Wired for Management Baseline

Version 2.0 Release

Specification to help reduce Total Cost of Ownership for business computers.

December 18, 1998
Intel Corporation

Table of Contents

Executive Summary

Wired for Management (WfM) is an Intel initiative to improve the manageability of desktop, mobile, and server systems. Better manageability will add value by decreasing the Total Cost of Ownership (TCO) of business computing while taking advantage of the power and flexibility of high-performance computing.

The Wired for Management Baseline Specification continues the work of the Wired for Management initiative. This Baseline is designed to help manufacturers produce systems that can be effectively managed to reduce customers' support costs. This version of Wired for Management Baseline extends and enhances the interfaces defined by previous versions of the Baseline. Desktop-, mobile-, and server-specific requirements for each area are gathered into platform-specific chapters. By outlining a required minimum level of management capabilities for all desktop, mobile, and server systems, the WfM Baseline lays a standards-based foundation on which manufacturers can build to provide even higher levels of system manageability.

Systems based on the WfM Baseline Specification feature software agents, hardware features, and other capabilities that enhance networked operation and reduce support costs, while taking advantage of the system's flexibility, performance, and compatibility with existing networks. The design guidelines specified in the Baseline enable manufacturers to quickly deliver the integrated capability for central administration, remote network configuration, off-hours maintenance, and constant monitoring of system health.

This version of the WfM Baseline defines the requirements for:

- Instrumentation to ensure that all hardware devices can be recognized and acted on by software, aiding in successful remote troubleshooting and repair.
- Standard interfaces and protocols to allow remote network access and interoperability between different vendor implementations of instrumentation and different management applications.
- Platform agents that allow remote configuration of all system software, from the operating system through drivers and applications, even without a formatted hard disk.
- Power management to allow maintenance of networked systems during off-hour periods.
- Standard trouble ticket transactions and formats to allow systems to participate in trouble ticket exchange with management applications and help desks.

1. Introduction

The Wired for Management Baseline focuses on reducing the cost of managing systems by identifying the key interoperability specifications required to allow any management application to perform required functions on any type of system from any manufacturer.

1.1 Need To Lower Total Cost of Ownership

The combination of personal computers, servers, and networks has proven potent for businesses, as connected systems have come to play a central role in a wide range of mission-critical business applications. But while businesses have enjoyed the increased abilities that networked systems bring to their organizations, their Information Technology (IT) departments have had to deal with the increasing administrative burden and costs of keeping things running smoothly. The proliferation of hardware and software choices, the expansion of system capability and the explosion of the Internet have made the corporate computing environment increasingly complex — and expensive — to deploy and manage. With large environments of mixed computing devices, it becomes increasingly difficult for IT organizations to maintain machines for maximum availability, repair them quickly, and manage them as a vital corporate asset.

The ongoing expense incurred in deploying and managing is a major factor in the Total Cost of Ownership (TCO) of personal computers and servers. Studies by Gartner Group and others use different methods for computing TCO. Some include so-called “soft” costs, such as lost productivity when an employee interrupts the colleague in the next office for help with a system configuration question. Others focus only on “hard” costs, such as administrative tools and salaries of administrative personnel. A typical figure for “soft” TCO is around $8,000 per year; “hard” cost numbers are usually in the $2,000-$3,000 range. Whichever way they choose to compute TCO, IT departments agree that lower costs are needed to keep their businesses competitive.

The Wired for Management Baseline addresses some key factors that contribute to high TCO:

- **Ease of use problems**. The number of supported users and devices has been increasing rapidly. Understandably, many of these new users lack the desire and/or the skills to effectively manage their systems. This causes frequent calls to technical support because end-users cannot resolve even the most basic of questions about their system.
- **Growing demands for availability.** As systems take on more of an organization’s core business functions as well as its communications and productivity applications, the pressure for IT to keep the computing environment available rises. Costs of downtime range from thousands to millions of dollars per hour. Yet a growing number of situations require taking the system down, for example, virus identification and cleaning, installing upgrades, or providing other types of system maintenance. Downtime also occurs because IT may lack needed information on configurations, bandwidth, or usage models.

- **Time-consuming repairs.** The complexity that comes with an open, mixed computing environment increases the time to diagnose, troubleshoot and fix a failure. Users who cannot accurately describe their system or problem increase the number of phone calls to technical support. Support personnel, who often lack the tools to remotely determine the problem or system configuration, must schedule on-site visits. Fixing a failure often requires multiple trips. Most current help desk products do not provide automated problem resolution assistance. As a result, repairs are slow and labor-intensive; support personnel tend to burn out quickly and have high turnover, all of which cause IT costs to rise.
- **Difficulties of asset management.** Asset management is the process of maximizing the use of assets to produce revenue while minimizing their overall costs. When the computing investment is distributed, inventory and tracking information is difficult and time-consuming to gather. Currently it is handled as a manual process — one that is often not accurate. Stories abound of otherwise well-managed companies that find after doing an inventory they have thousands more systems than they thought — or thousands fewer. IT organizations need inventory and configuration data for issues such as server/license consolidation and rationalization, leasing considerations, analyzing effectiveness and costs of training, software upgrade analysis for volume purchasing plans, cost efficiency of outsourcing, warranty usage improvements, and effective management of cascading technology.

1.2 The Solution: Making Systems Universally Manageable and Universally Managed

Fortunately, the connected system has the capacity to lower the cost of management. The “intelligence” of the platform and its connection to the network can enable the system to play an active role in self-management and make it easier for IT organizations to automate and centralize management activities. Today, however, fully managed systems are the exception rather than the rule.

There are two major reasons for this situation:

- While many systems contain features intended to make them manageable, there is no common set of features that are guaranteed to be found in *all* systems.
- Manageability features are typically made available through proprietary means, not open industry specifications.

The result is a tight coupling between management applications and particular platforms: XYZ-brand systems can be managed only by the XYZ management application. In a business with systems from multiple suppliers, this leads to two choices: deploy an unwieldy, complicated collection of management applications that don’t interoperate, or continue to use labor-intensive manual procedures. Both options increase TCO.

The Wired for Management Baseline resolves these shortcomings by defining a targeted minimum set of management features for all platforms. The features are made available through well-defined, and in some cases, industry-standard interface specifications, giving management applications a consistent way to access those features. The goal is a mix-and-match capability among platforms and management applications: any platform that meets the Baseline can be managed by any management application that uses the Baseline’s interface specifications.

Examples of features defined in the Baseline that help efficiently manage systems over networks include:

- A common set of system instrumentation that provides remote management applications access to a common set of platform characteristics and settings in a standard format using standard interface specifications.
- Power management and remote wake-up features that enable off-hours maintenance and management by providing interface specifications for ‘waking up’ a ‘sleeping’ system.
- Platform firmware is defined to provide instrumentation and standard access for configuration and remote booting in an OS absent environment.
- Standard trouble ticketing features that allow systems to automatically report problems and receive solutions.

Note that the Baseline specifies a *minimum* set of management capabilities. Vendors of both platforms and management applications are encouraged to further increase the value of their products by incorporating additional manageability features.

1.3 Target Audience

This document is intended as a reference for:

- System manufacturers, and their hardware and software suppliers, seeking to ensure their systems can be managed by a variety of management applications.
- Management application suppliers seeking a baseline for system features, to which they can design.
- IT professionals seeking to specify system features which will work with a variety of chosen management applications to reduce their overall cost of ownership.

1.4 Document Organization

The remainder of this document describes the Wired for Management Baseline.

Chapters 4 through 6 specify the Baseline functionality for each technology category and describe the interface specifications used to access that functionality.

Chapters 7 through 9 provide checklists describing the specific requirements for each platform type. Chapter 10 is a brief glossary of terms and acronyms. Chapter 11 lists documents that are referenced in this Baseline Specification and provides information about how to obtain them.

Throughout this Baseline, the following definitions apply:

- **Required.** Features that must be implemented to adhere to this Baseline for a particular platform type.
- **Recommended.** Features that support or improve manageability. If a recommended feature is implemented, it must meet the requirements for that feature as defined in this Baseline. Some recommended features may become requirements in the future.

- **[TAG]**: Identifies a document reference. See the associated [TAG] in Chapter 11 for information about how to obtain copies of the document.

1.5 Related Specifications

The WfM Baseline is designed for compatibility with current and emerging industry specifications/guidelines.

1.5.1 ACPI

The WfM Baseline Specification for remote wakeup and power management promotes the use of the Advanced Configuration and Power Interface (ACPI).

1.5.2 DMI, CIM and SNMP

Defining management-related data is a necessary task to enable access and use of this data. For this purpose, the Wired for Management Baseline specification defines the use of one of the following:

- Internet Engineering Task Force’s (IETF) Simple Network Management Protocol (SNMP) and Management Information Base (MIB) modules
- Desktop Management Task Force’s (DMTF’s) Desktop Management Interface (DMI) v2.00 and Management Information Format (MIF) syntax, OR
- Common Information Model (CIM) V2.0 and Management Object Format (MOF) syntax

1.5.3 Network PC Guidelines

The interface specifications in the WfM Baseline for instrumentation, remote wake-up, and remote new system setup are completely compatible with those described in the Network PC Guidelines. Thus, it is possible for the same applications to manage Network PCs and desktops, mobiles, and servers conforming to the current Baseline.

1.5.4 Integration with Enterprise Management

Managing systems in a corporate enterprise might require that these devices integrate with current enterprise management applications. WfM Baseline recommends using SNMP support, in addition to DMI or CIM, in cases where management applications do not support these other standards directly. If SNMP support is provided, the data required by this Baseline must be provided.

1.5.5 SES/SIS

In order for platforms to participate in standardized problem reporting and resolution, this Baseline describes the requirements for an agent that conforms to the Solution Exchange Standard and Service Incident Exchange Standard [SES/SIS], defined by the Customer Support Consortium and the Desktop Management Task Force.

1.5.6 SMBIOS

The SMBIOS specification [SMBIOS] describes system management requirements for BIOS's, including instrumentation.

1.5.7 WBEM

The Baseline is designed for compatibility with current and emerging industry standards. WBEM (Web-Based Enterprise Management) is an industry initiative, that promotes an architecture for object-based management for the enterprise. The initiative encompasses three tenets: 1) a unifying schema for representation of enterprise-wide management data; 2) a common access method to the management data; and 3) integration of existing management standards (SNMP, DMI, CMIP,...). CIM, being defined by the DMTF, provides the unifying schema for WBEM. As Web-Based Enterprise Management (WBEM) technology and specifications become available, the Baseline will also specify the use of WBEM mechanisms, where pertinent.

1.5.8 WMI

The Windows Management Instrumentation (WMI) is a new driver instrumentation interface provided in Windows† 98 and Windows NT† 5.0, as part of Microsoft's WBEM/CIM framework.

1.5.9 Boot Integrity Services

The *Boot Integrity Services Application Programming Interface* [BIS] specifies an interface for verifying the integrity of data using digital signatures, and related functions. These functions are intended for use during OS-absent operation of the platform.

1.5.10 Preboot Execution Environment (PXE) Specification

[PXE] specifies three technologies that establish a common and consistent set of preboot services within the boot firmware of Intel Architecture systems

- A uniform protocol for the client to request the allocation of a network address and subsequently request the download of a Network Bootstrap Program (NBP) from a network boot server.
- A set of APIs available in the preboot firmware environment in the machine that constitute a consistent set of services that can be employed by the NBP or the BIOS.
- A standard method of initiating the preboot firmware to execute the PXE protocol on the client machine.

1.6 Baseline Evolution

The WfM Baseline Specification cannot be static, and will evolve over time. Intel expects major revisions of the Baseline to occur approximately on a yearly basis as platform vendors consolidate the current feature set and additional features win widespread support.

This version of the Baseline focuses on desktop, mobile, and server systems. Subsequent versions may include additional platform types (for example, workstations) and will include WBEM interfaces and specifications as these are developed from the standards bodies.

1.6.1 Wired for Management 2.0 Updates

This version of the Wired for Management Baseline:

- updates the requirements of the Wired for Management 1.1a technologies.
- adds new technologies to the Baseline.
- more fully describes the requirements for different types of platforms.

Specifically, the Power Management and Remote Wakeup sections of Wired for Management 1.1a have been combined into a single Power Management section in Wired for Management 2.0, which:

- requires ACPI.
- (desktop) requires that, if a remote connection is available, a system must be capable of being woken up remotely from its recommended sleep state via one of three described wakeup techniques. Recommended for mobiles and servers.
- recommends a sleep state for each platform type.
- (desktop, mobile) requires bus power management based on the published specification for that bus type. Recommended for servers.

In general, the Wired for Management 2.0 Instrumentation section has been updated to encourage the use of WBEM technology while applying the current requirements more equally across DMI 2.0, SNMP, and WBEM. For instance, where Wired for Management 1.1a encouraged the use of DMI events and required that, if events were generated, the format comply to the DMI format; Wired for Management 2.0 requires that if events are generated, they be compliant to the active management framework. In summary, Wired for Management 2.0:

- requires that one of three management frameworks (DMI 2.0, SNMP, WBEM) must be installed and active on the system.
- requires that if a dynamic device is present, its instrumentation must also be present.
- addresses specific backward compatibility issues with Wired for Management 1.1a instrumentation and cross compatibility issues between the different management frameworks have been addressed.

The Wired for Management 1.1a Remote New System Setup has been expanded into the Platform Firmware section in Wired for Management 2.0, which:

- (desktop, mobile) requires that the PreBoot Execution Environment (PXE), now described in a separate PXE 1.0 specification be available. Recommended for mobiles which do not have LAN on motherboard and for servers. PXE 1.0 also includes a new, backward compatible split ROM architecture and more secure APIs.
- requires SM BIOS 2.2 or later, with tables populated by the specified required data.
- requires that a reason code for system startup be available.

- requires that hardware which allows a local lockout must also allow remote lockout.
- recommends that SNMP traps be used when the OS is absent to report events (which format to use is described in a separate specification).
- recommends that the Boot Integrity Services (described in a separate specification) be implemented.

Additionally, Wired for Management 2.0 recommends the use of standard trouble tickets, requiring that systems which generate trouble tickets do so in a standard manner.

2. Overview

The Wired for Management Baseline reduces system management costs by addressing requirements from the following three perspectives:

- Platforms. It describes those things which a system requires to be managed; for example, hardware failure points.
- Management Application. It provides information and access methods that a management application requires to be consistently present to perform its functions; for example, access to system characterization and identification.
- User. It identifies those procedures that need to be present in order to ensure that management occurs in a user-friendly manner; for example, resuming (not rebooting) systems upon wakeup.

These requirements are applied to the interfaces and information available from a particular system. This allows a Baseline system to be managed by a variety of management applications built to these interfaces, thus providing a minimum management cost reduction regardless of system manufacturer or application supplier.

2.1 Concepts

The features defined by the Baseline apply to several management environments or scopes:

- Enterprise. This usually refers to a “large” number of systems, connected by multiple LANs, WANs, and potentially many sites.
- Work group. This is a “small” group of systems, possibly located geographically close, potentially on the same LAN or a small number of LANs.
- Individual. This refers to the management of a single system. This may be an out-of-office mobile system that is managed by a corporate help desk, a consumer system that can be managed by a contract or OEM service, or a single server located in a branch office of a larger enterprise.

A Baseline system will make its defined interfaces available regardless of the management environment, and any level of management application can make use of them. All of these management environments are abstracted by this Baseline into **managed client**’s and **management server**’s. Any managed system is a managed client. In an enterprise or work group environment, the servers themselves are managed clients. A management application may reside on a management server or the managed client. There is no assumption that all management applications reside on the same management server.

Additionally, for a given system, several common system characteristics and environments must be taken into consideration by all of the Baseline features.

2.1.1 Guidelines for referring to products based on the requirements contained in this reference specification

Wired for Management 2.0 **enabled** systems, subsystems, or components (such as network adapter, hard disk drive, or power supply) ship with all hardware and software elements required by this Baseline for the platform or component type.

Systems, subsystems, and components are Wired for Management 2.0 **upgradeable** when the system, subsystem, or component can be easily brought to **enabled** status when a customer or integrator adds the missing Wired for Management 2.0 support (for example, by adding a NIC, an option ROM, DMI software, and so on) through easily obtainable means (for example, web downloadable, 1-800 number, or sending in a postcard).

Management applications are Wired for Management 2.0 compatible when the software includes provisions for managing WfM-enabled systems using one or more of the management functions specified by this Baseline.

2.1.2 Platform Types

This Baseline addresses three types of platforms:

- Desktop. Exemplified as the stationary single-user business or consumer system.
- Mobile. A portable system primarily distinguished by its battery-powered operation and roaming, occasionally-connected nature.
- Server. A system which provides services to other clients, for example, printing or shared file handling. In the context of specifying requirements for managed systems, this Baseline describes requirements which affect the server when it is the target of management operations. In these cases, the server is a *managed client* of the management environment defined above.

The Baseline describes the general requirements for a particular capability in its main chapters and then makes specific requirements as necessary for each platform type. More detailed definitions of these types and a summarized checklist of their specific Baseline requirements are given in the platform checklist chapters.

2.1.3 OS-Present vs. OS-Absent

Platforms must be manageable both before and after an OS is loaded. One goal of this Baseline is to provide a useful and consistent subset of management functions when the OS is not present. These functions are to be used if there are problems loading the OS, or for updates (for example, BIOS) which can not be made when the OS is present. For more advanced capabilities the OS must be loaded and, in fact, not all technologies are available when the OS is absent.

2.1.4 Occasionally-Connected Systems

Many existing management systems assume the systems they manage are always available to be managed if they are healthy. However, it is the normal condition for mobiles to be disconnected from a communication link. This is also true for some desktops and servers in a small business or home environment.

Taking the occasionally-connected system into consideration means some conceptually simple changes, such as queuing requests until a connection exists. It also means some more subtle considerations, such as handling the relationship to platform security—which has a different definition for mobiles which are inherently theft-prone.

2.1.5 Dynamic Peripherals

Traditionally, desktop systems have assumed that installed peripherals are always connected to the system (if they are healthy). Of course, mobile and server systems have not made the same assumption. Additionally, the advent into the desktop of dynamic buses, such as the Universal Serial Bus and 1394, makes dynamic peripherals a concern of all platform types.

Dynamic peripherals impact asset management, security, and instrumentation. For example, it is not generally workable for a system manufacturer to know *a priori* all of the managed peripherals that will be used with the system. Therefore, instrumentation can not be created only at system design time but must be able to handle the attachment and removal of devices.

2.1.6 Power Management Environment

To reduce power consumption, more and more systems are being power-managed into lower power states. However, not all of the system's management functions can be made available in such states. Therefore, consideration must be given to which functions are reasonably available in low power states, how functions react as the system transitions between power states, and how the system returns to a higher power state which provides the full set of management capabilities.

2.2 Baseline Technologies

The WfM Baseline addresses requirements in four technologies. Wherever possible, existing industry standards are identified for use. However, some technologies do not have adequate public definition, and additional detailed specifications for such technologies are in the appendices. The definition of each technology is summarized below. For more details, consult the chapter describing the technology.

2.2.1 Power Management

Power management allows a system to consume less power, but still be fully operational in a short period of time. This enables low power consumption and improved end-user experience when starting up (really 'waking up') a system.

To minimize the downtime end-users experience for system maintenance and upgrades, a manageable system must provide the ability to be automatically managed during off-hours, when the system is not otherwise in use.

The WfM Baseline specifies that it must be possible to remotely wake up a managed system from its sleep state. It further recommends a target sleep state based on platform type.

2.2.2 Instrumentation

To begin reducing TCO, management applications must have complete programmatic access to the state, control, and descriptive parameters of the system and its component subsystems. A manageable system with instrumentation provides such access for both local and remote applications. In addition, the instrumentation infrastructure provides the means by which the system and its components can report state changes and unusual conditions autonomously (that is, without being specifically queried by a management application).

To derive the full benefits of instrumentation, it must be possible to manage any system from any management application. This is possible when the data is provided in standard format through standard interfaces.

The WfM Baseline specifies standard data formats and standard remoteable interfaces for instrumentation and defines the minimum set of data that a managed system must supply through its instrumentation. This ensures the availability of information for:

- Problem detection and notification of the appropriate support organization.
- Problem diagnosis and correction.
- Tracking or logging the failure information.
- Computer equipment inventory.

By streamlining these processes, instrumentation increases a system's ease of use, reduces the labor costs involved in maintaining and supporting systems, and increases the IT organization's ability to administer and manage its business computing assets.

2.2.3 Preboot Execution

The Preboot Execution Environment (PXE) is a protocol and interfaces by which agents can be loaded remotely onto the client to perform management tasks in the absence of a running OS. The Preboot Execution Environment enables the automation of a number of management tasks. Examples include:

- Initial configuration of new machines.
- Diagnosis of problems that prevent the operating system from functioning correctly.
- Configuration updates prior to booting the operating system.

The remote operations enabled by the Preboot Execution Environment can lower the costs of administration and technical support, thus decreasing TCO.

2.2.4 Problem Resolution

Today, every management environment must have some form of trouble ticketing. However, a particular strategy usually only works with explicit cross-developments by individual vendors. This means that management applications usually cannot obtain timely and accurate information about specific problems because the information is transient and not when the problem occurred. This may also be because the only interface available to the management application (or help desk) is a human user who may be expressing an opinion about what was present on the system at the time the problem occurred.

In addition to being a lossy operation, manual trouble ticketing is also unnecessarily consumptive of elapsed time due to the required manual intervention, thus resulting in more down time.

The standard trouble tickets recommended by this Baseline do not, by themselves, eliminate the need for highly skilled technical personnel involved in problem-solving. But by identifying standard formats and transactions for a platform agent, it allows the automatic and accurate capture and forwarding of problem-solving information.

3. Power Management

Power management contributes to lower TCO while enhancing user experience. The need for power management is underscored by the requirement for performing remote off-hours management in a manner which does not decrease system life. In addition, new regulations and labeling programs are mandating tighter restrictions on platform power consumption. For example, Energy Star requires that systems automatically power down to 30W or less when not in use. Other programs call for lower power consumption than the current 30W limit, with some specifying 5W or less.

Properly addressing power consumption restrictions leads to requirements in two areas:

- Power-managed systems. This includes the state and state transition definitions necessary to appropriately describe power-managed states and the criteria under which they are entered. This Baseline describes a power-managed system using the Advanced Configuration and Power Interface (ACPI).
- Off-hours management. This includes the ability, commonly referred to as “Remote Wakeup”, for a system to be in a low power state during off-hours and still be available to be managed upon the request of a remote management server.

ACPI is an interface between the OS and the hardware and BIOS designed to achieve independence between the hardware and the OS. The ACPI Tables, which describe a particular platform’s hardware, are at the heart of the ACPI implementation and the role of the ACPI BIOS is primarily to supply the ACPI Tables (rather than an API). ACPI provides the opportunity to integrate the interface for controlling power management and Plug-and-Play features on system devices.

3.1 Requirements

The following sections summarize the power management requirements on Baseline systems. The section following discusses in more detail the exact nature of the requirements.

3.1.1 Platform Requirements

1. *ACPI-compliant*. Platforms must use ACPI-compliant motherboard components and BIOS, and thus be capable of being power-managed by the operating system. Legacy support for Advanced Power Management (APM) BIOS is not addressed by this specification.
2. *Recommended Sleep Mode*. At least one ACPI sleep mode should be implemented in the S1 to S3 range.

3.1.2 Waking up a System Remotely

3. *Remote Wakeup*. It must be possible to remotely wake the system for the purpose of remote management from at least one reduced power state, which may be either a Sleep State (S1–S4) or a Soft Off state (S5).. This must be implemented via one of the techniques described in section 3.1.3. Waking up the system should be as non-intrusive for the system user as possible, and it is desirable that the system wake up without requiring a reboot.

3.1.3 Remote Wakeup Techniques

4. *Magic Packet*. This technology uses a small circuit in the LAN adapter to listen to a unique frame when the rest of the system is asleep or off. A management application builds and sends the Magic Packet to the target machine.
5. *Packet Filtering*. With this technology, the LAN adapter is programmed to recognize several types of frames, depending on the protocols currently loaded by the OS, and to wake up the system if one of these frames is sent to the target machine. Any application can wake up a sleeping machine.
6. *Wake on Ring.* An incoming call on a modem wakes up a sleeping system.

3.1.4 Bus Requirements

7. *Bus Power Management*. If a given type of bus is implemented in a system, the devices attached to that bus and the system in general must follow the power management requirements for that particular bus type. For PCI devices this means that add-in adapters must be PCI-PM compliant. It is strongly recommended that both device and system vendors implement the 3.3Vaux power plane (PCI pin 14A) as defined by the "PCI Engineering Change Request - Addition of 3.3Vaux signal to Connector." The power management requirements for a given bus type (for example, PCI; PC Cards; USB; 1394) are given in its bus specification.

3.2 Power-managed Systems

This Baseline requires that a platform use ACPI-compliant motherboard components and BIOS, and be "power management ready". Additionally, all PCI add-in functions, such as network interface adapters, must comply with the PCI Bus Power Management Interface Specification [PCI-PM]. Support for device functionality in the $D3_{cold}$ state may be accomplished by implementing the "PCI Engineering Change Request - Addition of 3.3Vaux signal to Connector" or by an alternate power supply method. The [PCI-PM] specification serves to extend the benefits of an ACPI platform to also include PCI add-in adapters. It is strongly recommended that an ACPI-enabled operating system (for example, Windows NT 5.0) power-manages the system to reduce power consumption when the system is not in use.

