System Management Architecture for Server Hardware White Paper

Version 2.0.0

Status: Informational

Publication Date: 2007-08-25

DSP2001

System Management Architecture for Server Hardware Whitepaper

Version 2.0.0

Publication Date: 2007-08-25

DSP2001

Status: Informational

Abstract

The Systems Management Architecture for Server Hardware (SMASH) is an initiative that represents a suite of specifications that standardize the manageability interfaces for server hardware. The suite of specifications describes an architectural framework, interfaces in the form of protocols, an addressing scheme, and profiles for server platforms.

This document is an architectural white paper that describes the concepts used in SMASH.

Acknowledgments

The following persons were instrumental in the development of this specification: Bob Blair, AMD; Greg Dake, IBM; Jon Hass, Dell; Jeff Hilland, HP (editor); Steffen Hulegaard, OSA Technologies; Arvind Kumar, Intel; Jeff Lynch, IBM; Aaron Merkin, IBM (editor); Christina Shaw, HP; Enoch Suen, Dell; Michael Tehranian, Sun; Perry Vincent, Intel, John Leung, Intel; Khachatur Papanyan, Dell; Reddy Dasari, Dell.

Table of Contents

List of Figures

1 Introduction

This document is an introduction to the architectural framework required for managing server hardware in today's data centers. It describes the basic principles required for understanding and implementing the Systems Management Command Line Protocol (SM CLP) and DMTF Web Services for Management (WS-Management) as applied in this environment. The architectural framework is composed of technologies defined in multiple standard specifications, including the following documents:

- *WS-Man Specification* [1]
- *SMASH White Paper* (this document)
- *Server Management Command Line Protocol Specification* [4]
- *Server Management Managed Element Addressing Specification* [2]
- *SMASH Implementation Requirements* [3]
- *Server Management CLP to CIM Mapping Specification* [5]
- a variety of profiles (see section 5), which are applicable to this environment

The focus of SMASH is to enable the management of the server resources in a standard manner across any Manageability Access Point implementation, regardless of operating system state.

1.1 Target Audience

The intended target audience for this document is readers interested in understanding the DMTF's Server Management Architecture, specifically the use of the SM CLP and WS-Management as applied to the management of servers.

1.2 Related Documents

[1] *Common Information Model (CIM) Schema*, V2.14, December, 2006, www.dmtf.org/spec/cim.html.

[2] DSP0215, *SM Managed Element Addressing Specification*, V1.0.0, 2005, DMTF SMASH, www.dmtf.org/standards/smash.

[3] DSP0217, *SMASH Implementation Requirements*, V1.0.0, 2006, DMTF SMASH, www.dmtf.org/standards/smash.

[4] DSP0214, *Server Management Command Line Protocol Specification*, V1.0.0, 2005, DMTF SMASH, www.dmtf.org/standards/smash.

[5] DSP0216, *SM CLP to CIM Mapping Specification*, V1.0.0, 2006, DMTF SMASH, www.dmtf.org/standards/smash.

[6] *Posix Utility Conventions*, The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition, www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap12.html.

[7] DSP0226, *Web Services for Management (WS-Management)*, V1.0, 2006-03-14.

[8] DSP0227, *WS-Management CIM Binding Specification Preliminary*, V1.0.0b, 2006-08-09.

[9] DSP0230, *WS-CIM Mapping Specification Preliminary*, V1.0.0c, 2006-08-09.

[10] RFC2616, *Hypertext Transfer Protocol – HTTP 1.1*, IETF, June 1999.

[11] RFC2818, *HTTP over TLS 1.0*, IETF, May 2000.

[12] RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, IETF, June 1999.

[13] RFC2246, *The TLS Protocol*, Version 1.0, IETF, January 1999.

[14] RFC1157, *A Simple Network Management Protocol (SNMP)*, IETF, May 1990.

[15] DSP1054, *Indications Profile*, V1.0.

[16] DSP1033, *DMTF Profile Registration Profile*, V1.0.

[17] RFC4301, *Security Architecture for the Internet Protocol*, IETF, December 2005.

[18] RFC4303, *IP Encapsulating Security Payload (ESP)*, IETF, December 2005.

[19] RFC4305, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*, IETF, December 2005.

[20] CIM Schema, Version 2.15.0.

[21] Web Services Architecture, W3C Working Group, Note 11, February 2004.

[22] RFC2609, *Service Location Protocol*, Version 2, IETF, June 1999.

1.3 Terminology

Term	Definition
Administrator	A person managing a system through interaction with management clients, transport clients, and other policies and procedures
Autonomous Profile	A profile that defines an autonomous and self-contained management domain. This includes profiles that are standalone or have relationships to other profiles.
CIM Profile	A specification that defines the CIM model and associated behavior for a management domain. The CIM model includes the CIM classes, associations, indications, methods, and properties. The management domain is a set of related management tasks. A profile is uniquely identified by the name, organization name, and version.
Client	Any system that acts in the role of a client to a Manageability Access Point
Command Line Protocol (CLP)	The command line protocol used for managing systems, which is defined by the Server Management Architecture for Server Hardware
Command Processor Engine	The logical entity within a Manageability Access Point that is responsible for parsing incoming commands and returning responses
Common Information Model (CIM)	The DMTF's approach to the management of systems and networks that applies the basic structuring and conceptualization techniques of the object-oriented paradigm. The approach uses a uniform modeling formalism that — together with the basic repertoire of object-oriented constructs — supports the cooperative development of an object-oriented schema across multiple organizations.

