suspend, a RAM Disk will usually be designed to Persist across suspend. On devices that support a Warm Boot, a RAM Disk **may** be designed to be capable of being Persistent across a Warm Boot. Such enhancements may vary by the device and/or by the implementation of the RAM Disk. Few, if any, devices are likely to support a RAM Disk that is capable of being Persistent across a Cold Boot or Clean Boot.

Built-In Flash Memory

Many devices use Built-In Flash Memory to store the Operating System and may also allocate Built-In Flash to implement one or more Flash File Systems. Built-In Flash Memory is typically slower to read than RAM, much slower to write than RAM, but is non-volatile. Built-In Flash Memory is therefore generally **much more** Persistent than RAM, although most devices have at least **some** type(s) of reboots or shutdowns across even Built-In Flash Memory is **not** Persistent.

Flash File System

A File System located in Built-In Flash Memory typically provides a **much greater** degree of Persistence than would typically be provided for a File System located in RAM. In most cases, if a device provides a File System located in Built-In Flash Memory, it will be Persistent across **most** reboots and shutdowns. Such Persistence may vary by the device and/or by the implementation of the Flash File System.

To enhance performance, many devices will cache changes made to Flash File Systems. Cached changes are generally written on a delayed basis to avoid repeatedly writing data that may be frequently modified. Since caching is usually done in RAM, Cold Boot and Clean Boots may result in a loss cached changes that have not yet been “flushed”.

Windows Devices

For Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System, Flash File Systems are not a standard feature. Many Motorola and Symbol Windows Devices offer Super-Persistent File Systems as discussed later in this chapter.

For Windows Devices that run the Windows Mobile Operating System version 5.0 or higher, Flash File Systems are a standard feature. Many Motorola and Symbol Windows Devices also offer Super-Persistent File Systems as discussed later in this chapter.

Windows PCs

For Windows PCs (i.e. devices that run Windows XP), a Flash File System may or may not be available. When available, a Flash File System would generally have the same Persistence as a Storage Media File System as defined below.

Storage Media File System

Devices may support a variety of other Storage Media, such as hard disks, removable storage media (e.g. SD card), etc. on which File Systems can be located. A Storage Media File System typically provides a very high degree of Persistence. In most cases, the contents of a Storage Media File System will Persist across all types of reboots and shutdowns and will lose its contents **only** if explicit action is taken to do so (e.g. format disk).

To enhance performance, many devices will cache changes made to Storage Media File Systems. Cached changes are generally written on a delayed basis to avoid repeatedly writing data that may be frequently modified. Since caching is usually done in RAM, Cold Boot and Clean Boots may result in a loss cached changes that have not yet been “flushed”.

For security purposes, some devices might offer a mechanism (e.g. wipe removable storage) to erase data on Storage Media to protect proprietary data from unintended disclosure. MSP can usually offer the ability to Wipe data from such File Systems on most devices.

Device Registry

In devices running a Windows-type Operating System (e.g. Windows CE, Windows Mobile, Windows XP, etc.) the Device Registry is used to store information about the device, the Operating System, and applications. The Persistence of the Device Registry can often be as important as the Persistence of File Systems in determining how a device will perform across any type of reboot or shutdown.

Devices running non-Windows-type Operating Systems may or may not have anything that is equivalent to the Device Registry. Devices that do have some equivalent to the Device Registry likely will also need to provide a way for it to Persist across reboots and shutdowns.

To enhance performance, many devices will cache changes made to the Device Registry. Cached changes are generally written on a delayed basis to avoid repeatedly writing data that may be frequently modified. Since caching is usually done in RAM, Cold Boot and Clean Boots may result in a loss cached changes that have not yet been “flushed”.

Windows Devices

For Windows Devices that run versions of the Windows CE Operating System or Windows Mobile 2003 Operating System, the Device Registry is stored in RAM. Since RAM is volatile, the Device Registry generally will **not** Persist across any reboot or shutdown where RAM does not Persist. This generally means that the Device Registry will **only** Persist across a Suspend and Resume or a Warm Boot. In particular, the Device Registry will **not** Persist across a Cold Boot.

For Windows Devices that run the Windows Mobile Operating System version 5.0 or higher, the Device Registry is stored in Flash Memory. Since Flash Memory is non-volatile, the Device Registry generally **will** Persist across most reboots or shutdowns where Flash Memory Persists. This generally means that the Device Registry will Persist across everything except a Clean Boot.

Windows PCs

For Windows PCs (i.e. devices that run Windows XP), the Device Registry is typically stored in the Primary File System, which generally is Persistent across most reboots and shutdowns.

Enhanced Persistence Mechanisms

Some devices, especially many Motorola and Symbol Windows Devices, provide value-add Enhanced Persistence Mechanisms to increase the Persistence that can be provided across various reboots and shutdowns. For purposes of this discussion, we will concentrate on enhancements to Persistence across a Restore Boot, since it is often the worst-case scenario to which devices will commonly be subjected.