For context and terminology purposes, ACPI is briefly summarized below; however, for complete details, refer to the actual specification [ACPI].

For Operating System Directed Power Management (OSPM) platform implementation details and tradeoffs, refer to the Instantly Available PC Power Management Design Guide [PC POWER].

3.2.1 ACPI Summary

ACPI defines the interfaces between the operating system software, the hardware, and BIOS software. It also defines the semantics of these interfaces. The term **operating system directed power management** (OSPM) refers to an operating system that uses the ACPI interfaces to execute power management policies.

Figure 3-1 shows the software and hardware components relevant to ACPI and how they relate to each other. ACPI describes the *interfaces between* components, the contents of the ACPI Tables, and the related semantics of the other ACPI components. Note that the ACPI Tables, which describe a particular platform's hardware, are at the heart of the ACPI implementation, and the role of the ACPI BIOS is primarily to supply the ACPI Tables (rather than an API).

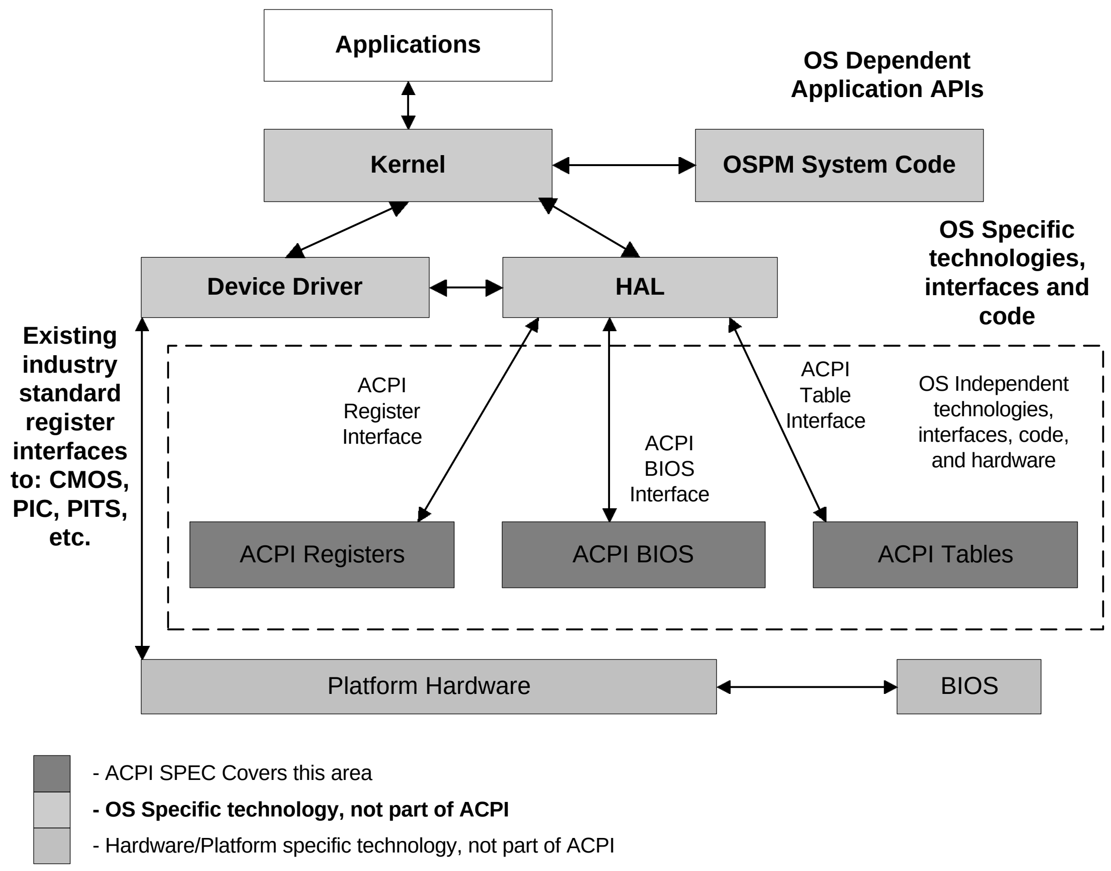
Applications
OS Dependent Application APIs
Kernel
OSPM System Code
OS Specific technologies, interfaces and code
Device Driver
HAL
Existing industry standard register interfaces to: CMOS, PIC, PITS, etc.
ACPI Register Interface
ACPI BIOS Interface
ACPI Table Interface
OS Independent technologies, interfaces, code, and hardware
ACPI Registers
ACPI BIOS
ACPI Tables
Platform Hardware
BIOS
- ACPI SPEC Covers this area
- OS Specific technology, not part of ACPI
- Hardware/Platform specific technology, not part of ACPI

Figure 3-1 OSPM/ACPI Global System

3.2.1.1 ACPI Description

ACPI has three runtime components:

1. ACPI Tables. These tables describe the interfaces to the hardware. These descriptions allow the hardware to be built in flexible ways, and can describe arbitrary operation sequences needed to make the hardware function. ACPI Tables may contain p-code language, the interpretation of which is performed by the OS.

2. ACPI Registers. The constrained part of the hardware interface, described (at least in location) by the ACPI Tables.
3. ACPI BIOS. Refers to the portion of the firmware that is compatible with the ACPI specification requirements. Typically, this is the code that boots the machine (as legacy BIOSs have done) and implements interfaces for suspend, resume, and some restart operations. The ACPI Description Tables are also provided by the ACPI BIOS. Note that in Figure 3-1, the boxes labeled “BIOS” and “ACPI BIOS” refer to the same component on a platform. The box labeled “ACPI BIOS” is broken out to emphasize that a portion of the BIOS is compatible with the ACPI specification requirements.

3.2.1.2 Power States

Baseline systems have four distinct power states: Working, Sleeping, Soft Off, and Mechanical Off. In the Working state, user mode application threads are dispatched and running. Individual devices and processors may be in low-power states if they are not being used. Any device the system turns off because it is not actively in use can be turned on with short latency. (The meaning of “short” depends on the device. An LCD display needs to come on in sub-second times, while it is generally acceptable to wait a few seconds for a printer to wake up.) When the computer is idle or the user has pressed the power button, the OS will put the computer into a reduced power state, Sleep (S1–S4) or Soft Off (S5). No user-visible computation occurs in a reduced power state. The different power states provide varying degrees of wake-up latency. For example, the Sleep states (S1–S4) save different levels of context, while a Soft Off state requires a complete system reboot.

Both Sleep and Soft Off states consume very little power. In Soft Off only the Real Time Clock and Standby power plane devices; such as the Power Button, portions of the ACPI controller, PCI 3.3Vaux devices, etc.; consume power.. Sleep states, like S3, consume slightly more power to support devices which retain system context or require lower latency transitions to the Working State..

The power button on the front of a Baseline machine is redefined to be a “soft” button, which will normally *not* turn the machine physically off, but will put it in a Soft Off or Sleeping state instead. Unlike today’s on/off button, the soft on/off button sends a request to the system. What the system does with this request depends on the policy issues derived from user preferences, user function requests, and application data.

This specification requires that a system be capable of being remotely awakened, which may occur from either a Sleep state (S1–S4), a Soft Off state (S5), or both. At least one of these reduced power states must support a transition to the Working State via one of the Remote Wake-up Techniques described in Section 3.1.3. This state would then be the power state of choice for assertion by a power button, which would enable after-hours maintenance via remote wake-up, although this is not required.

3.2.2 Platform Power Management Characteristics

Systems conforming to the WfM Baseline have the capability to be managed during off hours by having different modes for Active and Idle:

- Active Mode. In active mode, clients are in the Working state, although they may put unused devices into low power states whenever possible. OS-driven power management allows careful tuning of when to do this.

- Idle Mode. In idle mode, clients sleep as deeply as they can sleep but are still able to wake up and answer service requests coming in over the network, phone links, etc., within required latencies.

3.3 Off-hours Management

If a system supports a reduced power state, it must be possible to bring the system to a fully powered state in which all management interfaces are available. For the purposes of off-hours management this is done by a communications peripheral (for example, LAN Adapter or modem) recognizing that a wakeup has been requested. For example, the LAN adapter may recognize a special packet as a signal to wake up the system.

When waking up using LAN communications, it is recommended that the LAN Adapter and system use LAN packet filtering as a trigger. Packet filtering enables the system to recognize, while in a reduced power state, that it is being addressed using normal LAN communications. No special packets or other means are required.

A more detailed description of packet filtering as implemented in Windows NT 5.0 and Windows 98, based on matching patterns specified by the local networking software, is described in the *Network Device Class Power Management Reference Specification, Version 1.0* [NDC PM] or higher. This applies specifically to Ethernet and token ring adapters. [NDC PM] does not support ATM and ISDN adapters.

For additional packet filtering implementation guidelines, see the *PC 97 Hardware Design Guide* [PC HDG] from Microsoft. Implementation details are described in [NDC PM], also from Microsoft.

However, at the time of this writing, not all system and LAN adapter combinations are capable of packet filtering. Therefore, at a minimum it is recommended that wake events be triggered by the reception of a Magic Packet as an alternative means for waking the system. For information on where this technology is described, see Chapter 11 of this document.

When using a modem as the communications interface, waking the system whenever an incoming call is detected (“wake on ring”) is sufficient.

Once a remote wakeup event is detected through packet filtering, Magic Packet or wake on ring, the system resumes its previous Working state and is available for remote management. This may also be accomplished by rebooting the system from its low-power state; however, resuming is the preferred approach. For hardware implementations that don’t support a transition from the system’s current low-power state to its full-power state while preserving context, rebooting is the only viable solution.

For the purposes of this Baseline, when a system is resumed for remote management, it is required that it be in a state such that it is available for service and management from the network. This system state has the following characteristics:

- The platform’s power state is full ON (Working).
- Peripherals, such as the monitor, that are attached to the platform but are not required for the external management activity, need not be fully powered.
- All management interfaces provided by the loaded environment are available.

3.4 Considerations

This section discusses special considerations for implementing power management and remote wakeup in particular environments addressed by this Baseline.

3.4.1 OS-Absent Environment

Although Baseline platforms are capable of being power-managed independent of the loaded operating system, it is not required that a platform actually enter a reduced-power state in the absence of an operating system. If, however, the platform *does* enter such a state in the absence of an operating system, it must be capable of being awakened by a communications event. When awakened in this manner, all the management interfaces that would normally be present in an OS-absent environment must be available for use.

3.4.2 Desktop Platforms

It is required that desktop systems be ACPI-compliant, and that they respond to remote wakeup events over the LAN for off-hours management. Therefore, off-hours power settings should select the lowest possible power state that supports Remote Wake-up.

3.4.3 Mobile Platforms

It is the nature of mobile systems to be power-managed. It is recommended that they also be capable of being awakened by a communications event if they are connected to a communications link. However, a mobile system may not have either a LAN Adapter or modem available for use. Additionally, the policy for the system may be that such communications peripherals are not powered sufficiently during a power-reduced state, especially when the system is battery powered, to recognize a communications event.

It is acceptable for a manufacturer's implementation of remote wakeup to require that the notebook be on AC power. The decision to require a connection to AC power while waiting for a remote wakeup signal is the manufacturer's choice. Such a restriction is allowed but not required by the WfM Baseline.

3.4.4 Server Platforms

Note: ACPI compliance is not required for servers when they are executing operating systems from vendors other than Microsoft.

The *Hardware Design Guide Version 1.0* (or later) *for Microsoft Windows NT Server* [HDG] establishes requirements for vendors who build server systems, expansion cards, and peripheral devices that use the Microsoft Windows NT Server operating system. The [HDG] includes detailed requirements on how ACPI must be implemented on server systems. In the [HDG], three classes of servers are defined, and ACPI requirements vary by class of server. All Wired for Management servers must comply with the basic server Configuration and Power Management requirements. In addition, Small Office/Home Office (SOHO) and Enterprise servers must conform to the additional requirements associated with configuration and power management for these classes of servers, as defined in the HDG Version 1.0 or later.

Specification of a desired Idle Mode for servers is complicated by the potential complexity of server configurations and the nature of the client-server computing paradigm. Servers, especially departmental and enterprise class servers, may take significantly longer than clients to restore both hardware and applications to the Working state. The selection of an ACPI sleep state for any specific server is determined by the latencies that can be tolerated by the software running on its clients; the server must wake up and respond to the client's request before the client times out and assumes the server is out-of-service. This latency tolerance varies widely among client-server applications and middleware. When packet filtering is used to implement remote wakeup, application-specific latencies must be taken into account by the OS power-management policies before transitioning the server from the Working state to one of the several sleep or Soft-Off states. Because Magic Packet is not a general purpose solution to remote wakeup, the management application who send Magic Packets can be tailored to tolerate longer latencies when trying to manage servers. In the S0 (Working) state it is still possible to achieve power savings by powering off individual devices. Therefore, the desired Idle Mode for servers can be any of the sleep states (S1-S4) plus S0.

3.5 Summary of Platform Requirements

Table 3-1 Power Management Requirements Summary

Ref #	*Area*	*Desktop*	*Mobile*	*Server*
pm01	ACPI-compliant	Required	Required	Required as defined in [HDG]
pm02	Recommended Sleep Mode	S3	S3	S1
pm03	Remote Wakeup	Required	Recommended	Recommended
pm04	Magic Packet	Required	Recommended	Recommended
pm05	Packet Filtering	Recommended	Recommended	Recommended
pm06	Wake On Ring	Recommended	Recommended	Recommended
pm07	Bus Power Management	Required	Required	Recommended

4. Instrumentation Requirements

Instrumentation is a common methodology and syntax for defining the management features and capabilities of all hardware, software, and attached peripherals of desktop, mobile and server platforms. Instrumentation allows management applications to view and alter the state of a managed platform and to monitor (be notified of changes in) the state. Industry-standard instrumentation makes it possible for any Baseline platform to be managed remotely by any management application, regardless of the vendor or operating system.

To fulfill the promise of universally managed platforms, the WfM Baseline establishes instrumentation requirements for the Desktop Management Interface [DMI], Simple Network Management Protocol [SNMP], and Common Information Model [CIM] standards.

These requirements are established in three areas for:

1. Instrumentation Code. Platform specific and OS specific software that provides the management data and events.
2. Management Data and Events. The set of data and events used to describe and manage management components. This Baseline defines a minimum set of management data.
3. Management Framework and Agents. Provides means for management applications to access the management data and for instrumentation to make the management data available.

4.1 Requirements

The following enumerates the Baseline instrumentation requirements. As discussed in the Baseline Instrumentation section that follows, this Baseline takes into account a variety of instrumentation strategies. The list below enumerates the requirements for current platforms. Refer to the Desktop, Mobile, and Server Checklists for how these requirements apply to each platform type.

1. *Management Framework (DMI Version 2.0 Service Provider, SNMP Agent or WBEM Framework) Installed and Active*. The DMI Version 2.0 Service Provider must be installed and active to access DMI instrumentation. Similarly, an SNMP Agent may be required. For Microsoft Windows NT 5.0, Windows 98 and some later versions of NT4.0 and Windows 95 systems, the WBEM framework based on the CIM Schema is available for management. If multiple Management Frameworks are installed and active, at least one of the frameworks must support all of the instrumentation requirements defined in this chapter relevant to the supported framework; that is, a complete Wired for Management 2.0 instrumentation stack as defined in Section 4.3, 4.4, or 4.5 must be available within a single framework.
2. *Local (to the Platform) and Remote Access to Management Data Provided via Standard Access Mechanisms*. Platform management is a mix of local and remote

monitoring, reporting, and control. Capabilities must exist for either local (to the platform) or remote management applications to access and modify management data, via standard and well-defined access mechanisms specific to the Management Framework which is installed and active. For example, for DMI, the local access mechanisms are standard APIs, while remote access is via RPC (Remote Procedure Calls).

3. *Events Generated according to Standard Models (DMI Events, SNMP Traps or WBEM/CIM Events)*. Platforms are recommended to be able to generate management events under the appropriate circumstances. Such events must conform to the DMI Event, SNMP Trap, or WBEM/CIM Models, as appropriate to the Management Framework which is installed and active.
4. *DMI Events Conform to DMI Event Model.* If DMI events are generated, these events must conform to the DMI event model.
5. *WBEM Events Conform to the CIM Event Model. (Future)* WBEM events can be generated and filtered for Microsoft Windows systems, where the WBEM Framework is installed. These events report changes in the CIM Schema—the creation, deletion or modification of classes and properties. In the future, CIM will define standard event classes, similar to (and extending) the DMI Event Groups.
6. *SNMP Traps conform to DMTF SNMP to DMI Mapping Standard [SNMP to DMI].* If the instrumentation generates SNMP traps corresponding to the event generation groups in the required management data, the traps must conform to the event-to-trap mapping methodology described in [SNMP to DMI] (see platform checklists). Vendors supporting legacy SNMP instrumentation stacks are not required to meet this requirement provided they support an alternative WFM 2.0 Management Framework (CIM or DMI).
7. *Instrumentation Supports Dynamic Devices.* If a device is present and instrumented, its instrumentation must be available. In other words, the platform instrumentation implementation must account for peripherals that may be dynamically attached and removed.
8. *Instrumentation Deployed and Maintained with Product and Platform*. The instrumentation must be deployed with a product and the platform. As maintenance and Field Replaceable Units (FRUs) are applied to the products and platform, the instrumentation must be kept current with the configuration and capabilities of the system and its devices. It is recommended that CIM-based instrumentation be provided on platforms and operating systems that support it. Instrumentation is expected to be provided by the platform vendor where the instrumentation applies to the platform, and by the add-in adapter or peripheral vendor for their products.
9. *Management Data Available*. The management data defined by the appropriate platform checklist must be available.
10. *Backward Compatibility with the WfM 1.1 Standard for Data and Events.* As the managed environment adopts new standards, such as CIM, it is required that existing applications continue to function until they can be migrated or upgraded. For this purpose, it is required that management data be visible as specified in the WfM 1.1 Specification (data visible via DMI in the Desktop and Mobile environments, and via DMI or SNMP in the Server environment). One mechanism for achieving this requirement is the use of

"cross-mappers". Refer to Section 4.6 for more specific information on cross-mappers between DMI and WBEM/CIM.

4.2 General Instrumentation Requirements

The Baseline does not define how instrumentation is implemented beneath the required interfaces or what sources or methods are used to extract management information from the platform. The Baseline also does not prohibit vendors of platforms or platform components from extending the manageability of the platform by instrumenting additional management data or features.

The specific requirements for instrumentation are based on platform type and are defined in the Desktop, Mobile, and Server Checklists. These checklists contain the platform specific requirements and the baseline management data that must be instrumented and deployed on Baseline platforms.

4.2.1 Management Data

To ensure a baseline of functionality for management applications, a guaranteed minimum set of management data is required. This data is defined in terms of DMTF groups, SNMP OIDs (Object Identifiers), and CIM objects. The specific requirements for management data are based on platform type and are defined in the Desktop, Mobile, and Server Checklists.

4.2.2 Dynamic Devices

This Baseline requires that instrumentation support devices that can be inserted and removed (for example, PC Card, USB, hot swap drives, 1394). Upon such insertion or removal, instrumentation must represent the associated management data for the devices. The appropriate data must be inserted or removed from the management data store and/or framework.

4.2.3 Deployment

The management framework, agents, instrumentation, and associated drivers are platform- and OS-dependent. They must be deployed by the platform vendor along with the platform or its operating system and made active when the operating system is booted. This enables any management application to access and manage the platform via its instrumentation as soon as the platform is up and running.

4.3 Desktop Management Interface (DMI)

4.3.1 Management Data

The Desktop, Mobile, and Server Checklists contain the standard DMTF groups that must be instrumented and deployed on Baseline platforms. These requirements are consistent with the *Desktop Management Interface (DMI) 2.0 Conformance Guidelines*, Version 1.0 [DMI Conform].

DMTF groups are versioned. By design, each new version of a given DMI group is backward compatible. This Baseline requires the specified required group, or a later DMTF approved version of the specified group be implemented. These groups, as defined in the platform Checklists, provide the basic set of data for managing WfM Baseline machines.

4.3.2 Management Agent

The Management Agent required for DMI support in this Baseline is the DMI Version 2.00 Service Provider, as specified in [DMI]. A Baseline platform supporting DMI must deploy a DMI Version 2.00 Service Provider and have it active when the operating system boots. This enables management applications to access and manage the platform via its instrumentation as soon as the platform is up and running.

The Service Provider implementation is OS-dependent. The Service Provider can be deployed by the operating system vendor as part of the operating system or by the platform vendor as a separate software component installed in conjunction with the operating system.

4.3.3 Instrumentation Code

This Baseline requires that platforms provide instrumentation that delivers the management data identified in the appropriate platform checklist. If the instrumentation is DMI-based instrumentation, it must utilize the CI interface provided by the DMI v2.00 Service Provider.

Hardware and software vendors, if implementing DMI, are strongly encouraged to implement to the procedural CI defined in [DMI]. See the [DMI Conform] regarding backwards compatibility for existing instrumentation implemented using the DMI Version 1.x block interface.

4.3.4 Events

Some of the standard DMI groups listed in the platform checklists are associated with event generation groups. Event generation for these groups is optional. However, if the instrumentation generates events associated with a required group, event generation must be compliant with the event model specification defined by [DMI].

4.3.5 Guide to DMI Instrumentation Development

The DMTF supplies a set of guidelines to help hardware and software vendors of add instrumentation to their products. *Desktop Management Task Force: Enabling your product for manageability with MIF files, Revision 1.0* [MIF Guidelines] is available from the DMTF.

4.3.6 Local and Remote Access

The required DMI local access mechanisms are standard APIs, as specified in [DMI]. Remote access is via RPC (Remote Procedure Calls) are also defined in that Specification.

4.4 Simple Network Management Protocol (SNMP)

4.4.1 Management Data

The same set of management data, defined in the form of DMTF standard groups in the platform checklists, is required when delivered through SNMP. Refer to Attachment C, *WfM DMI Mapping to CIM and SNMP*, for information on the mappings between these standards. To ensure a consistent MIB representation of the management data, this Baseline requires the data to be represented as a set of SNMP Object Identifiers (OIDs) as defined by the [SNMP to DMI] standard, available from the DMTF. This mapping provides a consistent representation of the MIF data in the SNMP namespace. Only those groups required by the relevant platform checklist must be supported in the SNMP namespace.

4.4.2 Management Agent

If SNMP support is provided on the managed platform, an SNMP v1 (or later) Agent must be loaded and active when the operating system boots. This enables any management application to access and manage the platform via its instrumentation as soon as the platform is up and running.

The SNMP Agent implementation is OS-dependent. The SNMP Agent can be deployed by the operating system vendor as part of the operating system or by the platform vendor as a separate software component installed in conjunction with the operating system.

4.4.3 Instrumentation Code

This Baseline requires that platforms provide instrumentation that delivers the management data identified in the appropriate platform checklist.

There is no requirement for platforms that export management data via SNMP to implement native SNMP instrumentation. For example, instrumentation may be DMI-based with appropriate SNMP mappings, for use by SNMP applications.

If the instrumentation is SNMP-based, instrumentation code for SNMP extension agents is specific to the OS-specific SNMP Agent. Refer to the appropriate documentation included with the SNMP Agent for information on developing the instrumentation to deliver the required management data.

4.4.4 Traps

Some of the standard DMI groups listed in the platform checklists are associated with event generation groups. Event or trap generation for these groups is optional. If the instrumentation generates SNMP traps corresponding to the event generation groups in the required management data, the traps must conform to the event-to-trap mapping methodology described in [SNMP to DMI] standard.

4.4.5 Local and Remote Access

The required SNMP access mechanisms are defined by the IETF for SNMP v1 or later, primarily in RFCs 1157, 1592, 1905, and 2272.

4.5 Common Information Model (CIM)

CIM is an object-oriented schema for management which the industry is standardizing through the efforts of the DMTF. Object Managers using CIM will be designed with the ability to integrate and publish management data through a number of instrumentation means such as the DMI, SNMP, and others.

One of the first implementations and uses of CIM will be the Microsoft WBEM Framework, available for Windows 98, NT 5.0, and some later versions of NT 4.0 and Windows 95. This section addresses both the CIM Schema and the Microsoft WBEM Framework, since an implementation is needed to realize the CIM information model.

4.5.1 Management Data

The same set of management data, defined in the form of DMI standard groups in the platform checklists, is required when delivered through a CIM data repository. Refer to Attachment C, *WfM DMI Mapping to CIM and SNMP*, for information on the mappings between these standards.

4.5.2 Management Framework

The WBEM Management Framework uses a three-tiered approach for collecting and providing management data. This approach consists of:

- A single API through which all management information can be accessed.
- An information schema based on CIM and extended through Win32 classes.
- A provider architecture which allows population of the information model from the underlying instrumentation.

A WBEM/CIM implementation is OS-dependent. The Microsoft implementation of WBEM is deployed with its operating systems. Other vendors can choose to deploy a CIM object manager as part of an operating system or as a separate software component installed in conjunction with the operating system.

4.5.3 Instrumentation Code

This Baseline requires that platforms provide instrumentation that delivers the management data identified in the appropriate platform checklist. Within the Microsoft WBEM Framework, data from various instrumentation sources (supporting DMI, SNMP, WMI, and other sources) are supplied through “Providers” to the CIM data repository. Several standard Providers are available to input Registry, Event Log, SMBIOS, WMI, DMI, and other data to the CIM information model.

4.5.4 Events

Some of the standard DMI groups listed in the platform checklists are associated with event generation groups. Event generation for these groups is optional.

WBEM events can be generated and filtered for Microsoft Windows systems, where the WBEM Framework is installed. These events report changes in the CIM Schema – the creation, deletion or modification of classes and properties. Model changes can be the direct result of hardware errors and other events detected by drivers or instrumentation code. In the future, CIM will define standard event classes, similar to (and extending) the DMI Event groups.

4.5.5 Local and Remote Access

The local and remote access mechanisms for WBEM/CIM for Microsoft Windows NT 5.0 and Windows 98 environments are defined in the Microsoft online documentation, in particular, the *WBEM SDK*. In the future, the DMTF may address the definition of a secure wire protocol, standard methods and queries, and APIs to extend the CIM standard.

4.6 Cross-Mapping Strategies

Instrumentation or management applications may need to provide data, or access data, across multiple standards (DMI, SNMP, or CIM). In these cases, “cross-mappers” are recommended to be installed and active. A cross-mapper is code that allows management data to be visible through a variety of management technologies and standards, regardless of the source or destination of that data.

One scenario where mappers are important is to smooth the transition to future object managers based on CIM. Cross-Mappers allow existing management applications to use the data available from CIM-based Object Managers, as well as allow new CIM-based applications to manage existing WfM machines, thus preserving existing investments. This is shown in the diagram below:

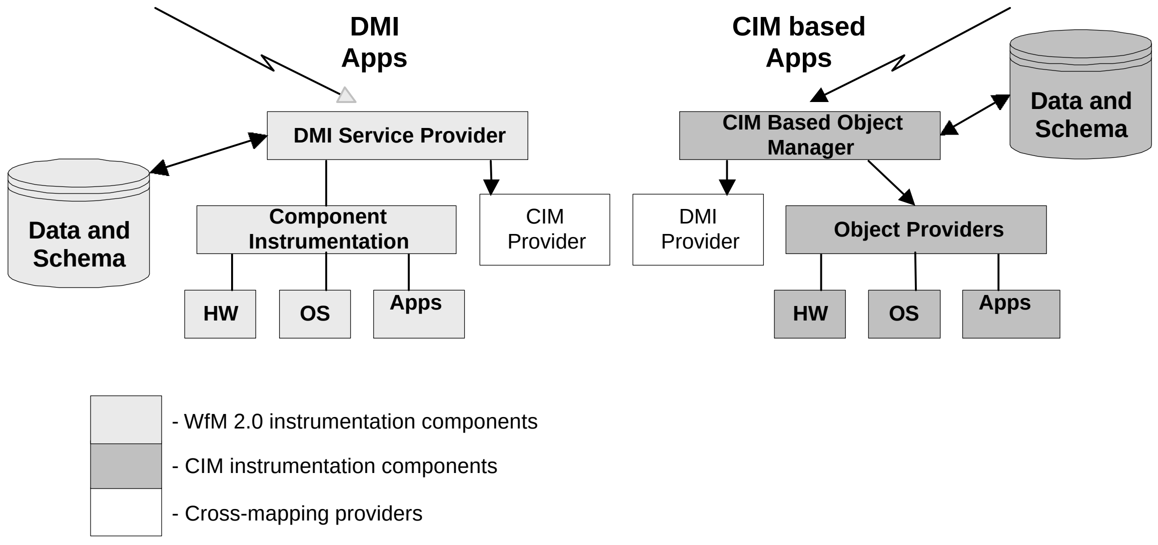

Figure 4-1 DMI/WBEM Cross-Mapping Architecture

For this diagram, the following definitions apply:

- **DMI SP.** The DMI Service Provider is a software component which supports the DMTF's DMI 2.0 management API. This API allows for both local and remote management of the platform.
- **DMI Adapter.** The DMI Adapter allows data provided by the WBEM Framework to be accessible through the DMI Service Provider. The adapter requires mapping collections, called group views and component views to expose the CIM data. The required WfM baseline group views for the desktop, mobile, and server platforms are included with the adapter infrastructure.
- **DMI Provider**. The DMI Provider allows data provided by the DMI infrastructure to be accessible through the WBEM Framework.

4.7 Considerations

This section discusses special considerations for implementing instrumentation in some particular environments addressed by this Baseline.

4.7.1 OS-Absent Environment

The OS-Absent section of the Baseline describes the requirements for a Baseline platform's support of [SMBIOS]. SMBIOS extends the platform BIOS to support retrieval of management data required in the Desktop, Mobile, and Server checklists.

4.7.2 Desktop Platforms

None.

4.7.3 Mobile Platforms

It is highly recommended that the client-side management application or agent queue events when disconnected for later forwarding upon reconnection.

4.7.4 Server Platforms

None.

4.8 Summary of Platform Requirements

Table 4-1 Instrumentation Requirements Summary

Ref #	Area	Desktop	Mobile	Server
in01	Management Framework (DMI Version 2.0 Service Provider, SNMP Agent or WBEM Framework) Installed and Active	Required (Either DMI, WBEM or SNMP)	Required (Either DMI, WBEM or SNMP)	Required (Either DMI, WBEM or SNMP)
in02	Local and Remote Access to Management Data via Standard Access Mechanisms	Required	Required	Required
in03	Events Generated according to Standard Models (DMI, SNMP or WBEM/CIM)	Recommended	Recommended	Recommended
in04	DMI Events conform to DMI Event Model	Required if DMI Events Implemented	Required if DMI Events Implemented	Required if DMI Events Implemented
in05	WBEM Events Conform to CIM Event Model	Future	Future	Future
in06	SNMP Traps conform to "DMTF SNMP to DMI Mapping Standard".	Required if SNMP Framework Implemented	Required if SNMP Framework Implemented	Required if SNMP Framework Implemented
in07	Instrumentation Supports Dynamic Devices	Required	Required	Required
in08	Instrumentation Deployed and Maintained with Product and Platform	Required - CIM Recommended Where Supported by Platform/OS	Required - CIM Recommended Where Supported by Platform/OS	Required - CIM Recommended Where Supported by Platform/OS
in09	Management Data Available	Required (per Desktop Checklist) – CIM Recommended Where Supported	Required (per Mobile Checklist) – CIM Recommended Where Supported	Required (per Server Checklist) – CIM Recommended Where Supported
in10	Backward Compatibility with the WfM 1.1 Standard for Data and Events	Required – Data and Events Visible via DMI	Required – Data and Events Visible via DMI	Required – Data and Events Visible via DMI

5. Platform Firmware

This chapter contains interface specifications for capabilities to be implemented by the platform firmware. Most of these capabilities relate to OS-absent operation of the platform, that is, operation when no OS image is resident on the platform or when a fault (for example, hard drive failure) prevents a resident OS image from booting. The capabilities addressed in this version of the Baseline include:

- A protocol and interfaces that enable a platform to locate, download from a server, and execute a bootable image appropriate to the platform's current state. These protocol and interface specifications are referred to here as the “Preboot Execution Environment” (PXE) and were previously documented as “Appendix B” of the Network PC System Design Guidelines.
- A well-known interface (SMBIOS) for access to platform data, plus a minimum set of data required to be available through this interface.
- The interface to a mechanism for disabling input devices such as keyboard and mouse during a remote boot operation.
- A format for network messages indicating exceptional events sent by the platform firmware to a management server.
- A Platform Event Trap format for communicating platform events occurring in the OS-absent interval to a remote management application.

5.1 Requirements

The following enumerate the platform firmware requirements:

1. *Preboot Execution Environment*. For platforms that have a LAN adapter on the motherboard or a LAN adapter card with an option ROM, the platform must implement the *Preboot Execution Environment v2.0* (PXE) client specifications.
2. *SMBIOS 2.2 or later* The platform must implement SMBIOS version 2.2 or later.
3. *System boot status*. Platform makes available the reason for the current system boot.
4. *Remote lockout*. If access to the platform via some hardware (for example, keyboard) may be locked out locally, the platform also allows the hardware to be locked via the described software interface.
5. ***Platform Event Trap*.** Platform supports SNMP traps which meet the Platform Event Trap format specification.
6. *Boot Integrity Services*. The platform implements the interface specified in *Boot Integrity Services Application Programming Interface, v1.0*.

7. System Network Boot Control. The platform makes available the user-controllable network-boot enable status.

5.2 Preboot Execution Environment (PXE)

5.2.1 Overview

5.2.1.1 Purpose

PXE can enable any of several functions, depending on what the downloaded image does. Examples include installing an OS on the client system, running diagnostics on the client system, and updating firmware on the client system. The choice of image to download is ultimately up to the server, but can be influenced by information conveyed by the client in its request.

Interoperability

Interoperability of PXE requires two sets of agreements:

- Between the client and the server as to the protocol used to request and download the bootable image.
- Between the downloaded image and the client as to the environment that exists on the client when the downloaded image begins executing.

The remainder of this section gives an overview of specifications for each of these.

5.2.1.2 Preboot Execution Environment (PXE) Protocol Overview

The PXE protocol is based on the Dynamic Host Configuration Protocol (DHCP), an IETF standard protocol by which clients can request configuration parameter values, and a bootable image from a server. Standard DHCP operates as follows (a highly abstracted description):

- The client broadcasts a request message. This message implicitly requests values for certain critical network-communications-related parameters and may explicitly request other values. Whether the client is requesting a bootable image is ambiguous.
- One or more servers respond to the client with configuration parameter values, and possibly, the filename of a bootable image. Generally, another message exchange occurs at this point to confirm the IP address allocated to the client.
- If the client wanted a bootable image and the server supplied the name of one, the client uses the Trivial File Transfer Protocol (TFTP) to download the image.

The PXE protocol is distinguished from standard DHCP principally by:

- The requirement of additional information (in the form of DHCP options) in the messages passed between the client and the server, including tags positively identifying the messages as originating from PXE clients and servers.
- The implicit assumption that the client *does* need a bootable image.
- The option that the downloaded image has a corresponding digital signature in order to allow detection of corruption or tampering.

The not-yet-standardized DHCP options that are either required or permitted in PXE messages convey the following types of information:

- Client hardware characteristics
- Client request for a bootable image
- Client request for a corresponding digital signature

5.2.1.3 Preboot Execution Environment (PXE) Overview

The PXE specification comprises a memory configuration plus a number of APIs to which the downloaded image must have access. These APIs provide network communication services.

A Modular PXE implementation provides for the storage of support drivers in nonvolatile storage and the subsequent retrieval to and execution from the Preboot Code and Data Space defined by [PXE].

5.2.2 PXE Client Specifications

A PXE client must be able to do all of the following as an alternative to booting from a local device.

- Request a bootable image using the PXE protocol. The platform’s implementation of this protocol must comply with the client behavior specified in [PXE]. The client must be capable of retrieving the digital signature corresponding to the boot image; the *use* of this capability *may* be configurable.
- Download the bootable image using TFTP.
- Use the boot image’s digital signature to differentiate between valid and invalid boot images, and avoid executing invalid ones.
- Initiate execution of the image in an environment that complies with the PXE memory map and PXE APIs as specified in [PXE].

A modular implementation of PXE must support all of the following:

- Support the PXE Client Specifications defined above.
- Execute the PXE BUS Driver Architecture allowing preboot support drivers to be stored and retrieved from nonvolatile storage and subsequently executed from Preboot Code and Data Space (see [PXE]).

5.2.3 Deployment Considerations

5.2.3.1 Configuration Servers

For clients to receive digital signatures for boot images, each PXE server must be supplied with either the digital signatures, or the key pair needed to create them.

5.2.3.2 Clients

To be able to validate downloaded boot images, each client must implement [BIS] and be configured with one or more trusted public keys.

5.2.4 Remote Lockout Considerations

5.2.4.1 Downloaded Executables

Downloaded executables may require that the user be “locked out” or inhibited from interfacing with the PXE client during sensitive operations, such as BIOS flash updates. Interface operations that need to be inhibited include on/off button press, reset button press, mouse movement, and key press.

5.2.4.2 Client

The System BIOS of PXE clients shall provide functionality to allow downloaded executables to lock out user interface functionality. INT 15 functions shall be provided to get and set the remote lock out settings.

5.2.4.3 Int 15h Get Remote Lockout settings

CX contains the current lockout settings as indicated below. An unsupported modification of a particular setting is indicated by attempting setting the appropriate bit in CX and noting via a read of that register that the setting is unchanged.

```
        Enter:
        AX := 2501h
        Int 15h

        Exit:
        CF == 1 || AH != 0  Failure.  This function not supported by the
                            BIOS.

        CF == 0 && AH == 0  Success.  Lockout(s) exist as follows:
        (CX & 01h) == 1     Hard On/Off switch press.
        (CX & 02h) == 2     Soft On/Off button press.
        (CX & 04h) == 4     Reset button press.
        (CX & 08h) == 8     Mouse movement.
        (CX & 10h) == 10h   Ctl-Alt-Del key sequence press.
        (CX & 20h) == 20h   Any key press.
Upper two bits of CX are reserved.All other register contents must be
        preserved.
```

5.2.4.4 Int 15h Set Remote Lockout settings

Any lockout settings set through this interface are not persistent across power cycles and reboots.

```
        Enter:
        AX := 2502h
        (CX & 01h) == 1     Lock out hard On/Off switch press.
        (CX & 02h) == 2     Lock out Soft On/Off button press.
        (CX & 04h) == 4     Lock out Reset button press.
        (CX & 08h) == 8     Lock out Mouse movement.
        (CX & 10h) == 10h   Lock out Ctl-Alt-Del key sequence press.
        (CX & 20h) == 20h   Lock out all key press.
Upper two bits of CX are reserved.

        Int 15h
        Exit:
        CF == 1 || AH != 0  Failure.  This function not supported by the
                            BIOS or could not be set.

        CF == 0 && AH == 0  Success.  Remote Lockout(s) set successfully.
All other register contents must be preserved.
```

5.3 SMBIOS Data and Standard Access to Platform Data

Some of the management data that managed systems must export, as required in the Instrumentation (Chapter 4) and the platform-specific checklists (Chapters7-9) of this Baseline, is only detectable by the platform BIOS during the preboot state. The *System Management BIOS Reference Specification* allows the BIOS to export critical management data in a standard table format. These tables allow instrumentation software running on the system in the post-boot state to retrieve the management data.

The platform BIOS must comply with [SMBIOS].

5.3.1 SMBIOS Data

The following table enumerates the required SMBIOS structures and the required minimum field content within each of those structures.

Structure Name and Type	Required?	Data Requirements
BIOS Information (Type 0)	Required	One and only one structure is present in the structure-table. *BIOS Version* and *BIOS Release Date* strings are non-null;. the date field uses a 4-digit year (for example, 1999). All other fields reflect full BIOS support information.
System Information (Type 1)	Required	*Manufacturer* and *Product Name* strings are non-null. *UUID* field identifies the system's non-zero UUID value. *Wake-up Type* field identifies the wake-up source and cannot be Unknown.
System Enclosure (Type 3)	Required	*Manufacturer* string is non-null; the *Type* field identifies the type of enclosure (Unknown is disallowed).
Processor Information (Type 4)	Required	One structure is required for each system processor. The presence of two structures with the *Processor Type* field set to *Central Processor*, for instance, identifies that the system is capable of dual-processor operations. *Socket Designation* string is non-null. *Processor Type, Max Speed*, and *Processor Upgrade* fields are all set to "known" values — in other words, the Unknown value is disallowed for each field. If the associated processor is present (i.e. the *CPU Socket Populated* sub-field of the *Status* field indicates that the socket is populated), the *Processor Manufacturer* string is non-null and the *Processor Family*, *Current Speed*, and *CPU Status* sub-field of the *Status* field are all set to "known" values. Each of the *Lx Cache Handle* fields is either set to 0xFFFF (no further cache description) or reference a valid *Cache Information* Structure.
Cache Information (Type 7)	Required	One structure is required for each external-to-the-processor cache. *Socket Designation* string is non-null if the cache is external to the processor. If the cache is present (i.e. the Installed Size is non-zero), the *Cache Configuration* field is set to a "known" value — in other words, the Unknown value is disallowed.

Structure Name and Type	Required?	Data Requirements
System Slots (Type 9)	Required	One structure is required for each upgradeable system slot. A structure is not required if the slot must be populated for proper system operation (for example, the system contains a single memory-card slot). *Slot Designation* string is non-null. *Slot Type*, *Slot Data Bus Width*, *Slot ID*, and *Slot Characteristics 1 & 2* are all set to “known” values. If device presence is detectable within the slot (for example, PCI), the *Current Usage* field must be set to either *Available* or *In-use*. Otherwise (for example, ISA), the <u>Unknown</u> value for the field is also allowed.
Physical Memory Array (Type 16)	Required	One structure is required for the system memory. *Location*, *Use*, *Memory Error Correction*, and *Maximum Capacity* are all set to “known” values. *Number of Memory Devices* is non-zero and identifies the number of *Memory Device* structures that are associated with this *Physical Memory Array*.
Memory Device (Type 17)	Required	One structure is required for each socketed system-memory device, whether or not the socket is currently populated; if the system includes soldered system-memory, one additional structure is required to identify that memory device. *Device Locator* string is set to a non-null value. *Memory Array Handle* contains the handle associated with the Physical Memory Array structure to which this device belongs. *Data Width*, *Size*, *Form Factor*, and *Device Set* are all set to “known” values. If the device is present (i.e. *Size* is non-zero), the *Total Width* field is not set to 0xFFFF (*Unknown*).
Memory Array Mapped Address (Type 19)	Required	One structure is required for each contiguous block of memory addresses mapped to a Physical Memory Array. *Ending Address* is larger than *Starting Address*. Each structure’s address range is unique and non-overlapping. *Memory Array Handle* references a Physical Memory Array structure. *Partition Width* is non-zero.
Memory Device Mapped Address (Type 20)	Required	Sufficient structures are required to map all enabled memory devices to their respective memory-array mapped addresses. *Ending Address* is larger than *Starting Address*. *Memory Device Handle* references a Memory Device structure. *Memory Mapped Address Handle* references a Memory Array Mapped Address structure. *Partition Row Position* is neither 0 nor 0xFF, nor is it greater than the *Partition Width* of the referenced Memory Array Mapped Address structure. *Interleave Position* and *Interleaved Data Depth* are not set to 0xFF (Unknown).
Boot Integrity Services Entry Point (Type 31)	Required, if BIS is supported by the platform	Both 16-bit real-mode and 32-bit flat protected-mode entry points are non-zero. The overall structure checksum evaluates to 0.
System Boot Information (Type 32)	Required	The structure’s length is at least 0x0B, i.e. at least one byte of *System Boot Status* is provided.

5.3.2 System Boot Information

5.3.2.1 Identifying the System Boot Status

The client system firmware, for example, BIOS, communicates the *System Boot Status* to the client's PXE boot image or OS-present management application via a System Management BIOS (SMBIOS) structure. When used in the PXE environment, for example, this code identifies the reason the PXE was initiated and can be used by boot-image software to further automate an enterprise's PXE sessions. For example, an enterprise could choose to automatically download a hardware-diagnostic image to a client whose reason code indicated either a firmware- or operating system-detected hardware failure.

Note: Within the following section, "this specification" refers to the [SMBIOS] specification.

Offset	Name	Length	Value	Description
00h	Type	BYTE	32	System Boot Information structure identifier
01h	Length	BYTE	Varies	Length of the structure, in bytes.
02h	Handle	WORD	Varies	
04h	Reserved	6 BYTEs	00h	Reserved for future assignment via this specification; all bytes are set to 00h.
05h	Reason Code Data	*Length*–10 Bytes	Varies	The Status and Additional Data fields that identify the system boot status. See System Boot Status for additional information.

5.3.2.1.1 System Boot Status

Boot Status Name	**Status**	**Additional Data**
No errors detected	0	None
No bootable media	1	none
The "normal" operating system failed to load.	2	none
Firmware-detected hardware failure, including "unknown" failure types.	3	none
Operating system-detected hardware failure. For ACPI OS's, the system firmware might set this reason code when the OS reports a boot failure via interfaces defined in the *Simple Boot Flag Specification*.	4	none
User-requested boot, usually via a keystroke	5	none
System security violation	6	none
Previously-requested image. This reason code allows a coordination between OS-present software and the OS-absent environment. For example, an OS-present application might enable (via a platform-specific interface) the system to boot to the PXE and request a specific boot-image.	7	varies
A system watchdog timer expired, causing the system to reboot.	8	none
Reserved for future assignment via this specification.	9-127	Varies

Boot Status Name	Status	Additional Data
Vendor/OEM-specific implementations. The Vendor/OEM identifier is the "Manufacturer" string found in the SMBIOS *System Identification* structure.	128-191	Varies
Product-specific implementations. The product identifier is formed by the concatenation of the "Manufacturer" and "Product Name" strings found in the SMBIOS *System Information* structure.	192-255	Varies

5.3.3 System Network Boot Control

The client system firmware, e.g. BIOS, indicates whether or not the client system firmware supports function key-initiated Network Service Boot via the System Management BIOS (SMBIOS) BIOS Information structure.

When function key-initiated Network Service Boot is not supported by the client system firmware, the network adapter option ROM may choose to offer this functionality on its own, thus offering this capability to legacy systems. When the function is supported, (BIOS Characteristic Extension Byte 2, bit 1 is set to 1), the network adapter option ROM shall not offer this capability.

Offset	Name	Length	Value	Description
00h	Type	BYTE	0	BIOS Information Indicator
01h	Length	BYTE	Varies	12h + number of *BIOS Characteristics Extension* Bytes. If no Extension Bytes are used the Length will be 12h. For v2.1 and v2.2 implementations, the length is 13h since one extension byte is defined. For v2.3 and later implementations, the length is at least 14h since two extension bytes are defined.
02h	Handle	WORD	Varies	
04h	Vendor	BYTE	STRING	String number of the BIOS Vendor's Name
05h	BIOS Version	BYTE	STRING	String number of the BIOS Version. This is a free form string which may contain Core and OEM version information.
06h	BIOS Starting Address Segment	WORD	Varies	Segment location of BIOS starting address, e.g.0E800h. Note: The size of the runtime BIOS image can be computed by subtracting the Starting Address Segment from 10000h and multiplying the result by 16.
08h	BIOS Release Date	BYTE	STRING	String number of the BIOS release date. The date string, if supplied, is in either mm/dd/yy or mm/dd/yyyy format. If the year portion of the string is two digits, the year is assumed to be 19yy. ***Note***: The mm/dd/yyyy format is required for SMBIOS version 2.3 and later.
09h	BIOS ROM Size	BYTE	Varies (n)	Size (n) where 64K * (n+1) is the size of the physical device containing the BIOS, in bytes
0Ah	BIOS Characteristics	QWO RD	Bit Field	Defines which functions the BIOS supports. PCI, PCMCIA, Flash, etc.

Offset	Name	Length	Value	Description
12h	BIOS Characteristics Extension Bytes	Zero or more BYTEs	Bit Field	Optional space reserved for future supported functions. The number of Extension Bytes that are present is indicated by the Length in offset 1 minus 12h.

5.3.3.1 BIOS Characteristics Extension Byte 2

This information, available for SMBIOS version 2.3 and later, appears at offset 13h within the BIOS Information structure.

Byte Bit Position	Meaning if Set
Bit 0	*BIOS Boot Specification* supported
Bit 1	F12=Network Boot supported.
Bits 2:7	Reserved for future assignment via this specification.

5.4 OS-absent Security Infrastructure

It is recommended that the PXE-based boot process be able to verify the integrity and source of the boot image that it downloads. Also, each subsequent bootstrap stage can benefit from support for verifying the integrity and source of the items that it downloads. The [BIS] document describes these usage models in more detail, and describes an interface that supports them. It is recommended that platforms implement the interface specified in [BIS].

5.5 SNMP Traps

A common Platform Event Trap format for OS-absent, preboot, and boot failure events is defined in The Platform Event Trap Format specification. A platform event is defined as one that originates from platform firmware (BIOS, OS Bootstrap Loader) or platform hardware (ASIC, chip set, or micro-controller) independent of the state of the operating system or instrumentation software.

Platform event notification under OS-absent conditions requires a very simple event format and a network protocol which can be generated by state machines of very limited complexity. SNMP traps provide a low cost, well known method of achieving these goals. SNMP traps can be handled by most existing management applications.

Systems which support OS-absent, preboot, or boot failure alerting should support the Platform Event Trap. This enables management applications to receive asynchronous notification of critical platform state changes which may prevent a system from booting and to interpret them in a common fashion across Wired for Management platforms.

The OS or instrumentation software may also generate traps using this format, though it is expected that runtime software will use standard events already defined in the SNMP, DMI, or CIM management frameworks.

5.6 Considerations

5.6.1 Desktop Platforms

None.

5.6.2 Mobile Platforms

On mobile platforms, the PXE client specifications is required only if the platform has built-in LAN on the notebook platform or docking station. For mobile platforms without LAN down, but which support LAN adapter cards (CardBus or NIC in docking station), implementation of PXE client specifications is recommended.

Mobile manufacturers' implementation mechanisms may require that the notebook be on AC power for PXE to be enabled. The decision to require a connection to AC power for PXE to be enabled is the manufacturer's choice. Such a restriction is allowed but not required by the WfM Baseline.

5.6.3 Server Platforms

It is strongly recommended that LAN adapter add-in cards used on servers implement option ROMs which support the PXE. PXE as defined in this Baseline provides a standard mechanism for loading alternative service environments (for example, OS patches and diagnostics environments) which can be exploited to reduce the Mean Time to Repair, and thereby increase the Reliability, Availability, and Serviceability of server systems.