Super-Persistent File Systems

The first part of providing Persistence across Restore Boot in most devices typically involves the addition of one or more Super-Persistent File Systems. The key characteristic of a Super-Persistent File System is that the contents of such a File System are not lost as a result of a Restore Boot. Each Super-Persistent File System has a unique root path that is used to access the files in that File System.

Many Motorola and Symbol Windows Devices have a Super-Persistent File System, known as the Application Partition, whose unique path is “\Application”. MSP can leverage this “Super-Persistent File System”, on devices which support it, to enhance the overall usability and functionality of the devices in a MSP-managed environment. The methods MSP uses to achieve this are covered in Understanding MSP 3.3.1 – Understanding Persistence.

Transient File Persistence

Super-Persistent File Systems alone do not directly solve all issues of Persistence related to File Systems. Some applications may require that certain files be located on and/or be run from File Systems other than Super-Persistent File Systems. For example, an application might require that a certain .DLL file be located in the “\Windows” folder, (which is generally **not** part of any Super-Persistent File System.

Many Motorola and Symbol Windows Devices provide a “Copy Files” mechanism that looks for and executes .CPY files located in the root folder of a Super-Persistent File System. Each .CPY file directs the “Copy Files” mechanism to copy one or more files from a Super-Persistent File System to some other File System (usually one that is **not** Super-Persistent). The end result is that files in a File System that is not a Super-Persistent File System gain the appearance of being Persistent across a Restore Boot.

MSP can leverage the “Copy Files” mechanism, on devices which support it, to enhance the overall usability and functionality of the devices in a MSP-managed environment. The methods MSP uses to achieve this are covered in Understanding MSP 3.3.1 – Understanding Persistence.

Device Registry Persistence

Super-Persistent File Systems alone do not address the need for Persistence of content in the Device Registry across a Restore Boot. This is because even on devices that support one or more Super-Persistent File Systems, the Device Registry is typically **not** stored in a Super-Persistent File System and hence is **not** inherently Persistent across a Restore Boot. In the event of a Restore Boot, the Device Registry typically reverts to factory-defined defaults.

Many Motorola and Symbol Windows Devices provide a “Registry Merge” mechanism that looks for and executes .REG files located in the root folder of a Super-Persistent File System. Each .REG file directs the “Registry Merge” mechanism to merge a set of changes into the Device Registry. The end result is that changes to the Device Registry gain the appearance of being Persistent across a Restore Boot, even though the Device Registry itself is not.

MSP can leverage the “Registry Merge” mechanism, on devices which support it, to enhance the overall usability and functionality of the devices in a MSP-managed environment. The methods MSP uses to achieve this are covered in Understanding MSP 3.3.1 – Understanding Persistence.

Re-launch Persistence

In some cases, an application or other process is designed to run continuously and hence is expected to re-launch automatically on any reboot. While all Windows Devices have one or more mechanisms to automatically launch process during a boot, the automatic launching implemented by these mechanisms is generally **not** Persistent across a Restore Boot.

By using some of the other Enhanced Persistence mechanisms listed above, such as Transient File Persistence or Device Registry Persistence, it may be possible to achieve Re-launch Persistence. The Re-launch Persistence mechanisms available will vary by device.

MSP can leverage Re-launch Persistence mechanisms, on devices which support them, to enhance the overall usability and functionality of the devices in a MSP-managed environment. The methods MSP uses to achieve this are covered in Understanding MSP 3.3.1 – Understanding Persistence.

The following are some examples of Re-launch Persistence mechanisms that may be available on some Motorola and Symbol Windows Devices:

“\Windows\Startup” Folder

On most Windows Devices, any file located in the “\Windows\Startup” folder will automatically be launched during a boot. By using the Transient File Persistence mechanism, a copy of or a link to an executable program contained within a Super-Persistent File System can be copied to the “\Windows\Startup” folder. This can cause the executable program to be automatically launched following a Restore Boot, on devices that support all the Enhanced Persistence mechanisms described above.

“\Application\Startup” Folder

On many Motorola and Symbol Windows Devices, any file located in the “\Application\Startup” folder will automatically be launched during a boot. Since this folder is already part of a Super-Persistent File System, simply placing the required executable program or link into the location can cause the executable program to be automatically launched following a Restore Boot, on devices that support the Enhanced Persistence mechanisms described above.

Program Launch Registry Keys

On many Motorola and Symbol Windows Devices, entries can be placed into the Device Registry to control which executable programs are launch during boot, and in which order. By using the Device Registry Persistence mechanism on such devices, a desired set of executable programs can be automatically launched following a Restore Boot.

Chapter 5 – Network Adapters

What is a Network Adapter?

"A network interface card, network adapter, network interface controller (NIC), or LAN adapter is a computer hardware component designed to allow computers to communicate over a computer network."

(Wikipedia)

In MSP, a Network Adapter is a physical or logical component of a managed device that permits that device to access a network. MSP is concerned with access to a network over which management operations can be performed. Nonetheless, MSP can also provide services related to Network Adapters that are used to access networks over which management operations are never performed.