Remote management is a high priority for server systems, since servers will typically be isolated from local human user intervention (see Management Characteristics of Servers, Section 9.2). This specification does not address technologies which enable automatic or remote initiation of the PXE boot service. These capabilities may be addressed in future versions of this specification. Server vendors are encouraged to implement mechanisms, such as the expiration of watchdog timers, that allow the platform to automatically initiate the PXE protocol when critical conditions warrant a remote preboot intervention. Using PXE and the System Reboot Reason Codes, the managed server can request a variety of repair services from the remote configuration server, based on the various critical conditions which can cause the managed server to contact the configuration server.

Servers are also encouraged to implement mechanisms that allow the remote configuration server or other remote management applications to force the server to initiate the PXE protocol. The Platform Event Trap can be used to bring a critical server condition to the attention of the remote application or system administrator. Mechanisms which remotely force the server to initiate a remote boot via PXE should be sensitive to the current state of the server, and should allow the OS to gracefully shutdown the server whenever possible, prior to bringing the server up through BIOS to the point where the boot options can execute.

5.7 Summary of Platform Requirements

Table 5-1 Platform Firmware Requirements Summary

Ref #	Area	Desktop	Mobile	Server
pf01	Preboot Execution Environment	Required	Required if LAN on motherboard present Recommended if LAN adapter card present	Recommended
pf02	SMBIOS or later	Required	Required	Required
pf03	System boot status	Required	Required	Required
pf04	Remote lockout	Required	Required	Required
pf05	Platform Event Traps	Recommended	Recommended	Recommended
pf06	Boot Integrity Services	Recommended	Recommended	Recommended
pf07	SMBIOS Data	Required	Required	Required
pf08	System Network Boot Control	Required	Required	Required

6. Problem Resolution Requirements

The Problem Resolution Standards refer collectively to the *Solution Exchange Standard V1.0 (SES), Service Incident Exchange Standard V1.0 (SIS),* and *MOF Addendum V1.0* [SES/SIS] which are published jointly by the Desktop Management Taskforce (DMTF) and the Customer Support Consortium (CSC). These standards describe the formats and transactions necessary to exchange trouble tickets between SIS Service Requesters and SIS Service Providers, as well as to exchange solution knowledge between content providers/publishers and knowledge engines. For a complete definition of the standard trouble tickets and transactions, refer to these documents.

A SIS Service Requester is any entity which is requesting service from an SIS Service Provider (for example, help desk or management application). An SIS Service Requester can be the platform originally experiencing the difficulty or a help desk which needs to escalate a problem to another provider. This Baseline describes the trouble ticketing requirements for a platform agent. The presence of such an agent on the platform allows that platform to participate in the trouble ticket environment provided by a help desk or management application.

The [SES/SIS] standards derive much of their power by allowing automated, lossless transportation of problem information from beginning (with the detection of the problem on the platform) through intermediate providers to the return and activation of the solution to the platform. However, this Baseline addresses only the requirements for the platform agent which allow standard trouble tickets (incidents) to be generated. In particular, no requirements are placed on help desks or management applications.

6.1 Requirements

The following enumerate the requirements on the platform trouble ticket agent:

1. *Standard Trouble Ticket Agent Installed and Active*. The trouble ticket agent should be installed and available on the platform. The presence of this agent will guarantee that a local management application has a compatible interface into the enterprise's trouble ticket environment. It also guarantees that a Baseline-enabled remote management application will be able to receive and resolve trouble tickets from a Baseline platform.

Note: Requirements 2 through 4 must be met only if the Trouble Ticket Agent is installed and active on the platform.

2. *SES Compliance Level 2*. All SES objects within the trouble ticket must comply with SES Level 2 definitions where they exist. The sole exception to this is the SOLUTION object presented as an instance of a SERVICE INCIDENT RESOLUTION, which may be of any compliance level.

3. *Transaction Objects*. For a given transaction, all of the SIS objects required by the *Service Incident Exchange Standard* must be included.

4. *Required Transactions*. All transactions defined by SIS must be supported.
5. *Trouble Ticket Initiation*. At a minimum, the user should be allowed to generate a trouble ticket based on an observed problem. Additionally, the system may be able to automatically generate a trouble ticket based on instrumented alarms or application errors.

6.2 Solution Exchange

A standard trouble ticket may contain several objects defined by the Solution Exchange Standard, including PROBLEM, SOLUTION and CONTACT information. In general, all SES objects in a trouble ticket must conform to the Level 2 compliance definitions. However, in the case of an existing SOLUTION which is incorporated in a trouble ticket, as is, no compliance requirements are made.

6.3 Service Incident Exchange Requirements

This Baseline does not require that a system generate standard trouble tickets based on observed or monitored events. However, the potential cost reduction (by allowing automation and accurate data transfer end-to-end) is great, and it is recommended that platforms in fact do so. To allow the deployment of applications which implement automated problem resolution widely into any management application or help desk, platforms must be equipped with a standard trouble ticketing agent. This agent is not responsible for the initiation or resolution of a trouble ticket. Rather, it ensures that the platform remotely supports the object definitions and transactions described by [SES/SIS]. This agent must be able to:

- Present the problem, contact, contract, and activity information in the standard object format.
- Receive solutions in the same format.
- Pose questions to an SIS Service Provider in the form of SIS transactions (for example, *Request Entitlement*).
- Respond to a SIS Service Provider's transactions (for example, *Request Problem Information*, *Provide Problem Resolution*, *Request Closure*).

The system need not be capable of automatically activating the solution. Simply receiving it (including the correct response transactions back to the SIS Service Provider) and presenting it to the user is sufficient.

6.4 Considerations

This section describes environmental considerations when implementing standard trouble tickets.

6.4.1 OS-absent Environment

Not applicable.

6.4.2 Desktop Platforms

None.

6.4.3 Mobile Platforms

It is recommended that mobiles be able to generate standard trouble tickets. However, mobile systems are not always connected, and trouble ticket transactions may need to be queued until a connection is made. When the connection is made, the transactions must follow the normal procedure with the SIS Service Provider—in other words, transaction, followed by responding transaction.

6.4.4 Server Platforms

Most applicable to servers is the automatic (versus user-initiated) trouble ticket generation model. It is highly recommended that standard trouble tickets be generated by the trouble ticket agent when a DMI indication or an SNMP trap which indicates serious or critical state changes is reported by the instrumentation software.

6.5 Summary of Platform Requirements

Table 6-1 Problem Resolution Requirements Summary

Ref #	*Area*	*Desktop*	*Mobile*	*Server*
pr01	Standard Trouble Ticket Agent Installed and Active	Recommended	Recommended	Recommended
pr02	SES Compliance Level 2	Required*	Required*	Required*
pr03	Transaction Objects	Required*	Required*	Required*
pr04	Required Transactions	Required*	Required*	Required*
pr05	Trouble Ticket Initiation	Recommended	Recommended	Recommended

**** = If pr01 is met on the platform***

7. Checklist: Desktop Platforms

Desktops cover the widest variety of any class of system. They constitute the “vast multitude” in most computing environments.

A major trend for desktops is to adopt more and more of the features pioneered by servers and mobile systems. In particular, desktops have increasing power and noise management requirements with solutions that were only found on mobile platforms a few years ago. On the other hand, it is probable that more desktops are used as servers than servers are used as servers. Desktops, therefore, have the requirements for high availability and manageability that are typically associated with servers. Desktops are the most price-sensitive class due to their predominance in the market.

Desktops usually have the widest variety of peripherals attached since desktops are generally adapted to the environment of those working on them. For example, graphics artists might require graphics tablets while engineers may have instrumentation attachment requirements.

Desktops share with mobiles the lack of any commonly accepted back-up strategy. They are easier to monitor and track than mobile systems but are portable enough to still have substantial tracking issues. They are ubiquitous enough that many companies simply swap out non-functioning systems rather than attempt to resolve issues.

Desktop offerings change more rapidly than servers so it is probable that most large networks of desktops will be heterogeneous, even if all of the systems are purchased from the same vendor.

Desktops have found the most favor in the home environment. Home and branch office (remote site) manageability share much in common.

7.1 Detailed Requirements

The following table summarizes the WfM requirements on desktop systems.

Table 7-1 Desktop Requirements

Ref #	*Capability*	*Required?*	*Comments*
in01	Management Framework (DMI Version 2.0 Service Provider, SNMP Agent or WBEM Framework) Installed and Active	Required	Either DMI, WBEM or SNMP
in02	Local and Remote Access to Management Data via Standard Access Mechanisms	Required	
in03	Events Generated according to Standard Models (DMI, SNMP or WBEM/CIM)	Recommended	
in04	DMI Events conform to DMI Event Model	Required	if DMI Events Implemented
in05	WBEM Events Conform to CIM Event Model	Future	
in06	SNMP Traps conform to “DMTF SNMP to DMI Mapping Standard”.	Required	if SNMP Framework implemented
in07	Instrumentation Supports Dynamic Devices	Required	
in08	Instrumentation Deployed and maintained with Product and Platform	Required	CIM Recommended where supported by platform/OS
in09	Management Data Available	Required	per Desktop Checklist
in10	Backward Compatibility with the WfM 1.1 Standard for Data and Events	Required	Data and Events Visible via DMI
pf01	Preboot Execution Environment	Required	
pf02	SMBIOS 2.2 or later	Required	
pf03	System boot status	Required	
pf04	Remote lockout	Required	
pf05	Platform Event Traps	Recommended	
pf06	Boot Integrity Services	Recommended	
pf07	SMBIOS data	Required	
pf08	System Network Boot Control	Required	
pm01	ACPI-compliant	Required	
pm02	Recommended Sleep Mode	S3	
pm03	Remote Wakeup	Required	

Ref #	Capability	Required?	Comments
pm04	Magic Packet	Required	
pm05	Packet Filtering	Recommended	
pm06	Wake On Ring	Recommended	
pm07	Bus Power Management	Required	
pr01	Standard Trouble Ticket Agent Installed and Active	Recommended	
pr02	SES Compliance Level 2	Required	If pr01 is met on the platform
pr03	Transaction Objects	Required	If pr01 is met on the platform
pr04	Required Transactions	Required	If pr01 is met on the platform
pr05	Trouble Ticket Initiation	Recommended	

The following two tables summarize the DMI MIF requirements for desktops. Table 7-2 lists DMI groups required in all desktops; Table 7-3 lists DMI groups required for desktops in some circumstances.

Table 7-2 DMI Group Requirements for Desktops

Group	Required?	Comments
DMTF|ComponentID|001	Required	
DMTF|Processor|003	Required	
DMTF|System Memory Settings|001	Required	
DMTF|Motherboard|001	Required	
DMTF|Keyboard|003	Required	
DMTF|Pointing Device|001	Required	See Note 3.
DMTF|Parallel Ports|003	Required	
DMTF|Serial Ports|003	Required	
DMTF|Disks|002	Required	
DMTF|General Information|001	Required	
DMTF|Memory Array Mapped Addresses|001	Required	
DMTF|Memory Device Mapped Addresses|001	Required	
DMTF|Physical Memory Array|001	Required	
DMTF|Operating System|001	Required	
DMTF|Physical Container Global Table|001	Required	
DMTF|System BIOS|001	Required	
DMTF|System Cache|002	Required	
DMTF|System Slots|003	Required	
DMTF|Video BIOS|001	Required	
DMTF|Video|002	Required	
DMTF|Network Adapter 802 Port|001	Required	
DMTF|Network Adapter Driver|001	Required	
DMTF|Memory Device|001	Required	

Note 1: Group revision levels must be equal to or greater than those listed.

Note 2: 'Legacy' System Resource Management and Physical Memory Management groups (see Tables 7-2 and 7-3 in version 1.1) no longer allowed.

Note 3: DMTF|Mouse|003 is replaced by DMTF|Pointing Device|001.

Note 4: The groups are identical to the ones required by the DMTF [DMI Conform].

Table 7-3 Additional DMI Group Requirements for Desktops

Group	Required?	Comments
DMTF|System Resources 2|001	Required	See Note 5.
DMTF|System Resource Device Info|001	Required	See Note 5.
DMTF|System Resource DMA Info|001	Required	See Note 5.
DMTF|System Resource I/O Info|001	Required	See Note 5.
DMTF|System Resource IRQ Info|001	Required	See Note 5.
DMTF|System Resource Memory Info|001	Required	See Note 5.
DMTF|Monitor Resolutions|002	Required	See Note 6.

Note 5: These groups are optional for those cases where the OS itself provides the instrumentation.

Note 6: This group is required only for systems that have video and monitor (or other display device) hardware that supports DDC interfaces.

8. Checklist: Mobile Platforms

Mobile computers have many characteristics that make them different from desktop systems. Among these are size, weight, power constraints, and roaming (that is, occasionally-connected, pull-model communications). Dynamically attached devices (for example, PC Cards and swappable drives), while not unique to mobile computers, have long been a distinguishing characteristic for mobiles and are covered in this specification. Mobile computers are also powered from batteries when not connected to AC power. Several of these characteristics have major impacts on the way the Wired for Management Baseline is implemented on mobile computers.

Mobile clients are occasionally connected to communications lines meaning they roam and are not always reachable by management applications. Managing a mobile asset is a difficult task for corporations, resulting in a higher TCO for notebook systems. The connection type also varies based on what is available at the traveler’s destination (LAN or dial-up). This affects the bandwidth and latency available for remote management. Additionally, the connection is a pull model, meaning it is made and broken at the discretion of the mobile worker—not IT. All these differences profoundly impact the way the Baseline is implemented on a mobile computer.

Mobile platforms are designed to be dynamically expandable. Instrumentation must be “smarter” because devices can dynamically appear/disappear in a mobile system (for example, PC cards, hot docking). This is especially true of dynamic communications adapters. Key parts of the Baseline rely on communications that occur when the OS has not yet booted or is inactive (that is, remote wakeup, Preboot Execution Environment). That necessitates the platform have intimate knowledge of the communications adapter. Most mobile computers have communications adapters that can be dynamically added/removed at any moment (that is, PC card NIC, hot docking to a station that contains a NIC). The ease with which these adapters can be swapped means that a new and different adapter, unplanned by the system manufacturer, could be inserted by the user. This must be taken into account when applying the Baseline to mobile platforms.

This section of the WfM Baseline specifies the minimum level of manageability that all mobiles must meet regardless of how they are used. The objective is to ensure that mobile computers can be managed by any management application that uses the Baseline’s open interface specifications the same as any other system that conforms to the Baseline.

8.1 Detailed Requirements

The following table gives specific requirements for mobile platforms.

Table 8-1 Mobile Requirements

Ref#	***Capability***	***Required?***	***Comments***
in01	Management Framework (DMI Version 2.0 Service Provider, SNMP Agent or WBEM Framework) Installed and Active	Required	Either DMI, WBEM or SNMP
in02	Local and Remote Access to Management Data via Standard Access Mechanisms	Required	
in03	Events Generated according to Standard Models (DMI, SNMP or WBEM/CIM)	Recommended	
in04	DMI Events conform to DMI Event Model	Required	if DMI Events Implemented
in05	WBEM Events Conform to CIM Event Model	Future	
in06	SNMP Traps conform to *"DMTF SNMP to DMI Mapping Standard".*	Required	If SNMP framework implemented
in07	Instrumentation Supports Dynamic Devices	Required	
in08	Instrumentation Deployed and Maintained with Product and Platform	Required	CIM Recommended Where Supported by Platform/OS
in09	Management Data Available	Required	per Mobile Checklist
in10	Backward Compatibility with the WfM 1.1 Standard for Data and Events	Required	Data and Events Visible via DMI
pf01	Preboot Execution Environment	Required if LAN on motherboard present Recommended if LAN adapter card present	
pf02	SMBIOS 2.2 or later	Required	
pf03	System boot status	Required	
pf04	Remote lockout	Required	
pf05	Platform Event Traps	Recommended	
pf06	Boot Integrity Services	Recommended	
pf07	SMBIOS data	Required	
pf08	System Network Boot Control	Required	

Ref#	*Capability*	*Required?*	*Comments*
pm01	ACPI-compliant	Recommended	
pm02	Recommended Sleep Mode	S3	
pm03	Remote Wakeup	Recommended	
pm04	Magic Packet	Recommended	
pm05	Packet Filtering	Recommended	
pm06	Wake On Ring	Recommended	
pm07	Bus Power Management	Required	
pr01	Standard Trouble Ticket Agent installed and active	Recommended	
pr02	SES Compliance Level 2	Required	If pr01 is met on the platform
pr03	Transaction Objects	Required	If pr01 is met on the platform
pr04	Required Transactions	Required	If pr01 is met on the platform
pr05	Trouble Ticket Initiation	Recommended	

The following two tables summarize the DMI Group requirements for mobile platforms. Table 8-2 lists DMI groups required in all mobile platforms; Table 8-3 lists DMI groups required for mobile platforms in some circumstances.

Table 8-2 DMI Groups for Mobile Platforms

Group	Required?	Comments
DMTF|ComponentID|001	Required	
DMTF|Disks|002	Required	
DMTF|General Information|001	Required	
DMTF|Keyboard|003	Required	
DMTF|Operating System|001	Required	
DMTF|Physical Container Global Table|001	Required	
DMTF|Processor|003	Required	
DMTF|System BIOS|001	Required	
DMTF|System Cache|002	Required	
DMTF|System Slots|003	Required	Required if system slots (including PC Card slots) are present
DMTF|Video BIOS|001	Required	
DMTF|Video|002	Required	
DMTF|Memory Device|001	Required	
DMTF|Memory Array Mapped Addresses|001	Required	
DMTF|Memory Device Mapped Addresses|001	Required	
DMTF|Physical Memory Array|001	Required	
DMTF|Network Adapter 802 Port|001	Required	Required if network adapter is present
DMTF|Network Adapter Driver|001	Required	Required if network adapter is present
DMTF|Pointing Device|001	Required	See Note 3.
DMTF|Portable Battery|001	Required	
DMTF|Dynamic States|001	Required	
DMTF|System Memory Settings|001	Required	
DMTF|Motherboard|001	Required	
DMTF|Parallel Ports|003	Required	
DMTF|Serial Ports|003	Required	

Note 1: Group revision levels must be equal to or greater than those listed.

Note 2: 'Legacy' System Resource Management and Physical Memory Management groups (see Tables 7-2 and 7-3 in version 1.1) no longer allowed.

Note 3: DMTF|Mouse|003 is replaced by DMTF|Pointing Device|001.

Note 4: The groups are identical to the ones required by the DMTF [DMI Conform].

Table 8-3 Additional DMI Groups for Mobile Platforms

Group	Required?	Comments
DMTF|System Resources 2|001	Required	See Note 5.
DMTF|System Resource Device Info|001	Required	See Note 5.
DMTF|System Resource DMA Info|001	Required	See Note 5.
DMTF|System Resource I/O Info|001	Required	See Note 5.
DMTF|System Resource IRQ Info|001	Required	See Note 5.
DMTF|System Resource Memory Info|001	Required	See Note 5.
DMTF|Monitor Resolutions|002	Required	See Note 6.

Note 5: These groups are optional for those cases where the OS itself provides the instrumentation.

Note 6: This group is required only for systems that have video and monitor (or other display device) hardware that supports DDC interfaces.

9. Checklist: Server Platforms

Server systems differ from desktop and mobile clients in a number of important ways. This section outlines how the technologies addressed by the Wired for Management Baseline apply to server systems.

In addition, an array of additional management technologies are in widespread use on server systems. These technologies are designed to ensure that servers meet the reliability, serviceability, and availability requirements appropriate to the business environments where servers are deployed. Additional server technologies and requirements may be defined at a later date.

9.1 What Makes Servers Different

The priorities for server management are quite different from those for client (desktop or mobile) management. While the major goal for client management is to reduce per-client maintenance and administration costs, the top priority for servers tends to be minimizing down time. The failure of a server, or of a service hosted by a server, can impact many clients simultaneously. The ability of the users of client systems to accomplish their tasks is directly impacted by the availability of critical services exported by servers in the computing environment. Examples include file and print services, electronic mail and messaging servers, Internet gateways, routing services, and business-critical application servers.

Servers have traditionally supported advanced manageability features, focusing on remote management from centralized administrative consoles, failure detection and critical event notification, and diagnostic tools. In addition, servers frequently are configured with redundant subsystems so that the server can tolerate the failure of critical components. Redundant subsystems may support hot-plug (hot-swap) connectors, so that repairs can be performed without bringing the server off-line. Servers are also configured for scaleable performance and capacity; multiple processors, multiple high performance I/O peripherals, multi-chassis configurations, intelligent I/O processors, and rack-mounted chassis all present additional management requirements. To support these availability, serviceability, and scaling requirements, servers frequently support sophisticated platform management infrastructures consisting of networks of environmental sensors, firmware-based event logs, and Field Replaceable Unit (FRU) identification support at the hardware level.

9.2 Management Characteristics of Servers

The TCO of the server is measured by metrics associated with Reliability, Availability, and Serviceability (RAS). Server management mechanisms focus on providing fault prediction, fault detection, and fault resilience. If a failure does occur, server management strives to provide rapid servicing of the problem, since the Mean Time to Repair directly affects the overall availability of the server. Rapid restoration of the server to full service usually requires remote notification when the problem occurs, information that identifies the FRU that failed, and, if possible, remote mechanisms for reconfiguration and recovery. In addition, servers frequently support automatic recovery mechanisms which allow the server to return to operation after a failure without external intervention; for example, servers may support an automatic server restart capability. Many of these services are accessed and controlled via instrumentation interfaces.

Servers frequently support redundant and hot-swap hardware subsystems; examples include disks, power supplies, fans/blowers, and hot swap PCI cards. Instrumentation needs to be smart enough to handle multiple instances of devices and individual devices which dynamically appear and disappear.

High quality remote management is also a high priority. Most client systems will have a user who, if necessary, can intervene to take local actions, such as typing at the keyboard, warm or cold reset, cycling system power, etc., whereas a server will usually be isolated from local user intervention - both logically and physically. Therefore, servers are usually managed using a push model, where management sessions are invoked by the remote administrator. Some servers are configured without local I/O devices ("headless" servers); these servers are completely dependent on remote management. When multiple servers are combined into clusters, remote management allows all servers in the cluster to be managed from a single administrative console and a single set of physical I/O devices.

Remote management provides the interfaces for retrieving system health information, delivering problem alerts, and driving system recovery functions (such as system power cycling, resetting, and reconfiguration). Remote management can also provide interfaces by which operating systems, drivers, applications, and additional management software, such as remote diagnostics, can be loaded.

The primary communication channel for remote server management may be WAN, LAN, or modem connections. For example, modem connections are common in small office or branch office environments. Servers, therefore, share some characteristics with mobile clients, in that their connection type can vary, and the bandwidth and latency available for remote management can vary greatly.

Emergency management is also a high priority in servers. Emergency management in this case refers to mechanisms that allow remote management to occur under conditions where the normal remote communications links have failed. This may be because the server is powered down, or because a hardware or software failure is preventing the operating system from running or making a remote connection.

Emergency management functions can be accomplished with preboot management technologies, such as the Preboot Execution Environment (PXE). These functions include running system configuration software, accessing system management information logs, and downloading and re-installing the operating system and platform firmware/BIOS.

Emergency management will often provide a secondary communication channel, used for remote management communications when the normal communication interfaces are inoperative. The secondary channel can be implemented using an alternate network connection, telephone line, or other communication media. The robustness of the implementation is judged on its degree of independence from the primary communication physical interfaces and software.

9.3 Detailed Requirements

The following table summarizes the WfM requirements on servers.

Table 9-1 Server Requirements

Ref#	*Capability*	*Required?*	*Comments*
in01	Management Framework (DMI Version 2.0 Service Provider, SNMP Agent or WBEM Framework) Installed and Active	Required	Either DMI, WBEM or SNMP
in02	Local and Remote Access to Management Data via Standard Access Mechanisms	Required	
in03	Events Generated according to Standard Models (DMI, SNMP or WBEM/CIM)	Recommended	
in04	DMI Events conform to DMI Event Model	Required	if DMI Events Implemented
in05	WBEM Events Conform to CIM Event Model	Future	
in06	SNMP Traps conform to "*DMTF SNMP to DMI Mapping Standard*".	Required	If SNMP Framework implemented
in07	Instrumentation Supports Dynamic Devices	Required	
in08	Instrumentation Deployed and Maintained with Product and Platform	Required	CIM Recommended Where Supported by Platform/OS
in09	Management Data Available	Required	per Server Checklist
in10	Backward Compatibility with the WfM 1.1 Standard for Data and Events	Required	Data and Events Visible via DMI
pf01	Preboot execution environment	Recommended	
pf02	SMBIOS 2.2 or later	Required	
pf03	System boot status	Required	
pf04	Remote lockout	Required	
pf05	Platform Event Traps	Recommended	
pf06	Boot Integrity Services	Recommended	

Ref#	***Capability***	***Required?***	***Comments***
pf07	SMBIOS Data	Required	
pf08	System Network Boot Control	Required	
pm01	ACPI-compliant	Required as defined in [HDG]	See server considerations
pm02	Recommended Sleep Mode	S1	
pm03	Remote Wakeup	Recommended	
pm04	Magic Packet	Recommended	
pm05	Packet Filtering	Recommended	
pm06	Wake On Ring	Recommended	
pm07	Bus Power Management	Recommended	
pr01	Standard Trouble Ticket Agent Installed and Active	Recommended	
pr02	SES Compliance Level 2	Required	If pr01 is met on the platform
pr03	Transaction Objects	Required	If pr01 is met on the platform
pr04	Required Transactions	Required	If pr01 is met on the platform
pr05	Trouble Ticket Initiation	Recommended	

Table 9-2 DMI Groups for Servers

Group	**Required?**	**Comments**
*DMTF|ComponentID|001	Required	
*DMTF|Cooling Device|002	Required	Required if supported by platform
*DMTF|Cooling Unit Global Table|001	Required	
DMTF|Disks Mapping Table|001	Obsolete	See Table 9.3 Mass Storage Groups for Servers
*DMTF|Disks|002	Obsolete	See Table 9.3 Mass Storage Groups for Servers
*DMTF|FRU|002	Required	
*DMTF|General Information|001	Required	
*DMTF|Keyboard|003	Required	
*DMTF|Memory Array Mapped Addresses|001	Required	
*DMTF|Memory Device Mapped Addresses|001	Required	
*DMTF|Memory Device|001	Required	
*DMTF|Motherboard|001	Required	

Group	Required?	Comments
*DMTF|Network Adapter 802 Port|001	Required	Required for adapters provided on the motherboard
*DMTF|Network Adapter Driver|001	Required	Required for adapters provided on the motherboard
*DMTF|Operating System|001	Required	
*DMTF|Operational State|003	Required	
*DMTF|Parallel Ports|003	Required	Required for ports provided on the motherboard
DMTF|Partition|001	Obsolete	
*DMTF|Physical Container Global Table|001	Required	
*DMTF|Physical Memory Array|001	Required	
*DMTF|Pointing Device|001	Required	See Note 3
*DMTF|Power Supply|002	Required	
*DMTF|Power Unit Global Table|001	Required	
*DMTF|Processor|003	Required	
*DMTF|Serial Ports|003	Required	Required for ports provided on the motherboard
*DMTF|System BIOS|001	Required	
*DMTF|System Cache|002	Required	
DMTF|System Hardware Security|001	Required	Required if supported by platform
*DMTF|System Memory Settings|001	Required	
DMTF|System Power Controls|001	Required	Required if supported by platform
*DMTF|System Slots|003	Required	
DMTF|Temperature Probe|001	Required	Required if supported by platform
*DMTF|Video|002	Required	
DMTF|Voltage Probe|001	Required	Required if supported by platform

Note 1: Group revision levels must be equal to or greater than those listed.