Network Adapter Status

An important concept in MSP is Network Adapter Status. The Network Adapter Status of a Network Adapter on a managed device determines how or if that Network Adapter can be used at a given time. The Network Adapter Status for a given Network Adapter might have any one of the following values:

Status = Present

A Network Adapter on a managed device is considered to have a Network Adapter Status of Present if the Network Adapter is built into the device or currently present and attached to the device, but may not be usable.

A Network Adapter Status of Present basically indicates that the device has the potential to use that Network Adapter, but is not currently be in state where it can actually realize that potential, such as when the Network Adapter is physically powered off.

When the Network Adapter Status of a Network Adapter is Present, it generally is not possible to actually use that Network Adapter at that time. For example, a managed device whose WWAN radio was physically powered off would typically be considered to have a Network Adapter Status of Present for that WWAN Network Adapter.

Network Adapters with a Network Adapter Status of Present exhibit the behaviors with regard to standard Condition Classes and Control Modules as described in the following subsections.

Adapter Condition Class*

The Adapter Condition Class cannot be used to detect Network Adapters with a Network Adapter Status of Present – it can only be used to detect a Network Adapters with a Network Adapter Status of Active or Connected. For more information on the Adapter Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

AdapterTime Condition Class

The AdapterTime Condition Class cannot be used to detect Network Adapters with a Network Adapter Status of Present – it can only be used to detect a Network Adapters with a Network Adapter Status of Connected. For more information on the AdapterTime Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

GetAdapters Control Module

The GetAdapters Control Module will not report information about Network Adapters with a Network Adapter Status of Present - it will only report information about Network Adapters with a Network Adapter Status of Connected. This is because software enumeration is generally not possible for such Network Adapters. For more information about the GetAdapters Control Module, see Understanding MSP 3.3.1 – Understanding Control Modules.

GetConfigData Control Module

The GetConfigData Control Module may be used to report information that could be used to determine that a device has a certain capability, such as WLAN or WWAN radio, and hence to infer that a Network Adapter on that device might have a Network Adapter Status of Present. For more information about the GetConfigData Control Module, see Understanding MSP 3.3.1 – Understanding Control Modules.

Status = Active

A Network Adapter on a managed device is considered to have a Network Adapter Status of Active if the Network Adapter is physically powered on and hence can be enumerated by software running on the device.

A Network Adapter Status of Active basically indicates that the Network Adapter is available on the device, but may or may not be configured for use on access any particular network.

When the Network Adapter Status of a Network Adapter is Active, it generally is not possible to actually use that Network Adapter to access a network, but it generally is possible to configure that Network Adapter such that it could be used to access a network.

Network Adapters with a Network Adapter Status of Active exhibit the behaviors with regard to standard Condition Classes and Control Modules as described in the following subsections.

Adapter Condition Class

The Adapter Condition Class can be used to detect Network Adapters with a Network Adapter Status of Active. For more information on the Adapter Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

AdapterTime Condition Class

The AdapterTime Condition Class cannot be used to detect Network Adapters with a Network Adapter Status of Present – it can only be used to detect a Network Adapters with a Network Adapter Status of Connected. For more information on the AdapterTime Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

GetAdapters Control Module

The GetAdapters Control Module will not report information about Network Adapters with a Network Adapter Status of Present - it will only report information about Network Adapters with a Network Adapter Status of Connected. This is because software enumeration is generally not possible for such Network Adapters. For more information about the GetAdapters Control Module, see Understanding MSP 3.3.1 – Understanding Control Modules.

Status = Connected

A Network Adapter on a managed device is considered to have a Network Adapter Status of Connected if the Network Adapter is physically powered on and has been assigned a valid IP Address.

Note:

For more information on what constitutes a valid IP Address, see the section Valid IP Addresses, later in this chapter.

A Network Adapter Status of Connected basically indicates that the Network Adapter **may** be ready to use to access some network.

Note:

For more information on when a Network Adapter can be used to access a network, see the section Network Accessibility, later in this chapter.

Network Adapters with a Network Adapter Status of Connected exhibit the behaviors with regard to standard Condition Classes and Control Modules as described in the following subsections.

Adapter Condition Class

The Adapter Condition Class can be used to detect Network Adapters with a Network Adapter Status of Connected. For more information on the Adapter Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

AdapterTime Condition Class

The AdapterTime Condition Class can be used to detect Network Adapters with a Network Adapter Status of Connected. For more information on the AdapterTime Condition Class, see Understanding MSP 3.3.1 – Understanding Condition Classes.

GetAdapters Control Module

The GetAdapters Control Module will report information about Network Adapters with a Network Adapter Status of Connected. For more information about the GetAdapters Control Module, see Understanding MSP 3.3.1 – Understanding Control Modules.