Note 2: ‘Legacy’ System Resource Management and Physical Memory Management groups (see Tables 7-2 and 7-3 in v1.1) no longer allowed.

Note 3: DMTF|Mouse|003 is replaced by DMTF|Pointing Device|001.

Note 4: Disks, Disk Mapping Table and Partitions are legacy groups superseded by the Mass Storage MIF.

Legend for Table 9-2:

DMTF CG = Desktop Management Task Force [DMI Conform]

* These groups are also required by the DMTF CG

Table 9-3 Additional DMI Groups for Server Platforms

DMTF|System Resource Device Info|001	Recommended	Downgrade from v1.1. See Note 5.
DMTF|System Resource DMA Info|001	Recommended	Downgrade from v1.1. See Note 5.
DMTF|System Resource I/O Info|001	Recommended	Downgrade from v1.1. See Note 5.
DMTF|System Resource IRQ Info|001	Recommended	Downgrade from v1.1. See Note 5.
DMTF|System Resource Memory Info|001	Recommended	Downgrade from v1.1. See Note 5.
DMTF|System Resources 2|001	Recommended	Downgrade from v1.1. See Note 5.

Note 5: It is recommended that the OS platform itself provides instrumentation for groups in this section. Data is OS-specific and may not be available from all server operating systems.

9.3.1 Mass Storage Subsystems

It is strongly recommended that mass storage subsystems are instrumented on server systems. If instrumentation is provided, it must support the following data:

DMTF|Storage Devices|001	REQUIRED
EventGeneration|DMTF^^Storage Devices|001	REQUIRED
DMTF|Storage Controller|001	REQUIRED
EventGeneration|DMTF^^Storage Controller|001	REQUIRED
DMTF|Bus Port|001	REQUIRED
EventGeneration|DMTF^^Bus Port|001	RECOMMENDED
DMTF|Fibre Channel Bus Port Extensions|001	RECOMMENDED
DMTF|SSA Bus Port Extensions|001	RECOMMENDED
DMTF|Mass Storage Association|001	REQUIRED
EventGeneration|DMTF^^Mass Storage Association|002	RECOMMENDED
DMTF|Bus Port Association|001	REQUIRED
DMTF|Worldwide Identifier|001	REQUIRED
DMTF|SubComponent Software|001	REQUIRED
DMTF|Mass Storage Statistics|001	RECOMMENDED
DMTF|Operational State|003	REQUIRED
DMTF|System Cache|003	REQUIRED
DMTF|ComponentID|001	REQUIRED

It is strongly recommended that RAID subsystems are instrumented on server systems. If instrumentation is provided, it must support the following data:

DMTF|Component Spare Association|001	REQUIRED
DMTF|Physical Extent|001	REQUIRED
DMTF|Aggregate Physical Extent|001	REQUIRED*
DMTF|Protected Space Extent|001	REQUIRED
DMTF|Aggregate Protected Space Extent|001	REQUIRED*
DMTF|Volume Set|001	REQUIRED
EventGeneration|DMTF^^Volume Set|001	REQUIRED
DMTF|Redundancy Group|001	REQUIRED
EventGeneration|DMTF^^Redundancy Group|001	REQUIRED

*One or the other of these groups is required, but not both.

10. Terms and Acronyms

This section briefly describes the platform management information technologies referenced in these guidelines and gives references to full definitions and descriptions of these technologies.

ACPI	Advanced Configuration and Power Interface; an interface between the OS and the hardware and BIOS designed to achieve independence between the hardware and the software.
API	Application Programming Interface.
asset management	The process of maximizing the use of assets to produce revenue while minimizing their overall costs. Manageable systems contribute to asset management by capturing inventory and tracking information, which enables organizations to analyze key cost variables of their technology purchases. The data gathered by manageable systems can assist asset management issues such as inventory, consolidating and rationalizing license issues, leasing considerations, analyzing training costs, analyzing software upgrades for volume purchasing plans, evaluating the cost-efficiency of outsourcing, and improving warranty usage.
attribute	A piece of information about a component; the building block of a MIF. An attribute describes a single characteristic of a manageable product or component. For example, the clock speed of a processor chip is an attribute of a group that describes that chip. A set of related attributes constitutes a group.
BINL	Boot Intervention Network Layer; extended DHCP service.
CI	Component Interface; the DMI layer used by component instrumentation.
CLSID	A class ID; a form of a UUID (GUID).
Common Information Model (CIM)	An object-oriented schema defined by the DMTF. CIM is an information model that provides a common way to describe and share management information enterprise-wide. It serves as the management information schema for WBEM, along with other specifications to be defined. CIM is designed to be extended for each management environment in which it is used.

CIM Object Manager (CIM OM)	Application level software that manages the CIM information model in a WBEM framework.
component	Any hardware, software, or firmware element contained in a computer system. A modem, printer, network interface card, spreadsheet software program, and the operating system are all components.
component instrumentation	The executable code that provides DMI (see entry) management functionality (attribute values, etc.) for a particular component. The component instrumentation uses the Component Interface (CI) to provide this information.
configuration management	The aspect of manageability dealing with tasks such as optimizing the user's configuration for given tasks, and discovering what hardware and components are in the system or server and ensuring compatibility among them.
CSC	The Customer Support Consortium is a standards body composed of technology companies and technology support providers which focuses on reducing support costs by establishing standards and business models to enable the sharing of solution information and to enable real-time collaboration.
Desktop Management Interface (DMI)	A platform management information framework, built by the DMTF, designed to provide manageability for desktop and server computing platforms by providing an interface that is: ■ Independent of any specific desktop operating system, network operating system, network protocol, management protocol, processor, or hardware platform. ■ Easy for vendors to implement. ■ Easily mapped to higher-level protocols.
Desktop Management Task Force (DMTF)	The DMTF is a standards organization comprised of companies from all areas of the computer industry. Its purpose is to create the standards and infrastructure for cost-effective management of PC systems.
DHCP	Dynamic Host Configuration Protocol; used to get information from the configuration server
DMI compliance	The DMTF owns the responsibility for the definition of DMI 2.0 compliance.

DMI Service Provider (DMI SP)	System software on the target system which provides services between the management application and component instrumentation. The DMI SP arbitrates access to component instrumentation and manages the MIF database.
event	Unsolicited information sent from a component to the management framework detailing an unusual circumstance or notable event. This information is forwarded to all management applications registered for event notification. For example, events can be sent when an error occurs or when a new version of a piece of software is installed. Component manufacturers determine which events will be related to their product and what information will be passed about the event.
fault notification	The ability of instrumented systems to detect when a component has exceeded system reliability watermark; such as maximum temperature. The system can then send out detailed alerts and can save critical data or files (if events occur that indicate the system is failing or may be about to fail). These alerts are also useful in building predictive failure models and developing a fault-tolerant computing structure.
Field Replaceable Unit (FRU)	Hardware component which can be replaced in the field, without swapping out the entire system.
group	A set of related attributes for a given component.
GUID	A globally unique identifier and a synonym for UUID.
IDL	Interface Description Language.
Internet Engineering Task Force (IETF)	The Internet Engineering Task Force is an industry standards group focusing on the evolution of the Internet architecture and the smooth operation of the Internet.
IID	An interface identifier and also a form of a UUID (GUID).
instrumentation	A common methodology and syntax for defining and reporting the management features and capabilities of all hardware, software, and attached peripherals of the system. Instrumentation enables management applications to understand and change the state of a system and to be notified of state changes.
interoperability	The ability of management applications, consoles, and manageable products to work together and make systems manageable.

inventory management	The aspect of manageability that addresses the need to identify system and software components. Instrumented systems can show a complete inventory of all components and subsystems—information that's highly useful for diagnosing problems remotely, tracking assets, and optimizing configurations and performance levels.
LOM (LAN on Motherboard)	The concept of integrating the LAN subsystem onto the motherboard.
manageability	The use of technologies and products to enhance the usability of desktops, mobile, and server platforms, and thus reduce the cost of deployment, ownership, and administration. Aspects of manageability include asset management, configuration management, fault management, inventory management, network management, performance management, and system management.
Managed Client	A desktop, mobile, or server system which provides management information and interfaces to a management application.
Management Application	Any program local to the managed client or resident on the management server, which addresses one or more of the aspects of manageability.
Management Information Base (MIB)	A collection of managed SNMP objects, residing in an information store.
Management Information Format (MIF)	An ASCII text file in the DMI architecture that describes a product's manageable features and attributes. The DMI maintains this information in a MIF database and makes it available to operating systems and management applications. The DMTF has specified MIF formats for a variety of system types and peripheral devices.
Management Object Format (MOF)	Language that describes CIM management information, based on the Interface Definition Language (IDL). MOF data is stored in an ASCII text file. DMTF is releasing MOF formats specifying the objects and associations necessary for management of systems, devices, applications, and networks
Management Interface (MI)	The DMI (see entry) layer between management applications and the Service Provider.
MIB Module	Collections of related SNMP objects are defined in MIB modules. These modules are written using a subset of OSI's Abstract Syntax Notation One (ASN.1).

MOF Specification	Refers to a collection of management information described in a manner that conforms to the MOF syntax.
MTFTP	Multicast TFTP; used to download NBP to many clients simultaneously.
network management	Along with server and system management, one of the three major components of managing a computing environment. Network management includes optimizing the performance, configuration, security, failure analysis, and repair of the infrastructure components in a LAN, WAN, or Internet/intranet. Infrastructure components include items such as switches, adapter cards, routers, bridges, and gateways, but not the end-nodes themselves.
OS	Operating System.
OSPM	Operating System Directed Power Management.
PCI PM	PCI Bus Power Management Interface Specification
power management	Technology that allows a system to consume less power when not in use and to be fully operational when awakened.
proxyDHCP	"Fake" DHCP; extended DHCP service.
Preboot Execution Environment (PXE)	A means by which agents can be loaded remotely onto systems to perform management tasks in the absence of a running OS. To enable the interoperability of clients and downloaded bootstrap programs, the client preboot code must provide a set of services for use by a downloaded bootstrap. It also must ensure certain aspects of the client state at the point in time when the bootstrap begins executing.

Reliability, Availability, Serviceability (RAS)	Three vital aspects of manageability: Reliability encompasses hardware and software that reduce failure rates and protect data if a system or server fails. Features that increase reliability include parity checking over the I/O bus, fail-safe processors, processor self-checking, and error-correcting code (ECC) memory. Availability includes hardware redundancy and software features that allow a system or server to continue operation in spite of a component failure. Key availability features range from redundant fans and power supplies to mirrored disk arrays to environmental monitoring. Serviceability refers to hardware and software that reduce downtime once a failure occurs. Serviceability includes features such as failure alerts and monitoring capabilities, as well as vendor support offerings such as on-site service and support hot-lines.
Remote Procedure Call (RPC)	An industry-standard method for communication with remote, networked systems. DMI Version 2.0 specifies RPCs as the standard mechanism for remote access to manageable systems.
remote wakeup (RWU)	The ability of a managed computer to be remotely awakened from a sleeping state.
server management	Dealing with server systems as a managed object, rather than as the manager of other managed objects.
Simple Network Management Protocol (SNMP)	The most widely used protocol for communicating management information for networks. SNMP focuses primarily on the network backbone; it is complemented by standards such as DMI, which extend manageability to end systems.
system management	Along with network and server management, one of the three major components of managing a computing environment. System management refers to controlling, configuring, installing, and monitoring the applications, servers, and clients in a distributed computing environment.
System Management (SM) BIOS	A standard interface to management software via data structures through which system attributes are reported.
TFTP	Trivial File Transport Protocol; used to download NBP from TFTP server.

Total Cost of Ownership (TCO)	The lifetime costs of a product, including the initial hardware and software purchase price as well as installation, service, support, upgrades, training, downtime, and other factors.
Trap	See **event**.
UNDI	Universal NIC Driver Interface; a PXE API that provides a device independent network interface to the NBP.
Unicode	A character encoding standard. Unicode characters are 2 octets each. When the first octet is zero, the second octet maps to the characters in ISO 8859-1.
UUID	A Universally Unique IDentifier generated by the GUIDGEN utility.
Web-Based Enterprise Management (WBEM)	A set of platform management information technologies originally proposed by BMC Software, Inc., Cisco Systems, Inc., Compaq Computer Corporation, Intel Corporation, and Microsoft Corporation to enable manageability over the Internet and corporate intranets. WBEM seeks to reduce the complexity and costs of enterprise management by allowing administrators to use any Web browser to manage disparate systems, networks, and applications. It envisions a common data model and Internet protocol that integrate existing standards like the DMI, SNMP, CMIP, and HTTP.
Win32 Driver Model (WDM)	A driver model based on the Windows NT driver model that is designed to provide a common set of I/O services and binary-compatible device drivers for both Windows NT and future Windows operating systems for specific classes of drivers. These driver classes include USB and IEEE 1394 buses, audio, still-image capture, video capture, and HID-compliant devices such as USB mice, keyboards, and joysticks.
Win32 Extensions Schema	Another name for the Win32 extensions to the CIM schema for Windows operating systems. For specifications on the Win32 Extensions Schema, see [WBEM].
Windows Management Interface (WMI)	WMI is an extension to WDM, developed for Windows NT 5.0 and Windows 98, to provide an operating system interface through which instrumented components can provide information and notifications. See [WBEM] for more information.

11. Information and Resource References

For **Wired for Management** information for developers:

http://developer.intel.com/ial/wfm/

For background on the **Wired for Management Initiative**:

http://www.intel.com/businesscomputing/tech/

Related Specifications

For a copy of the **ACPI** specification [ACPI]:

http://www.teleport.com\~acpi

For more information on **AMD's Magic Packet** technology [Magic Packet]:

http://www.amd.com/products/npd/overview/20212d.html.

For more information about the DMTF's **Common Information Model** [CIM]:

[1] Common Information Model (CIM) Specification, Version 2.0, March 3, 1998

http://www.dmtf.org

For more information about **DMI**:

[1] [DMI] Desktop Management Interface Specification, Version 2.00, March 27, 1996, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/specs.html

[2] Desktop Management Interface Compliance Guidelines, Version 1.0, September 11, 1995, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/specs.html

[3] Systems Standard Group Definitions, Approved Version 1.0, May 1, 1996, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/apps.html

[4] LAN Adapter Standard Groups Definition, Release Version 1.0, October 4, 1994, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/apps.html

[5] [MIF Guidelines] Desktop Management Task Force: Enabling your product for manageability with MIF files, Revision 1.0, November 1994, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/apps.html

[6] [DMI Conform] Desktop Management Task Force: Desktop Management Interface (DMI) 2.0 Conformance Guidelines, Version 1.0, Desktop Management Task Force, Inc..

http://www.dmtf.org/tech/specs.html

[7] [SNMP to DMI] Desktop Management Task Force: DMTF SNMP to DMI Mapping Standard, Desktop Management Task Force, Inc.

http://www.dmtf.org/tech/specs.html

For the **'Instantly Available' PC Power Management Design Guide,** Version 1.0, [PC POWER], including more information on **Operating System Directed Power Management**:

http://developer.intel.com/design/power/pcpower.htm

For additional information on the **Network Device Class Power Management Reference Specification** (Version 1.0) [NDC PM]:

http://www.microsoft.com/hwdev/ONNOW.HTM#pmSPECS

For the **PC'9x System Design Guide** [PC HDG], and for the **Hardware Design Guide Version 1.0 for Microsoft Windows NT Server** [HDG]:

http://developer.intel.com/solutions/tec/pc98.htm

http://www.microsoft.com/hwdev/desguid/default.htm

The **PCI Bus Power Management Interface Specification (PCI-PM)** [PCI PM] can be found at:

http://developer.intel.com/ial/powermgm/specs.htm

The PCI Engineering Change Request - Addition of 3.3Vaux signal to Connector can be found at:

http://developer.intel.com/design/power/pcipower.htm

For additional information on the **PCI Bus Power Management Interface Specification for CardBus Cards** and **PCI Bus Power Management Interface Specification for PCI to CardBus Bridges** published by the PCMCIA:

http://www.pc-card.com

For more information on the **Problem Resolution Standards** [SES/SIS] for standard trouble tickets, published by the Customer Support Consortium and the Desktop Management Task Force:

[1] Solution Exchange Standard, Version 1.0.

http://www.customersupport.org/Working.htm
http://www.dmtf.org/work/smwc.html

[2] Service Incident Exchange Standard, Version 1.0.

http://www.customersupport.org/Working.htm
http://www.dmtf.org/work/smwc.html

[3] MOF Addendum, Version 1.0rc.

http://www.customersupport.org/Working.htm
http://www.dmtf.org/work/smwc.html

Additional information about the **SMBIOS** specification [SMBIOS]:

http://developer.intel.com/ial/wfm/design/smbios

For **Web Based** Enterprise **Management** (WBEM) [WBEM]:

http://www.microsoft.com/management/wbem/

For the **Windows Management Interface** (WMI):

http://www.microsoft.com/hwdev/desinit/wmi.htm

For **SNMP information**, check the Internet Drafts available through the Internet Engineering Task Force:

http://www.ietf.cnri.reston.va.us/

For the **Preboot Execution Environment Specification**, version 2.0 [PXE]:

http://developer.intel.com/ial/wfm/wfmspecs.htm

For the **Boot Integrity Services Specification**, version 1.0 [BIS]:

http://developer.intel.com/ial/wfm/wfmspecs.htm

For the **Platform Event Trap Format Specification**, version 1.0 [PET]:

http://developer.intel.com/design/servers/ipmi

Standards' Groups - Contact Information

To contact the CSC:

Customer Support Consortium
101 Fifth Ave. N., Suite 1900
Seattle, WA, 98101
Phone: (206) 622-5200 x442
Fax: (206) 667-9181
URL: http://www.customersupport.org

To contact the DMTF:

Desktop Management Task Force
M/S JF2-53
2111 N.E. 25th Avenue
Hillsboro. OR 97124
Phone: (503) 264-9300
Fax: (503) 264-9027
Email: dmtf-info@dmtf.org

To contact the IETF:

Internet Engineering Task Force
URL: www.ietf.org

To contact the PCMCIA (Personal Computer Memory Card International Association):

PCMCIA
2655 North First St. Suite 209
San Jose, CA 95151
Phone: (408) 433-CARD
Fax: (408) 433-9558
URL: www.pc-card.com

Attachment A
UUIDs and GUIDs

Network Working Group	Paul J. Leach, Microsoft
INTERNET-DRAFT	Rich Salz, Open Group

<draft-leach-uuids-guids-00.txt>

Category: Informational

Expires August 24, 1997	February 24, 1997

Status of this Memo

Abstract

This specification defines the format of UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier). A UUID is 128 bits long, and if generated according to the one of the mechanisms in this document, is either guaranteed to be different from all other UUIDs/GUIDs generated until 3400 A.D. or extremely likely to be different (depending on the mechanism chosen). UUIDs were originally used in the Network Computing System (NCS) [1] and later in the Open Software Foundation's (OSF) Distributed Computing Environment [2].

This specification is derived from the latter specification with the kind permission of the OSF.

Introduction

This specification defines the format of UUIDs (Universally Unique IDentifiers), also known as GUIDs (Globally Unique IDentifiers). A UUID is 128 bits long, and if generated according to the one of the mechanisms in this document, is either guaranteed to be different from all other UUIDs/GUIDs generated until 3400 A.D. or extremely likely to be different (depending on the mechanism chosen).

Motivation

One of the main reasons for using UUIDs is that no centralized authority is required to administer them (beyond the one that allocates IEEE 802.1 node identifiers). As a result, generation on demand can be completely automated, and they can be used for a wide variety of purposes. The UUID generation algorithm described here supports very high allocation rates: 10 million per second per machine if you need it, so that they could even be used as transaction IDs.

UUIDs are fixed-size (128 bits), which is reasonably small relative to other alternatives. This fixed, relatively small, size lends itself well to sorting, ordering, and hashing of all sorts, storing in databases, simple allocation, and ease of programming in general.

Specification

A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. To be precise, the UUID consists of a finite bit space. Thus the time value used for constructing a UUID is limited and will roll over in the future (approximately at A.D. 3400, based on the specified algorithm). A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime to reliably identifying very persistent objects across a network.

The generation of UUIDs does not require that a registration authority be contacted for each identifier. Instead, it requires a unique value over space for each UUID generator. This spatially unique value is specified as an IEEE 802 address, which is usually already available to network-connected systems. This 48-bit address can be assigned based on an address block obtained through the IEEE registration authority. This section of the UUID specification assumes the availability of an IEEE 802 address to a system desiring to generate a UUID, but if one is not available section 4 specifies a way to generate a probabilistically unique one that can not conflict with any properly assigned IEEE 802 address.

Format

The following table gives the format of a UUID. The UUID consists of a record of 16 octets. The fields are in order of significance for comparison purposes, with “time_low” the most significant, and “node” the least significant.

Field	Data Type	Octet #	Note
time_low	unsigned 32 bit integer	0-3	The low field of the timestamp.
time_mid	unsigned 16 bit integer	4-5	The middle field of the timestamp.
time_hi_and_version	unsigned 16 bit integer	6-7	The high field of the timestamp multiplexed with the version number.
clock_seq_hi_and_reserved	unsigned 8 bit integer	8	The high field of the clock sequence multiplexed with the variant.
clock_seq_low	unsigned 8 bit integer	9	The low field of the clock sequence.
node	unsigned 48 bit integer	10-15	The spatially unique node identifier.

To minimize confusion about bit assignments within octets, the UUID record definition is defined only in terms of fields that are integral numbers of octets. The version number is in the most significant 4 bits of the time stamp (*time_hi*), and the variant field is in the most significant 3 bits of the clock sequence (*clock_seq_high*).

The timestamp is a 60-bit value. For UUID version 1, this is represented by Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582 (the date of Gregorian reform to the Christian calendar).

The following table lists currently defined versions of the UUID.

Msb0	Msb1	Msb2	Msb3	Version	Description
0	0	0	1	1	The version specified in this document.
0	0	1	0	2	Reserved for DCE Security version, with embedded POSIX UIDs.

The variant field determines the layout of the UUID. The structure of UUIDs is fixed across different versions within a variant, but not across variants; hence, other UUID variants may not interoperate with the UUID variant specified in this document. Interoperability of UUIDs is defined as the applicability of operations such as string conversion, comparison, and lexical ordering across different systems. The *variant* field consists of a variable number of the msbs of the *clock_seq_hi_and_reserved* field.

The following table lists the contents of the variant field.

Msb0	***Msb1***	***Msb2***	***Description***
0	-	-	Reserved, NCS backward compatibility.
1	0	-	The variant specified in this document.
1	1	0	Reserved, Microsoft Corporation GUID.
1	1	1	Reserved for future definition.

The clock sequence is required to detect potential losses of monotonicity of the clock. Thus, this value marks discontinuities and prevents duplicates. An algorithm for generating this value is outlined in the "Clock Sequence" section below.

The clock sequence is encoded in the 6 least significant bits of the *clock_seq_hi_and_reserved* field and in the *clock_seq_low* field.

The *node* field consists of the IEEE address, usually the host address. For systems with multiple IEEE 802 nodes, any available node address can be used. The lowest addressed octet (octet number 10) contains the global/local bit and the unicast/multicast bit and is the first octet of the address transmitted on an 802.3 LAN.

Depending on the network data representation, the multi-octet unsigned integer fields are subject to byte swapping when communicated between different endian machines.

The nil UUID is special form of UUID that is specified to have all 128 bits set to 0 (zero).