Network Adapter Names

A Network Adapter on a managed device must always be identified by a Network Adapter Name that must be unique (at least on that device). Depending on the Device Class and even on the model of device within a Device Class, Network Adapter Names may vary tremendously. There is no common standard mechanism that can be described to query the available Network Adapter Names from the UI of a device.

As described above, the GetAdapters Control Module can enumerate and report Network Adapters with a Network Adapter Status of Connected. In rare cases, it may be necessary to know a Network Adapter Name in order to create an AdapterTime Condition Object to test whether that Network Adapter has a Network Adapter Status of Connected on a given device. In such cases, the recommended approach would be to deploy the GetAdapters Control Module to that device and then examine the Device Attributes reported to MSP for that device. For more information on the AdapterTime Condition, see the MSP Online Help – Condition Create.

When entering a Network Adapter name into an AdapterTime Condition Object, use the Device Attribute reported by the GetAdapters Control Module whose name ends in ".Name". For example, the Device Attribute "UserAttribute.Adapter.SomeAdapter.Name" will contain the Network Adapter name that must be entered for the Network Adapter "SomeAdapter".

Network Adapter Class

When managing a device, it is often undesirable to have to know and specify the actual Network Adapter Name for a Network Adapter Name. Network Adapter Names are often long, complex, and unintuitive. In many cases, it would be much easier and less error-prone to simply specify the type of Network Adapter we are interested in. For example, we might want to permit some operation to occur over a WLAN Network Adapter, but not over a WWAN Network Adapter.

It would be ideal if devices had an inherent capability to report a Network Adapter Class for each Network Adapter that could be used to make such operations simpler. Alas most devices do not. Nonetheless, MSP has introduced a Network Adapter Class concept to make working with Network Adapters easier and more effective. But since the devices do not have an inherent capability to report a Network Adapter Class, MSP must provide special code to do it.

The GetAdapters Control Module and the AdapterTime Condition Class both incorporate special code to determine the Network Adapter Class for each Network Adapter. Since there is no way to determine the Network Adapter Class for a Network Adapter from the device itself, both contain a lookup table for determining the Network Adapter Class based on the Network Adapter Name.

Since new Network Adapter Names may be added over time, the GetAdapters Control Module and the AdapterTime Condition Class are periodically updated to extend the table and permit Network Adapter Classes to be determined for new Network Adapter Names.

If a device has a Network Adapter Name that is not in the table, then the GetAdapters Control Module and the AdapterTime Condition Class will not be able to work with that Network Adapter based on its Network Adapter Class. In such a case, it is possible to work with that Network Adapter based on its Network Adapter Name or the table can be dynamically extended as described in the section Network Adapter Extensibility, later in this chapter.

The Network Adapter Classes that are currently known by the GetAdapters Control Module and the AdapterTime Condition Class are described in the following subsections.

Virtual Private Network (VPN)

This Network Adapter Class is used for Network Adapters that provide access to a Virtual Private Network, usually via some other Network Adapter. In some cases, a Virtual Private Network may be preferable or mandatory to use because of the access it provides (e.g. to an Enterprise network). In other cases, a Virtual Private Network may intentionally not be used for certain types of accesses (e.g. to go directly to the internet).

Ethernet Cradle

This Network Adapter Class is used for Network Adapters that provide access to wired Ethernet network, often via a docking cradle. Such networks are generally have higher bandwidth and are less costly than wireless networks. In some cases, this may make such a network preferable to use when possible.

Wireless LAN (WLAN)

This Network Adapter Class is used for Network Adapters that provide access to a Wireless Local Area Network (WLAN). Such networks are generally have higher bandwidth and are less costly than other types of wireless networks. In some cases, this may make such a network preferable to use when possible.

Wireless WAN (WWAN)

This Network Adapter Class is used for Network Adapters that provide access to Wireless Wide Area Network (WWAN). Such networks are generally have lower bandwidth and are more costly than other types of wireless networks. In some cases, this may make such a network preferable to use only when no other network is available.

ActiveSync

This Network Adapter Class is used for virtual Network Adapters implemented by Microsoft ActiveSync™ connections between a device and a Workstation PC. Such networks generally have moderate bandwidth but tend to be somewhat unreliable. In most cases, this may make such a network preferable to use only when no other network is available.

Network Adapter Priority

In a similar manner to that described earlier for Network Adapter Class, it is sometimes desirable to order or rank Networks Adapters based on their “desirability”. As with Network Adapter Class, devices generally lack an inherent capability to report a Network Adapter Priority. To enable ranking of Networks Adapters, MSP has introduced a notion of Network Adapter Priority.

As described for Network Adapter Class, Network Adapter Priority, the GetAdapters Control Module and the AdapterTime Condition Class both incorporate special code to determine the Network Adapter Priority for each Network Adapter. Since there is no way to determine the Network Adapter Priority for a Network Adapter from the device itself, both contain a lookup table for determining the Network Adapter Priority based on the Network Adapter Name. In fact the **same** lookup table used to determine Network Adapter Class is used to determine Network Adapter Priority.

Since new Network Adapter Names may be added over time, the GetAdapters Control Module and the AdapterTime Condition Class are periodically updated to extend the table and permit Network Adapter Priorities to be determined for new Network Adapter Names.

If a device has a Network Adapter Name that is not in the table, then the GetAdapters Control Module and the AdapterTime Condition Class will not be able to determine the Network Adapter Priority for that Network Adapter. In such a case, the table can be dynamically extended as described in the section Network Adapter Extensibility, later in this chapter.

Best Connected Network Adapter

Quite a variety of devices supported by MSP and an even greater of variety of Network Adapters might be available across those devices. In some cases, only one Network Adapter will have a Network Adapter Status of Connected on any one device at any one time. But in other cases, a single device might have multiple Network Adapters with a Network Adapter Status of Connected at the same time.

MSP needs a flexible way of dealing with the variety of possible combinations of Network Adapters that might occur across of potentially heterogeneous population of managed devices. When determining the Network Adapter to use, especially for initiation Remote Control, Remote UI, or Real-Time Management tasks, MSP needs a way to select a **single** Network Adapter.

MSP handles this by defining the notion of a Best Connected Network Adapter. The Best Connected Network Adapter is simply defined as the Network Adapter with a Network Adapter Status of Connected that has the highest Network Adapter Priority. The rules for the Best Connected Network Adapter are as follows:

- If a given managed device has no Network Adapters that have a Network Adapter Status of Connected at a given time, then that device has no the Best Connected Network Adapter for that device at that time.
- If a given managed device has a single Network Adapter with a Network Adapter Status of Connected at a given time, than that Network adapter that device is the Best Connected Network Adapter for that device at that time.
- If a given managed device has multiple Network Adapters that have a Network Adapter Status of Connected at a given time, then the Network Adapter with the highest (numerically greater) Network Adapter Priority is the Best Connected Network Adapter for that device at that time. If there is a tie for highest Network Adapter Priority, a winner is arbitrarily chosen.

Valid IP Addresses

When a Network Adapter is enumerated, the determination of whether it will be considered to have a Network Adapter Status of Active or Connected is made based on whether there is a valid IP Address assigned to that Network Adapter. The following rules are used to determine whether a Network Adapter has a valid IP Address:

- If a Network Adapter has nor been assigned an IP Address, then it is considered to not have a Valid IP Address, and hence is considered to have a Network Adapter Status of Active instead of Connected.
- If a Network Adapter has been assigned an IP Address that starts with “0.”, then it is considered to not have a Valid IP Address, and hence is considered to have a Network Adapter Status of Active instead of Connected.

- If a Network Adapter has been assigned an IP Address that starts with "169.254", then it is considered to not have a Valid IP Address, and hence is considered to have a Network Adapter Status of Active instead of Connected. Such an IP Address would typically be assigned by AutoIP and generally indicates a lack of usable network connectivity.
- If a Network Adapter has been assigned an IP Address of "127.0.0.1", then it is considered to not have a Valid IP Address, and hence is considered to have a Network Adapter Status of Active instead of Connected. Such an IP Address identifies the "virtual loopback interface" and generally indicates a lack of usable network connectivity.
- If a Network Adapter has been assigned any other IP Address, then it is considered to not have a Valid IP Address, and hence is considered to have a Network Adapter Status of Connected instead of Active.

Network Accessibility

When a Network Adapter is determined to have a Network Adapter Status of Connected, it is considered that the Network Adapter **may** be ready to use to access some network. This somewhat ambivalent statement is due to the uncertainties inherent in the determination of the Connected Network Adapter Status.

A Network Adapter on a device could be assigned in a variety of ways, most notably via Dynamic IP Assignment, such as using the Dynamic Host Configuration Protocol (DHCP), or via Static IP Assignment.

When an IP Address is assigned dynamically, it is likely, but not guaranteed, that the assigned IP Address can be used to access the network for which the IP Address was assigned. But DHCP, for example, permits IP Addresses to be cached and re-used, to better handle cases where a DHCP Server may be temporarily unreachable. This means that a Network Adapter on a device might be assigned what appears to be a valid IP Address that may not be valid for use on the network. In fact the network might not even be available. For this reason, a determination that the Network Adapter has a Network Adapter Status of Connected based on the dynamic assignment of an IP Address suggests, but does not guarantee that network access is possible via that Network Adapter.

When an IP Address is assigned statically, it can be anything the assigner wants it to be. An assigned IP Address may or may not be correct or suitable for a given network. For this reason, a determination that the Network Adapter has a Network Adapter Status of Connected based on the static assignment of an IP Address suggests, but does not guarantee that network access is possible via that Network Adapter.

When using an IP Address to contact a device, such as to initiate Remote Control, Remote UI, or a Real-Time Management function, it makes sense to base the choice of Network Adapter on the existence of a Valid IP Address. If the device turns out not to be contactable at that IP Address, the connection will simply fail gracefully.