Algorithms for Creating a UUID

Various aspects of the algorithm for creating a UUID are discussed in the following sections. UUID generation requires a guarantee of uniqueness within the node ID for a given variant and version. Interoperability is provided by complying with the specified data structure. To prevent possible UUID collisions, which could be caused by different implementations on the same node, compliance with the algorithm specified here is required.

Clock Sequence

The clock sequence value must be changed whenever:

- the UUID generator detects that the local value of UTC has gone backward.
- the UUID generator has lost its state of the last value of UTC used, indicating that time *may* have gone backward; this is typically the case on reboot.

While a node is operational, the UUID service always saves the last UTC used to create a UUID. Each time a new UUID is created, the current *UTC* is compared to the saved value and if either the current value is less (the non-monotonic clock case) or the saved value was lost, the *clock sequence* is incremented modulo 16,384, thus avoiding production of duplicate UUIDs.

The *clock sequence* must be initialized to a random number to minimize the correlation across systems. This provides maximum protection against *node* identifiers that may move or switch from system to system rapidly. The initial value MUST NOT be correlated to the node identifier.

The rule of initializing the *clock sequence* to a random value is waived if, and only if, all of the following are true:

- The *clock sequence* value is stored in non-volatile storage.
- The system is manufactured such that the IEEE address ROM is designed to be inseparable from the system by either the user or field service, so that it cannot be moved to another system.
- The manufacturing process guarantees that only new IEEE address ROMs are used.
- Any field service, remanufacturing, or rebuilding process that could change the value of the clock sequence must reinitialize it to a random value.

In other words, the system constraints prevent duplicates caused by possible migration of the IEEE address, while the operational system itself can protect against non-monotonic clocks, except in the case of field service intervention. At manufacturing time, such a system may initialise the clock sequence to any convenient value.

System Reboot

There are two possibilities when rebooting a system:

- The UUID generator state - the last UTC, adjustment, and clock sequence - of the UUID service has been restored from non-volatile store.
- The state of the last UTC or adjustment has been lost.

If the state variables have been restored, the UUID generator just continues as normal. Alternatively, if the state variables cannot be restored, they are reinitialised, and the clock sequence is changed.

If the clock sequence is stored in non-volatile store, it is incremented; otherwise, it is reinitialized to a new random value.

Clock Adjustment

UUIDs may be created at a rate greater than the system clock resolution. Therefore, the system must also maintain an adjustment value to be added to the lower-order bits of the time. Logically, each time the system clock ticks, the adjustment value is cleared. Every time a UUID is generated, the current adjustment value is read and incremented atomically then added to the UTC time field of the UUID.

Clock Overrun

The 100 nanosecond granularity of time should prove sufficient even for bursts of UUID creation in high-performance multiprocessors. If a system overruns the clock adjustment by requesting too many UUIDs within a single system clock tick, the UUID service may raise an exception, handled in a system or process-dependent manner either by:

- terminating the requester
- reissuing the request until it succeeds
- stalling the UUID generator until the system clock catches up.

If the processors overrun the UUID generation frequently, additional node identifiers and clocks may need to be added.

UUID Generation

UUIDs are generated according to the following algorithm:

- Determine the values for the UTC-based timestamp and clock sequence to be used in the UUID, as described above.
- For the purposes of this algorithm, consider the timestamp to be a 60-bit unsigned integer and the clock sequence to be a 14-bit unsigned integer. Sequentially number the bits in a field, starting from 0 (zero) for the least significant bit.
- Set the *time_low* field equal to the least significant 32-bits (bits numbered 0 to 31 inclusive) of the time stamp in the same order of significance.
- Set the time*_mid* field equal to the bits numbered 32 to 47 inclusive of the time stamp in the same order of significance.
- Set the 12 least significant bits (bits numbered 0 to 11 inclusive) of the *time_hi_and_version* field equal to the bits numbered 48 to 59 inclusive of the time stamp in the same order of significance.
- Set the 4 most significant bits (bits numbered 12 to 15 inclusive) of the *time_hi_and_version* field to the 4-bit version number corresponding to the UUID version being created, as shown in the table above.
- Set the *clock_seq_low* field to the 8 least significant bits (bits numbered 0 to 7 inclusive) of the *clock sequence* in the same order of significance.
- Set the 6 least significant bits (bits numbered 0 to 5 inclusive) of the *clock_seq_hi_and_reserved* field to the 6 most significant bits (bits numbered 8 to 13 inclusive) of the *clock sequence* in the same order of significance.
- Set the 2 most significant bits (bits numbered 6 and 7) of the *clock_seq_hi_and_reserved* to 0 and 1, respectively.
- Set the *node* field to the 48-bit IEEE address in the same order of significance as the address.

String Representation of UUIDs

For use in human readable text, a UUID string representation is specified as a sequence of fields, some of which are separated by single dashes.

Each field is treated as an integer and has its value printed as a zero-filled hexadecimal digit string with the most significant digit first. The hexadecimal values a to f inclusive are output as lower case characters, and are case insensitive on input. The sequence is the same as the UUID constructed type.

The formal definition of the UUID string representation is provided by the following extended BNF:

```
UUID                = <time_low> “-” <time_mid> “-”
                      <time_high_and_version> “-”
                      <clock_seq_and_reserved>
                      <clock_seq_low> “-” <node>
time_low            = 4*<hexOctet>
```

```
time_mid               = 2*<hexOctet>
time_high_and_version  = 2*<hexOctet>
clock_seq_and_reserved = <hexOctet>
clock_seq_low          = <hexOctet>
node                   = 6*<hexOctet
hexOctet               = <hexDigit> <hexDigit>
hexDigit =
          “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” |

          | “a” | “b” | “c” | “d” | “e” | “f”
          | “A” | “B” | “C” | “D” | “E” | “F”
```

The following is an example of the string representation of a UUID:

```
f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

Comparing UUIDs

Consider each field of the UUID to be an unsigned integer as shown in the table in Section 3.1. Then, to compare a pair of UUIDs, arithmetically compare the corresponding fields from each UUID in order of significance and according to their data type. Two UUIDs are equal if and only if all the corresponding fields are equal. The first of two UUIDs follows the second if the most significant field in which the UUIDs differ is greater for the first UUID. The first of a pair of UUIDs precedes the second if the most significant field in which the UUIDs differ is greater for the second UUID.

Byte order of UUIDs

UUIDs may be transmitted in many different forms, some of which may be dependent on the presentation or application protocol where the UUID may be used. In such cases, the order, sizes, and byte orders of the UUIDs fields on the wire will depend on the relevant presentation or application protocol. However, it is strongly RECOMMENDED that the order of the fields conform with ordering set out in Section 3.1 above. Furthermore, the payload size of each field in the application or presentation protocol MUST be large enough that no information is lost in the process of encoding them for transmission.

In the absence of explicit application or presentation protocol specification to the contrary, a UUID is encoded as a 128-bit object, as follows: the fields are encoded as 16 octets, with the sizes and order of the fields defined in Section 3.1, and with each field encoded with the Most Significant Byte first (also known as network byte order).

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          time_high                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       time_mid                |         time_hi_and_version   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|clk_seq_hi_res |  clk_seq_low  |         node (0-1)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         node (2-5)                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Node IDs When No IEEE 802 Network Card Is Available

This section describes how to generate UUIDs for systems with no IEEE 802-compliant network card or other source of IEEE 802 addresses.

The ideal solution is to obtain a 47-bit cryptographic quality random number, and use it as the low 47 bits of the node ID, with the most significant bit of the first octet of the node ID set to 1. This bit is the unicast/multicast bit, which will never be set in IEEE 802 addresses obtained from network cards; hence, there can never be a conflict between UUIDs generated by machines with and without network cards.

If a system does not have a primitive to generate cryptographic quality random numbers, then generate one using one of the sources of randomness available on most systems. Such sources are system specific, but often include:

- the percent of memory in use
- the size of main memory in bytes
- the amount of free main memory in bytes
- the size of the paging or swap file in bytes
- free bytes of paging or swap file
- the total size of user virtual address space in bytes
- the total available user address space bytes
- the size of boot disk drive in bytes
- the free disk space on boot drive in bytes
- the current time
- the amount of time since the system booted
- the individual sizes of files in various system directories
- the creation, last read, and modification times of files in various system directories
- the utilization factors of various system resources (heap, etc.)
- current mouse cursor position
- current caret position
- current number of running processes, threads
- handles or IDs of the desktop window and the active window
- the value of stack pointer of the caller
- the process and thread ID of caller
- various processor architecture specific performance counters (instructions executed, cache misses, TLB misses)

Note that the above kinds of sources of randomness are used to seed cryptographic quality random number generators on systems without special hardware for their construction.

In addition, items such as the computer's name and the name of the operating system, while not strictly speaking random, will help differentiate the results from those obtained by other systems.

The exact algorithm to generate a node ID using these data is system specific, because both the data available and the functions to obtain them are often very system specific. However, assuming that one can concatenate all the values from the randomness sources into a buffer, and that a cryptographic hash function such as MD5 [3] is available, the following code will compute a node ID:

```
#include <md5.h>
#define HASHLEN 16

void GenNodeID(
      unsigned char * pDataBuf,     // concatenated “randomness
values”
      long cData,             // size of randomness values
      unsigned char NodeID[6]       // node ID
)
{
  int i, j, k;
  unsigned char Hash[HASHLEN];
  MD_CTX context;

  MDInit (&context);
  MDUpdate (&context, pDataBuf, cData);
  MDFinal (Hash, &context);

  for (j = 0; j<6; j++) NodeId[j]=0;
  for (i = 0,j = 0; i < HASHLEN; i++) {
          NodeID[j++] ^= Hash[i];
          if (j == 6) j = 0;
      };
      NodeID[0] |= 0x80;            // set the multicast bit
};
```

Other hash functions, such as SHA-1 [4], can also be used (in which case HASHLEN will be 20). The only requirement is that the result be suitably random – in the sense that the outputs from a set uniformly distributed inputs are themselves uniformly distributed, and that a single bit change in the input can be expected to cause half of the output bits to change.

Obtaining IEEE 802 Addresses

The following URL contains information on how to obtain an IEEE 802 address block. Cost is $1000 US.

http://stdsbbs.ieee.org/products/oui/forms/index.html

Security Considerations

It should not be assumed that UUIDs are hard to guess; they should not be used as capabilities.

Acknowledgements

This document draws heavily on the OSF DCE specification for UUIDs. Ted Ts'o provided helpful comments, especially on the byte ordering section which we mostly plagiarized from a proposed wording he supplied (all errors in that section are our responsibility, however).

References

[1] Lisa Zahn, et. al., Network Computing Architecture, Prentice Hall, Englewood Cliffs, NJ, 1990

[2] DCE: Remote Procedure Call, Open Group CAE Specification C309 ISBN 1-85912-041-5 28cm. 674p. pbk. 1,655g. 8/94

[3] R. Rivest, RFC 1321, “The MD5 Message-Digest Algorithm”, 04/16/1992.

[4] SHA Spec - TBD

Authors’ addresses

Paul J. Leach
Microsoft
1 Microsoft Way
Redmond, WA, 98052, U.S.A.
Email: paulle@microsoft.co

Rich Salz
The Open Group
11 Cambridge Center
Cambridge, MA 02142, U.S.A.
Email r.salz@opengroup.org

Appendix A – UUID Reference Implementation

```
/*
** Copyright (c) 1990- 1993, 1996 Open Software Foundation, Inc.
** Copyright (c) 1989 by Hewlett-Packard Company, Palo Alto, Ca. &
** Digital Equipment Corporation, Maynard, Mass.
** To anyone who acknowledges that this file is provided “AS IS”
** without any express or implied warranty: permission to use, copy,
** modify, and distribute this file for any purpose is hereby
** granted without fee, provided that the above copyright notices and
** this notice appears in all source code copies, and that none of
** the names of Open Software Foundation, Inc., Hewlett-Packard
** Company, or Digital Equipment Corporation be used in advertising
** or publicity pertaining to distribution of the software without
** specific, written prior permission. Neither Open Software
** Foundation, Inc., Hewlett-Packard Company, nor Digital Equipment
** Corporation makes any representations about the suitability of
** this software for any purpose.
*/
```

```
#include <sys/types.h>
#include <sys/time.h>

typedef unsigned long   unsigned32;
typedef unsigned short  unsigned16;
typedef unsigned char   unsigned8;
typedef unsigned char   byte;

#define CLOCK_SEQ_LAST              0x3FFF
#define RAND_MASK                   CLOCK_SEQ_LAST

typedef struct _uuid_t {
    unsigned32          time_low;
    unsigned16          time_mid;
    unsigned16          time_hi_and_version;
    unsigned8           clock_seq_hi_and_reserved;
    unsigned8           clock_seq_low;
    byte                node[6];
} uuid_t;

typedef struct _unsigned64_t {
    unsigned32          lo;
    unsigned32          hi;
} unsigned64_t;

/*
**  Add two unsigned 64-bit long integers.
*/
#define ADD_64b_2_64b(A, B, sum) \
    { \
        if (!(((A)->lo & 0x80000000UL) ^ ((B)->lo &
0x80000000UL))) { \
            if (((A)->lo&0x80000000UL)) { \
                (sum)->lo = (A)->lo + (B)->lo; \
                (sum)->hi = (A)->hi + (B)->hi + 1; \
            } \
            else { \
                (sum)->lo  = (A)->lo + (B)->lo; \
                (sum)->hi = (A)->hi + (B)->hi; \
            } \
        } \
        else { \
            (sum)->lo = (A)->lo + (B)->lo; \
            (sum)->hi = (A)->hi + (B)->hi; \
            if (!((sum)->lo&0x80000000UL)) (sum)->hi++; \
        } \
    }

/*
**  Add a 16-bit unsigned integer to a 64-bit unsigned integer.
*/
#define ADD_16b_2_64b(A, B, sum) \
    { \
        (sum)->hi = (B)->hi; \
        if ((B)->lo & 0x80000000UL) { \
            (sum)->lo = (*A) + (B)->lo; \
            if (!((sum)->lo & 0x80000000UL)) (sum)->hi++; \
        } \
        else \
            (sum)->lo = (*A) + (B)->lo; \
    }

/*
**  Global variables.
*/
static unsigned64_t     time_last;
static unsigned16       clock_seq;

static void
mult32(unsigned32 u, unsigned32 v, unsigned64_t *result)
{
    /* Following the notation in Knuth, Vol. 2. */
    unsigned32 uuid1, uuid2, v1, v2, temp;

    uuid1 = u >> 16;
    uuid2 = u & 0xFFFF;
    v1 = v >> 16;
    v2 = v & 0xFFFF;
    temp = uuid2 * v2;
    result->lo = temp & 0xFFFF;
    temp = uuid1 * v2 + (temp >> 16);
    result->hi = temp >> 16;
    temp = uuid2 * v1 + (temp & 0xFFFF);
    result->lo += (temp & 0xFFFF) << 16;
    result->hi += uuid1 * v1 + (temp >> 16);
}

static void
get_system_time(unsigned64_t *uuid_time)
{
    struct timeval tp;
    unsigned64_t utc, usecs, os_basetime_diff;

    gettimeofday(&tp, (struct timezone *)0);
    mult32((long)tp.tv_sec, 10000000, &utc);
    mult32((long)tp.tv_usec, 10, &usecs);
    ADD_64b_2_64b(&usecs, &utc, &utc);
    /* Offset between UUID formatted times and Unix formatted
times.
     * UUID UTC base time is October 15, 1582.
     * Unix base time is January 1, 1970. */
    os_basetime_diff.lo = 0x13814000;
    os_basetime_diff.hi = 0x01B21DD2;
    ADD_64b_2_64b(&utc, &os_basetime_diff, uuid_time);
}

/*
** See “The Multiple Prime Random Number Generator” by Alexander
** Hass pp. 368-381, ACM Transactions on Mathematical Software,
** 12/87.
*/
static unsigned32 rand_m;
static unsigned32 rand_ia;
static unsigned32 rand_ib;
static unsigned32 rand_irand;

static void
true_random_init(void)
{
    unsigned64_t t;
    unsigned16 seed;

 /* Generating our 'seed' value Start with the current time, but,
  * since the resolution of clocks is system hardware dependent
and
  * most likely coarser than our resolution (10 usec) we 'mixup'
the
  * bits by xor'ing all the bits together.  This will have the
effect
  * of involving all of the bits in the determination of the seed
  * value while remaining system independent.  Then for good
measure
  * to ensure a unique seed when there are multiple processes
  * creating UUIDs on a system, we add in the PID.
  */
    rand_m = 971;
    rand_ia = 11113;
    rand_ib = 104322;
    rand_irand = 4181;
    get_system_time(&t);
    seed  =  t.lo        & 0xFFFF;
    seed ^= (t.lo >> 16) & 0xFFFF;
    seed ^=  t.hi        & 0xFFFF;
    seed ^= (t.hi >> 16) & 0xFFFF;
    rand_irand += seed + getpid();
}

static unsigned16
true_random(void)
{
    if ((rand_m += 7) >= 9973)
        rand_m -= 9871;
    if ((rand_ia += 1907) >= 99991)
        rand_ia -= 89989;
    if ((rand_ib += 73939) >= 224729)
        rand_ib -= 96233;
    rand_irand = (rand_irand * rand_m) + rand_ia + rand_ib;
    return (rand_irand >> 16) ^ (rand_irand & RAND_MASK);
}

/*
**  Startup initialization routine for the UUID module.
*/
void
uuid_init(void)
{
    true_random_init();
    get_system_time(&time_last);
#ifdef NONVOLATILE_CLOCK
    clock_seq = read_clock();
#else
    clock_seq = true_random();
#endif
}

static int
time_cmp(unsigned64_t *time1, unsigned64_t *time2)
{
    if (time1->hi < time2->hi) return -1;
    if (time1->hi > time2->hi) return 1;
    if (time1->lo < time2->lo) return -1;
    if (time1->lo > time2->lo) return 1;
    return 0;
}

static void new_clock_seq(void)
{
    clock_seq = (clock_seq + 1) % (CLOCK_SEQ_LAST + 1);
    if (clock_seq == 0) clock_seq = 1;
#ifdef NONVOLATILE_CLOCK
    write_clock(clock_seq);
#endif
}

void uuid_create(uuid_t *uuid)
{
    static unsigned64_t     time_now;
    static unsigned16       time_adjust;
    byte                    eaddr[6];
    int                     got_no_time = 0;

    get_ieee_node_identifier(&eaddr);      /* TO BE PROVIDED */

    do {
        get_system_time(&time_now);
        switch (time_cmp(&time_now, &time_last)) {
        case -1:
            /* Time went backwards. */
            new_clock_seq();
            time_adjust = 0;
            break;
        case 1:
            time_adjust = 0;
            break;
        default:
            if (time_adjust == 0x7FFF)
                /* We're going too fast for our clock; spin. */
                got_no_time = 1;
            else
                time_adjust++;
            break;
        }
    } while (got_no_time);

    time_last.lo = time_now.lo;
    time_last.hi = time_now.hi;

    if (time_adjust != 0) {
        ADD_16b_2_64b(&time_adjust, &time_now, &time_now);
    }

    /* Construct a uuid with the information we've gathered
     * plus a few constants. */
    uuid->time_low = time_now.lo;
    uuid->time_mid = time_now.hi & 0x0000FFFF;
    uuid->time_hi_and_version = (time_now.hi & 0x0FFF0000) >> 16;
    uuid->time_hi_and_version |= (1 << 12);
    uuid->clock_seq_low = clock_seq & 0xFF;
    uuid->clock_seq_hi_and_reserved = (clock_seq & 0x3F00) >> 8;
    uuid->clock_seq_hi_and_reserved |= 0x80;
```