When considering using an AdapterTime Condition to control when to perform some operation on a device, the uncertainty of basing the decision on the existence of a Valid IP Address might or might not be considered adequate. If the uncertainty of using the existence of a Valid IP Address is unacceptable, then it may be better to use the Connectivity Condition Class, since it actually bases its decision on whether a specified target can be reached via the network. For more information on the AdapterTime Condition Class and the Connectivity Condition Class, see the MSP Online Help – Condition Create.

Network Adapter Extensibility

As mentioned earlier, the GetAdapters Control Module and the AdapterTime Condition Class may not be able to determine the Network Adapter Class or Network Adapter Priority for a Network Adapter because the Network Adapter of that Network Adapter is not in the lookup table built into the code.

In such cases, the GetAdapters Control Module **will** report information about the Network Adapter, when it has a Network Adapter Status of Connected, but that information will not include the Network Adapter Class or Network Adapter Priority. In such cases, the AdapterTime Condition Class can test for the Network Adapter, but cannot do so based on Network Adapter Class.

As new versions of MSP and/or the GetAdapters Control Module and the AdapterTime Condition Class Plug-In are released, the built-in lookup table will be extended to add new Network Adapters that have been discovered since the prior release. But in some cases, it may be desirable to not have to wait for new versions. This section will describe how to use a feature called Network Adapter Extensibility to extend the GetAdapters Control Module and the AdapterTime Condition Class to recognize new Network Adapters and properly report Network Adapter Class or Network Adapter Priority for them.

Before the GetAdapters Control Module and the AdapterTime Condition Class Plug-In consult their built-in lookup tables to attempt to map a Network Adapter Name to a Network Adapter Class and Network Adapter Priority, they check the Device Registry to see if Network Adapter Extensibility is being used and if it provides information for that Network Adapter Name.

The base path in the Device Registry used to store information for Network Adapter Extensibility is “HKEY_LOCAL_MACHINE\Software\MSP\Adapters\ Inclusions”. To add information for a new Network Adapter, a sub-key under the above path with a name the same as the Network Adapter would be created. Under that sub-key, two values would be created to define the Network Adapter Class and Network Adapter Priority to be used for that Network Adapter. This might look as follows:

```
[HKEY_LOCAL_MACHINE\Software\MSP\Adapters\Reporting\Inclusions\MyAdapter]
“Class”=”WLAN”
“Priority”=DWORD:4000
```

Table 10 below lists the supported values for Network Adapter Class and the values for Network Adapter Priority that would typically be assigned to Network Adapters of each Network Adapter Class.

Table 10 – Command Line Arguments and Results

Network Adapter Class Name Value	Typical Network Adapter Priority Value
VPN	10000
Cradle	8000
WLAN	6000
WWAN	4000

Network Adapter Class Name Value	Typical Network Adapter Priority Value
ActiveSync	2000

To use Network Adapter Extensibility to extend the GetAdapters Control Module and the AdapterTime Condition Class, the requisite Registry changes must be made before the GetAdapters Control Module and the AdapterTime Condition Class is executed. Network Adapter Extensibility is typically configured by deploying a .REG file containing the requisite Registry changes. The .REG file might be deployed as part of a Package or using a Registry Settings Object.

Note:

Both the GetAdapters Control Module and the AdapterTime Condition Class Plug-In read the Registry to see if Network Adapter Extensibility is configured each time they need to determine the Network Adapter Class and Network Adapter Priority of a Network Adapter. So, if the Registry changes between executions, the behavior will change to match.

Chapter 6 – Control Modules

Overview

This chapter provides detailed information on the operation and behavior of any Control Modules whose that are specific to certain Device Classes and/or whose behavior varies by Device Class.

Windows Devices

GetConfigData

The GetConfigData Control Module obtains data using Resource Coordinator (RCM) APIs. The Resource Coordinator (RCM) APIs are generally only supported on Motorola and Symbol Windows Devices.

Operation

The GetConfigData Control Module obtains data for various configuration items that can be requested when it is configured via a Control.GetConfigData Settings Object. For each requested configuration item, one of the following results can occur.

- The device does not support the Resource Coordinator (RCM) APIs.

  In this case, the Device Attribute associated with the configuration item will **not** be reported to MSP.

- The device supports the Resource Coordinator (RCM) APIs, but does not support the code specified to be passed for the configuration item (i.e. the API returns an error when that code is passed).

  In this case, the Device Attribute associated with the configuration item will not be reported to MSP.

- The device supports the Resource Coordinator (RCM) APIs, and supports the code specified to be passed for the configuration item (i.e. the API returns success when that code is passed).

  In this case, the Device Attribute associated with the configuration item **will** be reported to MSP as having the value returned by the API.

Device Attributes

The actual Device Attributes reported by the GetConfigData Control Module will vary depending on the configuration items requested. The Device Attribute names for the pre-defined configuration items, along with a brief explanation of the possible values returned are shown in Table 11 below.

Note:

As new devices are added, new values will be returned. For the latest information on possible values, consult the Resource Coordinator (RCM) documentation in the latest Motorola Enterprise Mobility Developer's Kit (EMDK) for C and/or contact Motorola Technical Support for updated information for a specific device.

Configuration Item Name	Device Attribute Name	Possible Value Explanation
MPA Version	**UserAttributes.GetConfigData.MpaVersion**	Major Version.Minor Version (e.g. 2.0, 3.0, etc.)
Accelerometer	**UserAttributes.GetConfigData.Accelerometer**	1 = KIONIX (MC95) 3 = ??? (ES400)
Audio	**UserAttributes.GetConfigData.Audio**	0 = No Audio 1 = Beeper ❶ 2 = CODEC ❷ 4 = CODEC ❶ and Beeper ❷ 8 = Speaker ❸, Jack ❹, Mic❺, and Receiver ❻ 16 = Speaker ❸, Jack ❹, and Mic ❺ 32 = Speaker ❸ and Jack ❹ 64 = Speaker ❸, Mic ❺, and Receiver ❻
Barcode Scanner	**UserAttributes.GetConfigData.BarcodeScanner**	0 = No Scanner 1 = SE1200HP (1D) 2 = SE1224 (1D) 4 = SE2200 (1D/2D) 8 = Pico Imager (1D/2D) 16 = SE1524 (1D Long Range) 32 = SE923 (1D) 64 = SE824 (1D) 128 = SE950 (1D) 256 = SE955 (1D) 512 = Pico Imager HD (1D/2D) 1024 = Pico Imager w/Dot (1D/2D) 2048 = Block Buster Imager (1D/2D)

Configuration Item Name	Device Attribute Name	Possible Value Explanation
Biometrics	**UserAttributes.GetConfigData.Biometrics**	0 = No Biometrics 1 = Atrua 2 = UPEK 3 = Authentec
Bluetooth	**UserAttributes.GetConfigData.Bluetooth**	0 = No Bluetooth 1 = Taiyo Yuden (MPA 1.0) 2 = Broadcom 2045 (MPA 1.5) 3 = Broadcom 2046 (MPA 2.0) 4 = Qualcomm
Camera	**UserAttributes.GetConfigData.Camera**	0 = No Camera 1 = Type0 [TBD] 2 = 2 MP Auto Focus (Model 2MP_AF_FL) 3 = 3 MP Auto Focus (Model 3MP_AF_F0) 4 = 3 MP Auto Focus (Model 3MP_AF_J0) 5 = 2 MP Auto Focus (Model 2MP_AF_J0) 6 = 3 MP Auto Focus (Model 3MP_AF_J1)
Device	**UserAttributes.GetConfigData.Device**	[TBD]
Flash Size	**UserAttributes.GetConfigData.FlashSize**	Size of installed Flash in MegaBytes (MB)
GPS	**UserAttributes.GetConfigData.GPS**	0 = No GPS 1 = SiRFstar III 2 = Qualcomm

Configuration Item Name	Device Attribute Name	Possible Value Explanation
Keyboard	**UserAttributes.GetConfigData.Keyboard**	0 = No Keyboard 1 = MC90XX 53 Key 2 = MC90XX 43 Key 4 = MC90XX 53 Key (VT) 8 = MC90XX 53 Key (3270) 16 = MC90XX 53 Key (5250) 32 = MC90XX 28 Key 33 = MC90XX Numeric (3801) 34 = MC90XX Alpha (3802) 35 = MC90XX 53 Key (Custom) 36 = MC90XX 28 Key (Custom) 48 = MC95XX Alphanumeric 49 = MC95XX Telenum Biometrics 50 = MC95XX Telenumeric 51 = MC95XX Legacy numeric 52 = MC95XX Alpha primary 64 = MC90XX 20 Key 76 = PDT81XX 37 Key [TBD] = PDT81XX 47 Key 257 = MC30XX 28 Key 258 = MC30XX 38 Key 259 = MC30XX 48 Key 260 = MC30XX 28 Key (Custom) 513 = MC70XX Numeric 514 = MC70XX QWERTY 769 = MC40X0 Triple Tap 770 = MC40X0 Double Tap 1027 = CA50XX Small 1028 = CA50XX Big 1088 = Stealth [TBD] 1280 = MC17XX 1360 = MC55XX QWERTY 1361 = MC55XX AZERTY 1362 = MC55XX QWERTZ 1363 = MC55XX Numeric 1364 = MC55XX PIM 1440 = MC659B [TBD] 1536 = MK500 1792 = MC75XX Numeric 1793 = MC75XX QWERTY 1794 = MC75XX AZERTY 1795 = MC75XX QWERTZ 2048 = Aspen [TBD] 2304 = ES400 QWERTY 2305 = ES400 AZERTY 2306 = ES400 QWERTZ
RAM Size	**UserAttributes.GetConfigData.RamSize**	Size of installed RAM in MegaBytes (MB)
RFID HF	**UserAttributes.GetConfigData.RfidHF**	[TBD]