```
        memcpy(uuid->node, &eaddr, sizeof uuid->node);
    }
```

Attachment B Cross Mapper Considerations

This attachment establishes guidelines for constructing cross mapper solutions between CIM and DMI in a Windows environment. Current instrumentation solutions based on DMI and SNMP are receiving widespread deployment and use. As with all PC technologies, advances are made, and with Windows, come two new management technologies:

- Common Information Model Object Manager (CIMOM) - a management agent that manages objects based on the Common Information Model (CIM) schema.
- Windows Management Interface (WMI) - an instrumentation interface (an adjunct to the management agent) for instrumenting kernel-mode devices.

To make the best use of new management technologies delivered in Windows operating systems and to use the existing value of industry investments in managed systems, this attachment describes new technologies and sets out new recommendations to:

- Ensure the smooth migration and integration of existing instrumentation solutions with emerging instrumentation solutions on new Windows operating systems.
- Define how platforms with new Windows operating systems are made universally manageable by both existing DMI-based management applications and new CIMOM-based management applications.
- Allow the migration to new Windows instrumentation mechanisms while preserving the value of the industry's investment in DMI instrumentation.

Common Information Model

At the heart of the new management infrastructure is the Common Information Model (CIM). CIM is an object oriented schema for management, which the industry is standardizing through the efforts of the Desktop Management Task Force (DMTF). Representation of manageable data in Object Managers using CIM will provide powerful new capabilities for delivering management solutions.

Windows operating systems will include an object manager called the Common Information Model Object Manager (CIMOM). CIMOM will allow data to be published by a number of instrumentation means such as the DMI, WMI (Windows Management Interface), SNMP and other object providers.

Three components are needed to deliver instrumentation capability:

1. Instrumentation Code - Platform specific and OS specific software that provides the management data.

2. Management Data - The set of data used to describe and manage management components. This baseline defines a baseline set of the data that a management application can use to make platforms universally manageable.
3. Management Agent - provides means for management applications to access the management data and for instrumentation to make the management data available.

The remainder of this attachment discusses how these three areas are addressed for new Windows operating systems.

Windows Management Framework

DMI management applications should be able to access management information published through CIMOM, and CIMOM applications should be able to access the management information published through DMI. This requires two new elements in the management framework: the DMI Provider and the CIMOM Provider. Collectively, these are called "Cross-mappers". The overall architecture is described pictorially in Figure B-1 below.

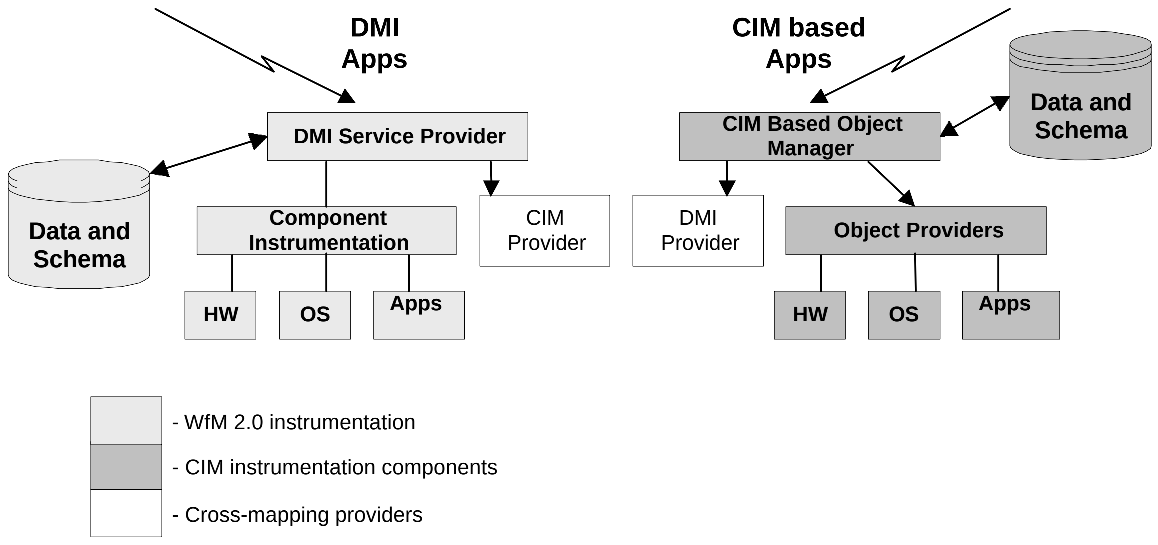

Figure B-1 Windows Management Architecture

The DMI Provider appears as an object provider to CIMOM. It treats DMI as a source of information with which to provide objects to the CIM schema. The DMTF CIM sub-committee is currently defining the mapping of DMTF Standard Groups to CIM objects. To make a DMI proprietary group visible through CIMOM, a MOF file must be registered in the CIM schema.

The CIMOM Provider appears to the DMI Service Provider as DMI-CI instrumentation. It treats CIMOM as a source of data with which to fill out the DMI groups. Data that is provided in certain standard CIM classes are automatically mapped to the appropriate DMI standard groups. Custom, or vendor-unique, CIM data are easily mapped to DMI with minimal developer effort. This mapping is done through the standard MIF file description and registration with the DMI Service Provider.

Instrumentation for Windows Operating Systems

Management Data

The specific requirements for instrumentation are based on platform type and are defined in the Desktop, Mobile, and Server checklists. These checklists list the standard DMTF groups that must be instrumented and deployed on Baseline-compliant systems.

Through the cross-mappers, this data can be provided by a combination of CIMOM Data Provider, WMI Data Provider, and DMI instrumentation code making the data visible to both CIMOM and DMI management applications

Events

Events can be delivered through either the DMI 2.00 SP or through CIMOM. The cross-mappers will provide DMI events to CIMOM and CIMOM events to DMI.

Dynamic Devices

This Baseline requires that instrumentation support devices that can be inserted and removed (for example, PC Card, USB, hot swap drives, 1394). Upon such insertion or removal, instrumentation must represent the associated management data additions or deletions. For CIMOM, this requires the instrumentation to add or delete object instances as appropriate when hardware is inserted or removed. The cross-mappers will automatically reflect these changes to both CIMOM and DMI applications.

Management Agent

To enable cross-mapping in DMI managed environments, the DMI 2.0 SP must be installed and active when the operating system boots. If the required data are also supplied by WMI or Object Providers, CIMOM must also be installed and active when the operating system boots.

Other management agents such as SNMP may also be loaded to support a broader range of management environments. SNMP “mappers” to DMI and to CIM may also be provided.

Instrumentation Code

So long as the required management data is provided, hardware and software vendors may write instrumentation code through one of three means:

1. Object Providers. Object providers are used to instrument anything other than kernel-mode managed components.
2. WMI. The Windows Management Interface is used to instrument kernel-mode managed components (that is, drivers).
3. DMI. DMI can be used to instrument both kernel-mode and user-mode managed components.

Cross-Mappers

The cross-mappers are instrumentation code. From the perspective of CIMOM, the DMI Provider instruments a source of data provided by DMI. From the perspective of DMI, the

CIMOM Provider instruments a source of data provided by CIMOM. To enable cross mapping between CIMOM and DMI, the cross-mappers must be installed on the managed platform. The application used on the Managed platform determines which cross-mapper is needed. This is a customer configuration choice.

Instrumentation Deployment

The instrumentation and associated drivers are both platform- and OS-dependent. They must be deployed by the platform vendor along with the platform. When the management agent is made active, its associated instrumentation must also be made available for use. This enables any compliant management application to access and manage the system via its instrumentation as soon as the system is up and running.

Windows Instrumentation References

The following represents some of the references, services, and tools available to help build instrumentation that works with Windows operating systems.

Web-Based Enterprise Management information

http://wbem.freerange.com
http://www.microsoft.com/management/wbem/
http:www.dmtf.org/work/cim.html

Wired for Management Baseline, Version 1.x, Intel Corporation.

http://www.intel.com/managedpc/

Attachment C
WfM DMI Mapping to CIM and SNMP

This appendix includes two tables identifying the relationship between the WfM Baseline DMI groups and the corresponding entities in SNMP and CIM. The first table identifies the relationship between the WfM groups and the SNMP Object Identifiers specified by [SNMP to DMI] The second table maps the WfM groups to the corresponding CIM Core and Common Model elements.

WfM Required Group	*SNMP Object Identifier*
DMTF|ComponentID|001	1.3.6.1.4.1.412.2.1.1
DMTF|Cooling Device|002	1.3.6.1.4.1.412.2.4.17
DMTF|Cooling Unit Global Table|001	1.3.6.1.4.1.412.2.4.67
DMTF|Disks Mapping Table|001	1.3.6.1.4.1.412.2.4.23
DMTF|Disks|002	1.3.6.1.4.1.412.2.4.22
DMTF|Dynamic States|001	1.3.6.1.4.1.412.2.8.2
DMTF|FRU|002	1.3.6.1.4.1.412.2.4.29
DMTF|General Information|001	1.3.6.1.4.1.412.2.4.1
DMTF|Keyboard|003	1.3.6.1.4.1.412.2.4.28
DMTF|Mass Store Array Info Table|001	1.3.6.1.4.1.412.2.4.46
DMTF|Mass Store Logical Drives Table|001	1.3.6.1.4.1.412.2.4.45
DMTF|Mass Store Mapping Table|001	1.3.6.1.4.1.412.2.4.43
DMTF|Mass Store Segment Table|001	1.3.6.1.4.1.412.2.4.44
DMTF|Memory Array Mapped Addresses|001	1.3.6.1.4.1.412.2.4.34
DMTF|Memory Device Mapped Addresses|001	1.3.6.1.4.1.412.2.4.36
DMTF|Memory Device|001	1.3.6.1.4.1.412.2.4.35
DMTF|Monitor Resolutions|002	1.3.6.1.4.1.412.2.6.2
DMTF|Motherboard|001	1.3.6.1.4.1.412.2.4.6
DMTF|Network Adapter 802 Port|001	1.3.6.1.4.1.412.2.2.1
DMTF|Network Adapter Driver|001	1.3.6.1.4.1.412.2.2.3
DMTF|Operating System|001	1.3.6.1.4.1.412.2.4.2
DMTF|Operational State|003	1.3.6.1.4.1.412.2.4.30
DMTF|Parallel Ports|003	1.3.6.1.4.1.412.2.4.10
DMTF|Partition|001	1.3.6.1.4.1.412.2.4.24
DMTF|Physical Container Global Table|001	1.3.6.1.4.1.412.2.4.63
DMTF|Physical Expansion Sites Table|001	1.3.6.1.4.1.412.2.4.65
DMTF|Physical Memory Array|001	1.3.6.1.4.1.412.2.4.33
DMTF|Physical Memory|002	1.3.6.1.4.1.412.2.4.7
DMTF|Pointing Device|001	1.3.6.1.4.1.412.2.8.5
DMTF|Portable Battery|001	1.3.6.1.4.1.412.2.8.1

DMTF|Power Supply|002	1.3.6.1.4.1.412.2.4.16
DMTF|Power Unit Global Table|001	1.3.6.1.4.1.412.2.4.66
DMTF|Processor|003	1.3.6.1.4.1.412.2.4.5
DMTF|Serial Ports|003	1.3.6.1.4.1.412.2.4.11
DMTF|System BIOS|001	1.3.6.1.4.1.412.2.4.3
DMTF|System Cache|002	1.3.6.1.4.1.412.2.4.9
DMTF|System Cache|003	1.3.6.1.4.1.412.2.4.9
DMTF|System Hardware Security|001	1.3.6.1.4.1.412.2.4.49
DMTF|System Memory Settings|001	1.3.6.1.4.1.412.2.4.71
DMTF|System Power Controls|001	1.3.6.1.4.1.412.2.4.51
DMTF|System Resource Device Info|001	1.3.6.1.4.1.412.2.4.38
DMTF|System Resource DMA Info|001	1.3.6.1.4.1.412.2.4.42
DMTF|System Resource I/O Info|001	1.3.6.1.4.1.412.2.4.40
DMTF|System Resource IRQ Info|001	1.3.6.1.4.1.412.2.4.41
DMTF|System Resource Memory Info|001	1.3.6.1.4.1.412.2.4.39
DMTF|System Resources 2|001	1.3.6.1.4.1.412.2.4.37
DMTF|System Resources Description|001	1.3.6.1.4.1.412.2.4.31
DMTF|System Resources|001	1.3.6.1.4.1.412.2.4.32
DMTF|System Slot|004	1.3.6.1.4.1.412.2.4.18
DMTF|System Slots|003	1.3.6.1.4.1.412.2.4.18
DMTF|Temperature Probe|001	1.3.6.1.4.1.412.2.4.54
DMTF|Video BIOS|001	1.3.6.1.4.1.412.2.4.20
DMTF|Video|002	1.3.6.1.4.1.412.2.4.19
DMTF|Voltage Probe|001	1.3.6.1.4.1.412.2.4.53

The following table identifies the relationship between the WfM Baseline DMI groups and the appropriate CIM Core and Common Model entities. Note that when accessing data through CIM Object Managers, the identified classes and properties must be provided to deliver the baseline data. The actual data can be instrumented by DMI, SNMP, or native interfaces of the CIM-based Object Managers, as long as they are appropriately mapped to the CIM Schema.

Some DMI attributes are not directly mapped in the CIM Schema and could be defined in extension schemas by subclassing CIM objects to add new properties. DMTF will continue to evolve the CIM Schema and improve the mapping coverage.

Group	Name	CIM Property

* **Indicates that the attribute is not directly mapped in the CIM Schema but could be mapped by appropriately defining properties in an object's instantiation or in an extension schema. All object names should be preceded with a "CIM_" to be valid. This was omitted throughout this table for brevity.**

BIOS Characteristic|003

BIOS Characteristic Index	BIOSFeatures and BIOSElements uniquely identified by their key properties.
BIOS Number	Characteristic is a property of BIOSFeature directly. (N/A)
BIOS Characteristic	BIOSFeature.Characteristics
BIOS Characteristic Description	BIOSFeature.CharacteristicDescriptions

ComponentID|001

Manufacturer	Product.Vendor, PhysicalElement.Manufacturer
Product	Product.Name, ManagedSystemElement.Name, OperatingSystem.OSType
Version	Product.Version, PhysicalElement.Version, SoftwareElement.Version, OperatingSystem.Version
Serial Number	Product.IdentifyingNumber, PhysicalElement.SerialNumber, SoftwareElement.SerialNumber
Installation	ManagedSystemElement.InstallDate
Verify	ManagedSystemElement.Status (Mapping is not 1-to-1 but similar info provided.)

Cooling Device|002

Cooling Device Table Index	CoolingDevice uniquely identified by its key properties.
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the CoolingDevice's physical “realization”. Alternately, the PhysicalElement(s) that realize the CoolingDevice could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
Cooling Unit Index	Redundancy information is provided via instantiation of an ExtraCapacityRedundancy or SpareGroup. Redundancy relationship between CoolingDevices indicated by instantiation of the RedundancyComponent association. RedundancyGroups have unique key properties
Cooling Device Type	Indicated by instantiating the appropriate CoolingDevice subclass.
Temperature Probe Index	Indicated by instantiation of a TemperatureSensor object and an AssociatedSensor relationship(between the TemperatureSensor and the CoolingDevice).

Cooling Unit Global Table|001- Is an instantiation of an ExtraCapacityRedundancy or SpareGroup

Cooling Unit Index	RedundancyGroups uniquely identified by their key properties.
Cooling Unit Status	Inherited from RedundancyGroup to ExtraCapacityRedundancy or SpareGroup, the RedundancyStatus property.

Disks Mapping Table|001

	Storage Type	Indicated by instantiating the appropriate MediaAccessDevice subclass and defining the StorageExtents accessed by the Device. These are identified by following the MediaPresent associations from MediaAccessDevice
	Disk Index	A Disk is an instantiation of a StorageExtent or one of its subclasses. It is related to a Partition as described for Partition Index, directly below.
	Partition Index	Partitions can be built on lower level StorageExtent(s). This is described using the BasedOn relationship between Storage Extents. Alternately, Partitions may be directly“realized” on PhysicalMedia. This relationship is indicated by the RealizesDiskPartition association, defined in the Physical Model.

Disks|002

	Storage Type	Indicated by instantiating the appropriate MediaAccessDevice subclass.
	Disk Index	Device uniquely identified by its key properties.
	Storage Interface Type	Information provided by following the ControlledBy association from the MediaAccessDevice to the Controller object that manages it. The Controller.ProtocolSupported property contains this data
	Interface Description	Inherited from ManagedSystemElement, the Description property.
	Media Loaded	Information provided by following the MediaPresent association between the MediaAccessDevice and StorageExtents accessed through the Device
	Removable Drive	Information found in Physical Package.Removeable for the Package which “realizes” the appropriate StorageExtent.
	Removable Media	MediaAccessDevice.Capabilities
	DeviceID	Inherited from LogicalDevice to MediaAccessDevice, the DeviceID property.
*	Logical Unit Number	Could be part of the LogicalDevice.DeviceID property for MediaAccessDevice.
*	Number of Physical Cylinders	(Not typically instrumented)
*	Number of Phys Sectors/Track	(Not typically instrumented)
*	Number of Physical Heads	(Not typically instrumented)
*	Phys Cylinder for Write Precompensation	(Not typically instrumented)
*	Physical Cylinder for Landing Zone	(Not typically instrumented)
*	Sector Size	(Not typically instrumented) Could instantiate a StorageExtent or one of its subclasses (such as PhysicalExtent or AggregatePExtent) using “sector size” as the StorageExtent.BlockSize.
	Total Physical Size	Calculated by multiplying StorageExtent.BlockSize by StorageExtent.NumberOfBlocks.
	Number of Current Bad Blocks	Obtained by totaling all StorageError objects associated with the StorageExtent

	or Sectors	(discovered by following the StorageDefect relationship).
	Partitions	Partitions can be built on lower level StorageExtent(s). This is described using the BasedOn relationship between Storage Extents. Alternately, Partitions may be directly “realized” on PhysicalMedia. This relationship is indicated by the RealizesDiskPartition association. To obtain the total number of Partitions, one must follow the appropriate associations from ComputerSystem to its component LogicalDevices and total the number of Partitions found.
	Physical Location	Indicated by instantiation of Location object(s) associated with a PhysicalMedia object or with a PhysicalPackage that “realizes” the MediaAccessDevice for this Disk (both PhysicalMedia and PhysicalPackage are subclasses of PhysicalElement). A Storage Extent is associated with a PhysicalMedia object using the Realizes relationship or one of its subclasses (for example, RealizesPExtent). Therefore, one can follow the Realizes relationship from the StorageExtent to the PhysicalElement and then follow the PhysicalElement's associations to the Location object.
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Disk's physical “realization”. Alternately, the PhysicalElement(s) that realize the Disk could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
*	Security Settings	

Dynamic States|001

	A/C Line Status	On- vs off-line information available by reading the Battery.TimeOnBattery property.
	Docking Status	Addressed using the Docked association between Chassis (a subclass of PhysicalElement).

FRU|002

	FRU Index	FRU is uniquely identified by its key properties.
	Device Group Index	The FRUPhysicalElements association identifies the Elements that compose the FRU.
	Description	FRU.Description
	Manufacturer	FRU.Vendor
*	Model	Recommend using the Product object to store Model-related information and using the ProductFRU relationship to associate the FRU and therefore obtain this data.
	Part Number	FRU.FRUNumber

	FRU Serial Number	FRU.IdentifyingNumber
	Revision Level	FRU.RevisionLevel
*	Warranty Start Date	(Not typically instrumented)
*	Warranty Duration	(Not typically instrumented)
	Support Phone Number	Support is associated with a Product, not a FRU. A FRU is applied to a Product. This relationship is indicated using the ProductFRU association. Support phone number would be found in SupportAccess.CommunicationInfo for the object where the CommunicationMode property = “Phone”.
	FRU Internet URL	Support is associated with a Product, not a FRU. A FRU is applied to a Product. This relationship is indicated using the ProductFRU association. Support URL information would be found in SupportAccess.CommunicationInfo for the object where the the CommunicationMode property = “Web Page”.

General Information|001

System Name	System.Name (CIM Name is part of the System key and should be created using the naming heuristic defined by the schema.)
System Location	Indicated by instantiation of Location object(s) associated with one or more PhysicalPackages (subclasses of PhysicalElement) that “realize” the System. A ComputerSystem is directly associated with a PhysicalPackage via an instantiation of the ComputerSystemPackage Dependency relationship (defined in the Physical Model). From the PhysicalPackage, one can follow its associations to Location object(s).
System Primary User Name	System.PrimaryOwnerName
System Primary User Phone	System.PrimaryOwnerContact
System Bootup Time	OperatingSystem.LastBootUpTime
System Date Time	OperatingSystem.LocalDateTime

Keyboard|003

Keyboard Layout	Keyboard.Layout
Keyboard Type	Inherited from ManagedSystemElement, the Description property.
Keyboard Connector Type	Must follow the Realizes relationship from the Keyboard to its PhysicalElement and then query its associated PhysicalConnectors. Alternately, one could enumerate the PhysicalConnectors for the PhysicalPackage(s) that realize the Keyboard's scoping ComputerSystem
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Keyboard's physical “realization”. Alternately, the PhysicalElement(s) that realize the Keyboard could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).

	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
	Security Settings	UserDevice.IsLocked (Mapping is not 1-to-1 but relevant data is provided.)

Memory Array Mapped Addresses|001

	Mem Array Mapped Addr Table Index	Memory and memory groupings uniquely defined by their key properties.
	Memory Array Index	Arrays, per se, are not modeled. However, system address info, etc is included in the basic Memory object definition. One could create both Memory Devices and Memory Arrays as instances of Volatile/NonVolatileStorage or Cache Memory. Explicit objects are not required. A Memory Array would be associated with (or built on) one or more Memory Devices using the “BasedOn” relationship.
	Mapped Range Starting Address	Memory.SystemStartingAddress
	Mapped Range Ending Address	Memory.SystemEndingAddress
	Partition ID	PhysicalMemory.BankNumber
*	Partition Width	
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.

Memory Device Mapped Addresses|001

	Mem Dev Mapped Addr Table Index	Memory and memory groupings uniquely defined by their key properties.
	Memory Device Set ID	Memory and memory groupings uniquely defined by their key properties. Device Sets are addressed as ReplacementSet objects in the Physical Model.
	Partition	PhysicalMemory.BankNumber
	Mapped Range Starting Address	Memory.SystemStartingAddress
	Mapped Range Ending Address	Memory.SystemEndingAddress
	Partition Row Position	PhysicalMemory.PositionInRow
	Interleave Position	PhysicalMemory.InterleavePosition
*	Data Depth	

Memory Device|001 - Can be VolatileStorage, NonVolatileStorage or CacheMemory

	Memory Device Table Index	Device uniquely identified by its key properties.
	Memory Array Index	Arrays, per se, are not modeled. One could create both Memory Devices and Memory Arrays as instances of Volatile/NonVolatileStorage or CacheMemory
	Device Locator	Information available by traversing the Realizes relationship from Logical to Physical Element and then following the PhysicalElementLocation association to a Location object.
	Bank Locator	Information available by traversing the Realizes relationship from Logical to Physical Element and then following the PhysicalElementLocation association to a Location object.
	Size	Calculated by multiplying the inherited (from StorageExtent) properties, BlockSize and NumberOfBlocks.

	Form Factor	PhysicalMemory.FormFactor
	Total Width	PhysicalMemory.TotalWidth
	Data Width	PhysicalMemory.DataWidth
	Memory Type	PhysicalMemory.MemoryType
*	Type Detail	
	Device Set	Modeled by instantiating the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
	Device Error Type	Memory.ErrorInfo
*	Error Granularity	
	Last Error Update	Memory.ErrorTime
	Error Operation	Memory.ErrorAccess
	Error Data Size	Memory.ErrorTransferSize
	Error Data	Memory.ErrorData
	Vendor Syndrome	Memory.AdditionalErrorData
	Device Error Address	Memory.ErrorAddress
	Array Error Address	Arrays, per se, are not modeled. One could create both Memory Devices and Memory Arrays as instances of Volatile/NonVolatileStorage or CacheMemory
	Error Resolution	Memory.ErrorResolution
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Memory's physical “realization”. Alternately, the PhysicalElements that realize the Memory could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.

Monitor Resolutions|002

	Monitor Resolution Index	The MonitorResolution object is uniquely defined by its key properties.
	Horizontal Resolution	MonitorResolution.HorizontalResolution
	Vertical Resolution	MonitorResolution.VerticalResolution
	Refresh Rate	MonitorResolution.RefreshRate
	Vertical scan mode	MonitorResolution.ScanMode
	Minimum Monitor Refresh Rate	MonitorResolution.MinRefreshRate
	Maximum Monitor Refresh Rate	MonitorResolution.MaxRefreshRate

Motherboard|001- Modeled as a Card (HostingBoard boolean=TRUE), a subclass of PhysicalElement

	Number of Expansion Slots	Card.NumberOfSlots
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Card that represents the Motherboard.
*	Operational Group Index	(Not typically instrumented) Since a Motherboard is a hardware container only, it does not have the operational characteristics of a LogicalElement.

Network Adapter 802 Port|001

	Port Index	Information available by traversing the Realizes relationship from NetworkAdapter to its PhysicalElement(s) and then following the PhysicalElementLocation association to a Location object which could describe the “port index”. Alternately, the PhysicalElement could have a Connector, found by traversing the ConnectorOnPackage association. This Connector's Location could identify “port index”.
	Permanent Network Address	NetworkAdapter.PermanentAddress
	Current Network Address	NetworkAdapter.NetworkAddresses
	Connector Type	Must follow the Realizes relationship from the NetworkAdapter Device to its Physical Element and then query its associated PhysicalConnectors.
	Data Rate	NetworkAdapter.Speed
*	Total Packets Transmitted	
	Total Bytes Transmitted	EthernetAdapter.OctetsTransmitted
*	Total Packets Received	
	Total Bytes Received	EthernetAdapter.OctetsReceived
	Total Transmit Errors	Sum of all transmit error counts in EthernetAdapter - for example, InternalMACTransmitErrors or CarrierSenseErrors.
	Total Receive Errors	Sum of all receive error counts in EthernetAdapter - For example, AlignmentErrors or FCSErrors.
*	Total Host Errors	
	Total Wire Errors	Sum of all wire error counts in EthernetAdapter - For example, ExcessiveCollisions.

Network Adapter Driver|001

	Driver Index	Driver should be an instantiation of a SoftwareElement and uniquely identified by its key properties.
	Driver Software Name	Instantiation of a SoftwareElement, related to the NetworkAdapter Device using the DeviceSoftware Dependency association. SoftwareElement has a Name property.
	Driver Software Version	SoftwareElement.Version
	Driver Software Description	Inherited from ManagedSystemElement, the Description property.
	Driver Size	Size of memory to run, or disk space to install is specified using the Application Model's MemoryCheck and DiskSpaceCheck classes. These are associated with SoftwareElement using the SoftwareElementChecks relationship.
*	Driver Interface Type	
*	Driver Interface Version	
*	Driver Interface Description	

Network Adapter Hardware|001

Partial,	Network Topology	For Ethernet and TokenRingAdapters, information will be provided in the MaximumSpeed property (to be defined in V2.1).
Future *****	Transmission Capability	
	Network Adapter RAM Size	Memory can be associated with a LogicalDevice using the AssociatedMemory relationship.
	Bus Type	Indicated by traversing the Realizes relationship from the NetworkAdapter to its PhysicalElement(s) - which could be a Card object directly or contained by a Card. Cards are inserted into other Cards (for example, a MotherBoard) at a SlotID. Also, the relationship between a Card and a Slot is indicated via the CardInSlot Dependency association. The Slot object has various properties such as “bus type” (Slot.ConnectorType).
	Bus Width	Indicated by traversing the Realizes relationship from the NetworkAdapter to its PhysicalElement(s) - which could be a Card object directly or contained by a Card. Cards are inserted into other Cards (for example, a MotherBoard) at a SlotID. Also, the relationship between a Card and a Slot is indicated via the CardInSlot Dependency association. The Slot object has various properties such as “bus width” (Slot.DataWidth).

Operating System|001

	Operating System Index	OperatingSystem uniquely identified by its key properties.
	Operating System Name	OperatingSystem.Name
	Operating System Version	Product.Version, OperatingSystem.Version
	Primary Operating System	InstalledOS.PrimaryOS, where InstalledOS associates a Computer and an OperatingSystem.
	OS Boot Device Storage Type	UnitaryComputerSystem.LastLoadInfo contains the Device key of the initial load device. Alternately, the currently executing OS is indicated using the RunningOS association (from ComputerSystem to OperatingSystem). The FileSystem from which the OS boots is indicated using the BootOSFromFS association. The StorageExtent on which the FileSystem resides is indicated by the ResidesOnExtent association.
	OS Boot Device Index	Detail as above.
	OS Boot Partition Index	The currently executing OS is indicated using the RunningOS association (from ComputerSystem to OperatingSystem). The FileSystem from which the OS boots is is indicated using the BootOSFromFS association. The StorageExtent on which the FileSystem resides is indicated by the ResidesOnExtent association. This StorageExtent could be a Partition.

	Operating System Description	Inherited from ManagedSystemElement, the Description property.

Operational State|003

	Operational State Instance Index	Operational state data included in the properties of LogicalDevice. No separate object.