Configuration Item Name	Device Attribute Name	Possible Value Explanation
RFID UHF	**UserAttributes.GetConfigData.RfidUHF**	[TBD]
Touch Panel	**UserAttributes.GetConfigData.TouchPanel**	0 = No Touch Panel 1 = TYPE0 [TBD] 2 = TYPE1/PRODUCTION [TBD] 3 = MC17XX AC97 CODEC 4 = Qualcomm
Trigger	**UserAttributes.GetConfigData.Trigger**	Device specific, consult EMDK.
WLAN	**UserAttributes.GetConfigData.WLAN**	00000000 = No WLAN 00000001 = Legacy On-Board WLAN 00000002 = Spectrum 24 FH 00000004 = Spectrum 24 DS 00000008 = Cisco 00000010 = Cardbus Prism 00000020 = Photon BG 00000040 = Photon ABG 00000080 = Jedi BG 00000100 = Jedi ABG 00000200 = Hornet ABG
WWAN	**UserAttributes.GetConfigData.WWAN**	00000000 = No WWAN 00000001 = MC45 GPRS 900 00000002 = CDMA Sprint 00000003 = MC46 GPRS 850 00000004 = CDMA Verizon 00000005 = CDMA Bell Mobility 00000006 = CDMA Western Wireless 00000007 = CDMA Telus 00000008 = CDMA MobileNet 00000009 = CDMA SK TeleCom 0000000A = CDMA NZ Mobile 0000000B = CDMA Telstra 0000000C = CDMA China Unicom 0000000D = Nextel iDEN 0000000E = MC75 0000000F = EVDO 00000010 = iDEN SouthernLinc 00000011 = iDEN KT PowerTel 00000012 = HSDPA HC25 00000013 = EVDO MC5725 00000014 = HSPA MC8790 00000015 = EVDO MC5727 00000016 = HSDPA HC28

Notes:	❶	A Beeper is a special purpose electric to acoustic transducer that is capable of reproducing audio sounds. A Beeper typically has a very limited range of frequency response and hence cannot generally be used to play music or reproduce voice.
	❷	A CODEC is hardware circuit capable of decoding digital audio data. A CODEC is typically assumed to be coupled to a speaker and hence can be used to play music and reproduce voice, although the quality will be limited by the frequency response of the speaker used.
	❸	A Speaker is a general purpose electric to acoustic transducer that is capable of reproducing audio sounds. A Speaker generally has a fairly broad frequency response, the exact range of which will vary according to its intended purpose.
	❹	A Jack is a surface-mounted female electrical socket for attaching an external Speaker and/or Microphone. If a device has an internal Speaker and/or Microphone, attaching an external Speaker and/or Microphone via a Jack will typically also disconnect the internal Speaker and/or Microphone.
	❺	A Mic (short for Microphone) is an acoustic to electrical transducer that converts audio sounds into an electrical signal. A Microphone is typically used for recording audio and/or for reproducing voice, such as in a telephone.
	❻	A Receiver is a Speaker that is specifically designed to be situated close to the ear and is typically intended to be used for reproducing voice, such as in a telephone.

Table 11 – GetConfigData Control Module Device Attributes

GetCabInventory

Operation

The GetCabInventory Control Module utilizes information stored in the Device Registry to determine the installed .CAB Files.

Notes:

It is possible for a .CAB File to be installed by MSP, as part of a Package. It is also possible for a .CAB File to be installed some other out-of-band manner, such as by manual installation by the Device User.

If a .CAB File is installed by MSP as part of a Package, then that Package will show up in the Package Inventory reported to MSP by the device. In addition, the .CAB File would appear in the .CAB File Inventory delivered for the device by the GetCabInventory Control Module. The names of the Package and the .CAB File might or might be similar to each other.

If a .CAB File was installed as part of a Package and that Package is uninstalled, then the .CAB File would generally be uninstalled by the Package and hence both the Package and the .CAB File would be removed from their respective inventories.

If a .CAB File was installed as part of a Package, it is possible for the .CAB File to be uninstalled through some other out-of-band mechanisms, such as by manual removal by the Device User. In such a case, the Package would still show up in the Package Inventory reported to MSP by the device, but the .CAB File would no longer appear in the .CAB File Inventory delivered for the device by the GetCabInventory Control Module.

If a .CAB File was **not** installed as part of a Package, then no Package for it will show up in the Package Inventory reported to MSP by the device. However, the .CAB File would still be part of the .CAB File Inventory delivered for the device by the GetCabInventory Control Module. This could indicate the presence of undesirable or “rogue” applications.

Windows PCs

At this time, there are no Control Modules that only work on Windows PCs or whose operation is substantially different on Windows PCs.

MOTOROLA

Motorola, Inc.
One Motorola Plaza
Holtsville, New York 11742, USA
1-800-927-9626
http://www.symbol.com

72E-128805-03