	Device Group Index	LogicalDevice uniquely identified by its key properties. Alternately, the Physical Elements that should be replaced as a "set" are identified using the Replacement Set object and the ParticipatesInSet association (defined in the Physical Model). A LogicalDevice is associated with a PhysicalElement using the Realizes relationship.
	Operational Status	LogicalDevice.StatusInfo
*	Usage State	N/A for most Devices.
	Availability Status	LogicalDevice.Availability
*	Administrative Status	N/A for most Devices. UserDevice has IsLocked boolean.
Future	Fatal Error Count	Errors and events to be addressed in future CIM release.
Future	Major Error Count	Errors and events to be addressed in future CIM release.
Future	Warning Error Count	Errors and events to be addressed in future CIM release.
	Current Error Status	Inherited from ManagedSystemElement, the Status property for the LogicalDevice.
Future	Device Predicted Failure Status	To be addressed in V2.1 with the addition of "Failure Predicted" to the Status property enumeration of ManagedSystemElement.

Parallel Ports|003

	Parallel Port Index	ParallelController uniquely identified by its key properties.
	Parallel Base I/O Address	MemoryMappedIO.StartingAddress - The MemoryMappedIO object is found by following the AllocatedResource relationships from the ParallelController instance.
	IRQ Used	IRQ.IRQNumber - The IRQ object is found by following the AllocatedResource relationships from the ParallelController instance.
	Logical Name	Inherited from Logical Device to ParallelController, the DeviceID property.
	Connector Type	Must follow the Realizes relationship from the ParallelController to its Physical Element and then query its associated PhysicalConnectors.
	Connector Pinout	PhysicalConnector.ConnectorPinout - The appropriate PhysicalConnector is found by following the Realizes relationships from the ParallelController instance.
	DMA Support	ParallelController.DMASupport
	Parallel Port Capabilities	ParallelController.Capabilities
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
*	Parallel Port Security Settings	

Partition|002

	Partition Index	Partition uniquely identified using its key properties.
	Partition Name	Inherited from LogicalDevice to Partition, the DeviceID property.
	Partition Size	Inherited from StorageExtent, the product of BlockSize and NumberOfBlocks.
	Free Space	FileSystem.AvailableSpace - File Systems have a Dependency relationship (ResidesOnExtent) to StorageExtent. Extents do not have “free space” - File Systems do.
*	Partition Label	
	File System	As above, File Systems have a Dependency relationship - ResidesOnExtent - to StorageExtents.
	Compressed	FileSystem.CompressionMethod
	Encrypted	FileSystem.EncryptionMethod
	Number Of Disks Occupied	Partitions can be built on lower level StorageExtent(s). This is described using the BasedOn relationship between Storage Extents. Alternately, Partitions may be directly realized on PhysicalMedia. This relationship is indicated by the RealizesDiskPartition association, defined in the Physical Model. To obtain the number of disks occupied follow these associations and total the number of PhysicalMedia found.

Physical Container Global Table|002

Container or Chassis Type	Chassis.ChassisType
Asset Tag	Inherited from Physical Element, the Tag property for the Chassis.
Chassis Lock Present	Rack.LockPresent, Chassis.LockPresent
Bootup State	The container/enclosure for a ComputerSystem is indicated using the ComputerSystem Package relationship. A ComputerSystem and/or its OperatingSystem have a “bootup state” inherited from ManagedSystemElement, the Status property. (Not a 1-to-1 mapping but similar information is provided.)
Power State	Must follow the SystemDevice associations to determine the Device components of the Computer System. Can then check the PowerSupply or Battery Devices' Status properties, inherited from ManagedSystemElement. (Not a 1-to-1 mapping, but similar info provided.)
Thermal State	An enclosure or Chassis (PhysicalPackage) can have an associated Temperature Sensor LogicalDevice. This is indicated by the PackageTempSensor Dependency association. Thermal state can be determined by analyzing the properties of the associated TemperatureSensor object.
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the appropriate PhysicalPackage or its subclasses.

Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object. Could check the operational data for both the ComputerSystem's aggregated Devices and for its OperatingSystem
Container Index	PhysicalElement and its subclasses uniquely identified by their key properties.
Container Name	Inherited from ManagedSystemElement, the Name property.
Container Location	Indicated by instantiation of Location object(s) associated with one or more PhysicalPackages, that make up the ComputerSystem hardware. A ComputerSystem is associated with its enclosure/container (ie, PhysicalPackage) using the ComputerSystemPackage Dependency relationship.
Container Security Status	Rack.SecurityBreach, Chassis.SecurityBreach

Physical Memory Array|001 - Can be VolatileStorage, NonVolatileStorage or CacheMemory

Memory Array Table Index	Memory and memory groupings uniquely defined by their key properties.
Memory Array Location	Information available by traversing the Realizes relationship from Logical to Physical Element and then following the PhysicalElementLocation association to a Location object.
Memory Array Use	Information could be placed in the Purpose property, inherited from StorageExtent to all Memory classes. Alternately, this data is available by examining the Memory subclass that is instantiated (Volatile Storage, NonVolatile Storage, Cache Memory) and checking whether this Memory is associated with a Device or not (for example, video memory would be associated with a Video Controller using the AssociatedMemory relationship).
Maximum Memory Capacity	MemoryCard.MaxMemoryCapacity
Number of Memory Device Sockets	Inherited from Card to MemoryCard, the NumberOfSlots property.
Numb of Mem Dev Sockets Used	Addressed by enumerating the MemoryOnCard association between Memory components and a HostingBoard or MemoryCard.
Memory Error Correction	Memory.ErrorMethodology, an override of StorageExtent's ErrorMethodology property.
Array Error Type	Memory.ErrorInfo
Last Error Update	Memory.ErrorTime
Error Operation	Memory.ErrorAccess
Error Data Size	Memory.ErrorTransferSize
Error Data	Memory.ErrorData
Vendor Syndrome	Memory.AdditionalErrorData
Error Address	Memory.ErrorAddress
Error Resolution	Memory.ErrorResolution

FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Memory's physical “realization”. Alternately, the PhysicalElement(s) that realize the Memory (MemoryCards or PhysicalMemory) could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.

Pointing Device|001

Pointing Device Type	Indicated by instantiation of the correct subclass of the PointingDevice class.
Pointing Device Interface	Information indicated by the type of Controller associated with the PointingDevice as well as PhysicalConnector information from the Physical Model. Must follow the Realizes relationship from the PointingDevice to its PhysicalElement(s) and then query its associated PhysicalConnectors. The associated Controller is found by following the ControlledBy association. Could also enumerate the PhysicalConnectors for the PhysicalPackage(s) that realize the Device's scoping ComputerSystem.
Pointing Device IRQ	IRQ.IRQNumber - The IRQ object is found by following the AllocatedResource relationships from the PointingDevice (or subclass) instance.
Pointing Device Buttons	PointingDevice.NumberOfButtons
Pointing Device Port Name	Information indicated by the Name property of the Controller associated with the Pointing Device. This is found by following the ControlledBy association from the PointingDevice.
Pointing Device Driver Name	Instantiation of a SoftwareElement, related to the PointingDevice using the DeviceSoftware Dependency association. SoftwareElement has a Name property.
Pointing Device Driver Version	SoftwareElement.Version
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Device's physical “realization”. Alternately, the PhysicalElement(s) that realize the PointingDevice could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
Security Settings	UserDevice.IsLocked (Mapping is not 1-to-1 but relevant data is provided.)

Portable Battery|001

Portable Battery Index	Battery uniquely identified by its key properties.

	Portable Battery Location	Indicated in an instantiation of a Location object that is associated with a PhysicalElement that represents the Battery. One can follow the Realizes relationship from the Battery LogicalDevice to the PhysicalElement, and then follow the PhysicalElementLocation association to the Location object.
	Portable Battery Manufacturer	Product.Vendor, PhysicalElement.Manufacturer
*	Portable Batt Manufacture Date	(Not typically instrumented)
	Portable Battery Serial Number	Product.IdentifyingNumber, PhysicalElement.SerialNumber
	Portable Battery Device Name	Inherited from Managed System Element to Battery, the Name property.
	Portable Batt Device Chemistry	Battery.Chemistry
	Portable Batt Design Capacity	Battery.DesignCapacity
	Portable Batt Design Voltage	Battery.DesignVoltage
	Smart Battery Version	Battery.SmartBatteryVersion
	Full Charge Capacity	Battery.FullChargeCapacity
*	Remaining Capacity	
*	Maximum Error	
	Portable Battery Charging Status	Battery.BatteryStatus
	Remaining Battery Time	Battery.EstimatedRunTime
	Remaining Time to Full Battery	Battery.TimeToFullCharge
	Power Unit Index	Batteries and their redundancy relationships uniquely identified by their key properties. A Battery's participation in a RedundancyGroup is indicated by the RedundancyComponent association.

Power Supply|002

	Power Supply Index	PowerSupply uniquely identified by its key properties.
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Device's physical “realization”. Alternately, the PhysicalElement(s) that realize the PowerSupply could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association.
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
	Power Unit Index	RedundancyGroups uniquely identified by their key properties. Either a Spare or an ExtraCapacityRedundancyGroup would be instantiated.
Partial	Power Supply Type	UPS or Battery identified by instantiation of an UninterruptablePowerSupply or Battery Device. Whether the Supply is Linear or Switching is identified by the IsSwitchingSupply boolean. (Most but not all info is mapped.)
	Input Voltage Capability Description	Info could be placed in PowerSupply.Description, inherited from Managed System Element.
	Range 1 Input Voltage Low	PowerSupply.Range1InputVoltageLow
	Range 1 Input Voltage High	PowerSupply.Range1InputVoltageHigh

Range 1 Voltage Probe Index	The AssociatedSupplyVoltageSensor relationship indicates the VoltageSensor(s) monitoring the PowerSupply. Whether monitoring Range 1 or 2 indicated by the MonitoringRange property of the association.
Range 1 Elect Current Probe Index	The AssociatedSupplyCurrentSensor relationship indicates the CurrentSensor(s) monitoring the PowerSupply. Whether monitoring Range 1 or 2 indicated by the MonitoringRange property of the association.
Range 2 Input Voltage Low	PowerSupply.Range2InputVoltageLow
Range 2 Input Voltage High	PowerSupply.Range2InputVoltageHigh
Range 2 Voltage Probe Index	The AssociatedSupplyVoltageSensor relationship indicates the VoltageSensor(s) monitoring the PowerSupply. Whether monitoring Range 1 or 2 indicated by the MonitoringRange property of the association.
Range 2 Current Probe Index	The AssociatedSupplyCurrentSensor relationship indicates the CurrentSensor(s) monitoring the PowerSupply. Whether monitoring Range 1 or 2 indicated by the MonitoringRange property of the association.
Active Input Voltage Range	PowerSupply.ActiveInputVoltage
Input Voltage Range Switching	PowerSupply.TypeOfRangeSwitching
Range 1 Input Frequency Low	PowerSupply.Range1InputFrequencyLow
Range 1 Input Frequency High	PowerSupply.Range1InputFrequencyHigh
Range 2 Input Frequency Low	PowerSupply.Range2InputFrequencyLow
Range 2 Input Frequency High	PowerSupply.Range2InputFrequencyHigh
Total Output Power	PowerSupply.TotalOutputPower

Power Unit Global Table|001- Is an instantiation of an ExtraCapacityRedundancy or SpareGroup

Power Unit Index	RedundancyGroups uniquely identified by their key properties.
Power Unit Redundancy Status	Inherited from RedundancyGroup to ExtraCapacityRedundancy or SpareGroup, the RedundancyStatus property.

Processor|004

Processor Index	Processor uniquely identified by its key properties.
Processor Type	Processor.Role (however, Role is not enumerated).
Processor Family	Processor.Family
Processor Version Information	PhysicalElement.Version / Could also be included in the Processor.Name property, inherited from ManagedSystemElement.
Maximum Speed	Processor.MaxClockSpeed
Current Speed	Processor.CurrentClockSpeed
Processor Upgrade	Processor.UpgradeMethod

	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Processor's physical “realization”. Alternately, the PhysicalElement(s) that realize the Processor could be associated with other Elements that are to be replaced as a“set”, using the ReplacementSet object and the ParticipatesInSet association.
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
	Level 1 Cache Index	Info available by instantiating a CacheMemory object and an AssociatedMemory relationship between the Processor and CacheMemory. CacheMemory.Level indicates Level 1, 2 or 3
	Level 2 Cache Index	As for L1 Cache, above.
	Level 3 Cache Index	As for L1 Cache, above.

Serial Ports|003

	Serial Port Index	SerialController uniquely identified by its key properties.
	Serial Base I/O Address	MemoryMappedIO.StartingAddress - The MemoryMappedIO object is found by following the AllocatedResource relationships from the SerialController instance.
	IRQ Used	IRQ.IRQNumber - The IRQ object is found by following the AllocatedResource relationships from the SerialController instance.
	Logical Name	Inherited from Logical Device to SerialController, the DeviceID property.
	Connector Type	Must follow the Realizes relationship from the SerialController to its PhysicalElement and then query its associated PhysicalConnectors.
	Maximum Speed	SerialController.MaxBaudRate
	Serial Port Capabilities	SerialController.Capabilities
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
*	Serial Port Security Settings	

System BIOS|001

	BIOS Index	BIOSElement uniquely identified by its key properties.
	BIOS Manufacturer	Inherited from SoftwareElement to BIOSElement, the Manufacturer property.
	BIOS Version	Inherited from SoftwareElement to BIOSElement, the Version property.
	BIOS ROM Size	Calculated by following the BIOSLoadedInNV relationship from the BIOSElement to a NonVolatileStorage Extent. ROM Size=BIOSLoadedInNV.EndingAddress - StartingAddress.
*	BIOS Starting Address	
*	BIOS Ending Address	
*	BIOS Loader Version	Could be addressed by creating a SoftwareElement specific to BIOS loading.
*	BIOS Release Date	
	Primary BIOS	BIOSElement.PrimaryBIOS

System Cache|003

	System Cache Index	CacheMemory uniquely identified by its key properties.
	System Cache Level	CacheMemory.Level
	System Cache Speed	PhysicalMemory.Speed
	System Cache Size	Inherited from StorageExtent to CacheMemory, cache size is the product of the properties, BlockSize (ie, bytes) and NumberOfBlocks.
	System Cache Write Policy	CacheMemory.WritePolicy
	System Cache Error Correction	Inherited from StorageExtent to CacheMemory, the ErrorMethodology property.
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Memory's physical “realization”. Alternately, the PhysicalElement(s) that realize the Memory (MemoryCards or PhysicalMemory) could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
	System Cache Type	CacheMemory.CacheType
	Line Size	CacheMemory.LineSize
*****	Volatility	(Not typically instrumented)
	Replacement Policy	CacheMemory.ReplacementPolicy
	Read Policy	CacheMemory.ReadPolicy
	Flush Timer	CacheMemory.FlushTimer
	Associativity	CacheMemory.Associativity

System Hardware Security|001

*****	Power-On Password Status	(Not typically instrumented)
	Keyboard Password Status	Keyboard.Password
*****	Administrator Password Status	(Not typically instrumented)
	Front Panel Reset Status	UnitaryComputerSystem.ResetCapability

System Memory Settings|001

	Total Physical Memory	Sum of all VolatileStorage Memory extents associated with a ComputerSystem. Could also be determined by analyzing the PhysicalMemory associated with Card(s), contained in one or more Chassis(s) that represent a ComputerSystem's hardware.
	Free Physical Memory	OperatingSystem.FreePhysicalMemory
	Total Size of Paging Files	OperatingSystem.SizeOfPagingFiles
	Total Free Space in Paging Files	OperatingSystem.FreeSpaceInPagingFiles
	Total Virtual Memory	OperatingSystem.TotalVirtualMemorySize
	Free Virtual Memory	OperatingSystem.FreeVirtualMemory

System Power Controls|001

	Power Control Request	UnitaryComputerSystem.SetPowerState method
	Timed Power-On Available	UnitaryComputerSystem.PowerManagementCapabilities

	Time to Next Scheduled Power-On	Time parameter for the UnitaryComputerSystem.SetPowerState method

System Resource DMA Info|001

	System Resource DMA Info Index	DMA uniquely identified by its key properties.
	DMA Transfer Width	DMA.TransferWidths
	DMA Address Size	DMA.AddressSize
	DMA Maximum Transfer Size	DMA.MaxTransferSize
*	DMA Transfer Preference	
Future	Bus Master	In V2.1, will be addressed by subclassing the AllocatedResource association for DMA (AllocatedDMA) and adding a property, BusMaster (boolean).
	Byte Mode	DMA.ByteMode
	Word Mode	DMA.WordMode
	Channel Timing	DMA.ChannelTiming
	Type-C Timing	DMA.TypeCTiming

System Resource Device Info|001

	Resource User	LogicalDevice.DeviceID
*	Device ID	Could be used in constructing the LogicalDevice.DeviceID.
	Device Serial Number	PhysicalElement.SerialNumber
*	Logical Device ID - Class Code	Could be used in constructing the LogicalDevice.DeviceID.
*	Device Flags	
*	Device Number	Could be used in constructing the LogicalDevice.DeviceID.
*	Device Number - Function Number	Could be used in constructing the LogicalDevice.DeviceID.
	Bus Type	Information available via the Physical Model - following the Realizes relationship from the LogicalDevice to its hardware/PhysicalElements. Either the Physical Element is a Card object or is contained by a Card. Cards are inserted into other Cards (for example, a HostingBoard) at a SlotID. Also, the relationship between a Card and a Slot is indicated via the CardInSlot Dependency association. Slots have various properties including ConnectorType and SlotID.
*	CM Reserved	

System Resource I/O Info|001

	System Resource I/O Info Index	MemoryMappedIO uniquely identified by its key properties.
*	I/O Decode	

System Resource IRQ Info|001

	System Resource IRQ Info Index	IRQ uniquely identified by its key properties.
	Trigger Type	IRQ.TriggerType
	Trigger Level	IRQ.TriggerLevel

System Resource Memory Info|001

	System Resource Memory Info Index	Memory and its subclasses uniquely identified by their key properties.
*	ISA/PCMCIA Range Descriptor	
*	EISA Range Descriptor	
*	Decode Support	
	Cacheable	VolatileStorage.Cacheable
	Cache Type	VolatileStorage.CacheType
	Read-Write	Inherited from StorageExtent to Memory and its subclasses, the Access property.

System Resources 2|001

	System Resources Index	N/A - This is a DMI row identifier only.
	Resource User	LogicalDevice.DeviceID
	Resource Set	N/A - Indicated as separate AllocatedResource associations from LogicalDevices to SystemResources. Regarding memory, there is an AssociatedMemory Dependency relationship between a Device and Memory. A “set” could be modeled as Setting objects, grouped in a Configuration.
	Resource Assignment	IRQ.Availability, DMA.Availability (“Temporary Assigment” is not an enumerated value in the CIM Device Model. Mapping is not 1-to-1 but similar information is available.)
	Resource Type	Indicated by the SystemResource subclass that is instantiated and associated with the LogicalDevice.
	Resource Number	IRQ.IRQNumber, DMA.DMAChannel
	Resource Info ID	IRQ, DMA and MemoryMappedIO uniquely identified by their key properties.
	Start Address	MemoryMappedIO.StartingAddress or regarding memory - there is an AssociatedMemory relationship between a Device and Memory. How this Memory is “Realized” in physical hardware or “BasedOn” lower level Storage Extents is indicated by associations.
	End Address	MemoryMappedIO.EndingAddress and see the discussion for “Start Address” above for memory-related information.
	Resource Size	MemoryMappedIO.EndingAddress - StartingAddress, BasedOn.EndingAddress - StartingAddress
*	Base Alignment	
	Shareable	IRQ.Shareable
	Shared	Could be determined by examining the DMA or IRQ.Availability properties and traversing the AllocatedIRQ, AllocatedDMA, … associations from the Resources to the using LogicalDevices.

System Slots|004

	Slot Index	Slot.Tag (inherited from PhysicalElement), Slot.SlotID

Slot Type	Slot.ConnectorType (inherited from PhysicalConnector), Slot.LengthAllowed
Slot Width	Slot.MaxDataWidth
Current Usage	Instance of CardInSlot associated with the Slot.SlotID (or Slot.Tag) of interest, or an instance of CardOnCard specifying the appropriate SlotID.
Slot Description	Info available via the associations specified for "Current Usage".
Slot Category	Slot.ConnectorType (inherited from PhysicalConnector). And, a PhysicalComponentcan be directly associated with a HostingBoard (Card) via the PackagedComponent relationship (not the CardOnCard association). PackagedComponent indicates that the hardware is integrated directly by the Card/HostingBoard
Virtual Slot	A PhysicalComponent can be directly associated with a HostingBoard/Card via the PackagedComponent relationship (not the CardOnCard association). PackagedComponent indicates that the hardware is integrated by the Card / Physical Package.
Resource User ID	Indicated by the "Realizes" association from a PhysicalElement (ie, a Card) to the LogicalDevices that are instantiated because that PhysicalElement is in a System.
Vcc Mixed Voltage Support	Slot.VccMixedVoltageSupport
Vpp Mixed Voltage Support	Slot.VppMixedVoltageSupport
Slot Thermal Rating	Slot.ThermalRating

Temperature Probe|001

Temperature Probe Table Index	TemperatureSensor uniquely identified by its key properties.
Temperature Probe Location	A TemperatureSensor is "Realized" in a PhysicalElement. This PhysicalElement can have an associated Location.
Temperature Probe Description	Inherited from ManagedSystemElement to TemperatureSensor, the Description property.
Temperature Status	Can be determined by examining the properties of TemperatureSensor.
Temp Probe Temp Reading	Inherited from NumericSensor, the CurrentReading property.
Monitored Temp Nominal Reading	Inherited from NumericSensor, the NominalReading property.
Monitored Temp Normal Maximum	Inherited from NumericSensor, the NormalMax property.
Monitored Temp Normal Minimum	Inherited from NumericSensor, the NormalMin property.
Temperature Probe Maximum	Inherited from NumericSensor, the MaxReadable property.
Temperature Probe Minimum	Inherited from NumericSensor, the MinReadable property.
...Lower Threshold-Non-Critical	Inherited from NumericSensor, the LowerThresholdNonCritical property.
…Upper Threshold-Non-Critical	Inherited from NumericSensor, the UpperThresholdNonCritical property.
…Lower Threshold-Critical	Inherited from NumericSensor, the LowerThresholdCritical property.

	…Upper Threshold-Critical	Inherited from NumericSensor, the UpperThresholdCritical property.
	…Lower Threshold-Non-recoverable	Inherited from NumericSensor, the LowerThresholdFatal property.
	…Upper Threshold-Non-recoverable	Inherited from NumericSensor, the UpperThresholdFatal property.
	Temperature Probe Resolution	Inherited from NumericSensor, the Resolution property.
	Temperature Probe Tolerance	Inherited from NumericSensor, the Tolerance property.
	Temperature Probe Accuracy	Inherited from NumericSensor, the Accuracy property.
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Probe's physical “realization”. Alternately, the PhysicalElement(s) that realize the Probe could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association.
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.

UPS Battery|001

	Battery Status	UninterruptablePowerSupply.RemainingCapacityStatus
	Seconds on Battery	UninterruptablePowerSupply.TimeOnBackup
	Estimated Minutes Remaining	UninterruptablePowerSupply.EstimatedRunTime
	Estimated Charge Remaining	UninterruptablePowerSupply.EstimatedChargeRemaining
*	Battery Voltage	
*	Battery Current	
	Temperature Probe Index	Indicated by instantiation of a TemperatureSensor object and an AssociatedSensor relationship (between the TemperatureSensor and the UPS Device).
	FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the UPS's physical “realization”. Alternately, the PhysicalElement(s) that realize the UPS could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association.
	Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.

Video BIOS|001

	Video BIOS Index	VideoBIOSElement uniquely identified by its key properties.
	Video BIOS Manufacturer	Inherited from SoftwareElement to VideoBIOSElement, the Manufacturer property.
	Video BIOS Version	Inherited from SoftwareElement to VideoBIOSElement, the Version property.
*	Video BIOS Release Date	
	Video BIOS Shadowing State	VideoBIOSElement.IsShadowed

Video|002

Video Index	VideoController uniquely identified by its key properties.
Video Type	PCVideoController.VideoArchitecture
Current Video Mode	PCVideoController.VideoMode
Minimum Refresh Rate	VideoController.MinRefreshRate
Maximum Refresh Rate	VideoController.MaxRefreshRate
Video Memory Type	VideoController.VideoMemoryType
Video RAM Memory Size	VideoController.MaxMemorySupported
Scan Mode	VideoController.CurrentScanMode
Video Physical Location	Must follow the Realizes relationship from the LogicalDevice to a Physical Element. This Element can be a Card or contained by a Card (such as a HostingBoard or an add-in Card). PhysicalElements can have Location objects associated with them (via the PhysicalElementLocation relationship).
Current Vertical Resolution	VideoController.CurrentVerticalResolution
Current Horizontal Resolution	VideoController.CurrentHorizontalResolution
Current Number of Bits per Pixel	VideoController.CurrentBitsPerPixel
Current Number of Rows	VideoController.CurrentNumberOfRows
Current Number of Columns	VideoController.CurrentNumberOfColumns
Current Refresh Rate	VideoController.CurrentRefreshRate
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Video's physical “realization”. Alternately, the PhysicalElement(s) that realize the Video could be associated with other Elements that are to be replaced as a“set”, using the ReplacementSet object and the ParticipatesInSet association.
Operational Group Idx	Operational state data is inherited as properties of the LogicalDevice object.

Voltage Probe|001

Voltage Probe Index	VoltageSensor uniquely identified by its key properties.
Voltage Probe Location	A VoltageSensor is “Realized” in a PhysicalElement. This PhysicalElement can have an associated Location.
Voltage Probe Description	Inherited from ManagedSystemElement to VoltageSensor, the Description property.
Voltage Status	Can be determined by examining the properties of VoltageSensor.
Voltage Probe Voltage Level	Inherited from NumericSensor, the CurrentReading property.
Monitored Voltage Nominal Reading	Inherited from NumericSensor, the NominalReading property.
Monitored Voltage Normal Maximum	Inherited from NumericSensor, the NormalMax property.
Monitored Voltage Normal Minimum	Inherited from NumericSensor, the NormalMin property.
Voltage Probe Maximum	Inherited from NumericSensor, the MaxReadable property.
Voltage Probe Minimum	Inherited from NumericSensor, the MinReadable property.

...Lower Threshold-Non-Critical	Inherited from NumericSensor, the LowerThresholdNonCritical property.
…Upper Threshold-Non-Critical	Inherited from NumericSensor, the UpperThresholdNonCritical property.
…Lower Threshold-Critical	Inherited from NumericSensor, the LowerThresholdCritical property.
…Upper Threshold-Critical	Inherited from NumericSensor, the UpperThresholdCritical property.
…Lower Threshold-Non-recoverable	Inherited from NumericSensor, the LowerThresholdFatal property.
…Upper Threshold-Non-recoverable	Inherited from NumericSensor, the UpperThresholdFatal property.
Voltage Probe Resolution	Inherited from NumericSensor, the Resolution property.
Voltage Probe Tolerance	Inherited from NumericSensor, the Tolerance property.
Voltage Probe Accuracy	Inherited from NumericSensor, the Accuracy property.
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Probe's physical “realization”. Alternately, the PhysicalElement(s) that realize the Probe could be associated with other Elements that are to be replaced as a “set”, using the ReplacementSet object and the ParticipatesInSet association.
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.