Term	Definition
Component Profile	A profile that describes a subset of a management domain. A component profile includes CIM elements that are scoped within an autonomous profile (or, in rare cases, another component profile). Multiple autonomous profiles may reference the same component profile.
Encapsulating Security Payload	An IPSec extension header that provides origin authenticity, integrity, and confidentiality protection for a packet
Extensible Markup Language (XML)	A simple, very flexible text format language derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.
Hypertext Transfer Protocol (HTTP)	An application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol that can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes, and headers.
HTTP over TLS	The Hypertext Transfer Protocol (HTTP) encapsulated in the Transport Layer Security Protocol
In-Band	Management that operates with the support of hardware components that are critical to and used by the operating system
IP Security	A suite of protocols for securing Internet Protocol (IP) communications
In-Service	Management that operates with the support of software components that run concurrently and are dependent on the operating system
Manageability Access Point (MAP)	A collection of services in a system that provides management in accordance to specifications published under the DMTF Server Management Architecture for Server Hardware initiative
Managed Element	The finest granularity of addressing that can be the target of commands or messages, or a collection thereof
Managed Element Access Method	The method by which a Managed Element performs a unit of work
Managed System	A collection of Managed Elements that comprise a Computer System for which a MAP has management responsibilities
Out-of-Band	Management that operates with hardware resources and components that are independent of the operating system’s control
Out-of-Service	Management that operates with the support of software components that require the operating environment to be put out-of-service and the system be placed into an alternate management environment. In this state, the operating system is not available.
Target Address Scheme Resolution Service	The entity responsible for discovering, enumerating, and determining the addresses of Managed Elements within the MAP
Transmission Control Protocol (TCP)	A connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols that support multi-network applications

Term	Definition
Transport	The layers of the communication stack responsible for reliable transportation of commands and messages from the Client to the MAP
Transport Layer Security (TLS)	A protocol that provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.
User	The set of Administrators and Management Clients that interact with the Transport Client to manage a Managed System through a Manageability Access Point
Web Services	A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically, WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.
WS-Addressing	Transport-neutral mechanisms used to address Web services and messages. Specifically, WS-Addressing defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. It enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.
WS-CIM Mapping	A specification that provides the normative rules and recommendations that describe the structure of the XML Schema, WSDL fragments, metadata fragments corresponding to the elements of CIM models, and the representation of CIM instances as XML instance documents
WS-Enumeration	A general SOAP-based protocol for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models
WS-Eventing	A protocol that allows Web services to subscribe to or accept subscriptions for event notification messages
WS-Management	A general SOAP-based protocol for managing systems such as PCs, servers, devices, Web services and other applications, and other manageable entities
WS-Management CIM Binding	A specification that describes how transformed CIM resources, as specified by the WS-CIM specification, are bound to WS-Management operations and WSDL definitions
WS-Transfer	A general SOAP-based protocol for accessing XML representations of Web service-based resources

1.4 Acronyms and Abbreviations

Term	Definition
CIM	Common Information Model
CLP	Command Line Protocol
DMTF	Distributed Management Task Force
ESP	Encapsulating Security Payload
FIFO	First in, First out
FRU	Field Replaceable Unit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over TLS
IP	Internet Protocol
IPSec	IP Security
KVM	Keyboard, video, mouse
LED	Light-emitting diode
MAP	Manageability Access Point
ME	Managed Element
NIC	Network Interface Card
PCI	Peripheral component interconnect
SSHv2	Secure Shell Version 2
SLP	Service Location Protocol
SMASH	Systems Management Architecture for Server Hardware
SM CLP	Server Management Command Line Protocol
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TCP/IP	*See* TCP *and* IP
TLS	Transport Layer Security
UFiP	User Friendly Instance Path
UFcT	User Friendly Class Tag
UFiT	User Friendly Instance Tag
URI	Universal Resource Identifier
WBEM	Web-Based Enterprise Management
WSDL	Web Service Definition Language
XML	Extensible Markup Language

2 Overview

Enterprise server management is comprised of a rich set of tools and applications that administrators can use to manage the data center. In many cases, these tools are specialized and adapted to each individual environment, installation, and product in the data center.

Currently, the richness of the CIM Schema provides a feature-rich systems management environment. In its current form, the CIM Schema also places an additional burden on vendors who attempt to implement the CIM Schema and Web-Based Enterprise Management (WBEM) Protocols to support server hardware management in the out-of-band and out-of-service scenarios. This burden has resulted in lack of interoperability in the server hardware management arena, particularly in the out-of-band and out-of-service cases. In addition, the resulting out-of-band and out-of-service management solutions are different from the operating systems' server management methods and server representations.

The Systems Management Architecture for Server Hardware initiative supports a suite of specifications that include architectural semantics, industry-standard protocols, and profiles to unify the management of the data center. By leveraging industry-standard protocols, interoperability is guaranteed over the network and the syntax and semantics of those protocols are guaranteed to be interoperable by compliant products that adhere to those standards. . Through the creation of industry-standard profiles, SMASH leverages the richness of the CIM Schema in a consistent manner so that systems offered by different vendors can be represented in similar ways.

Extra emphasis has been placed on the development of SMASH to enable lightweight implementations that are architecturally consistent. The goal of this effort is to enable a full spectrum of server implementations without sacrificing the richness of the CIM heritage. This spectrum of server implementations includes software-only solutions and small-footprint firmware solutions. Emphasis has been placed on ensuring that these implementations will be interoperable, regardless of implementation, CPU architecture, chipset solutions, vendor, or operating environment.

2.1 Principal Goals

One goal of SMASH is to enable the same interfaces regardless of server state. To this end, a Service Model has been included in 2.2.3 to illustrate that, regardless of Service Access Point or Operating System Service state, the same protocols should be able to be used for Systems Management.

Another goal of SMASH is to enable the same tools, syntax, semantics, and interfaces to work across a full range of server products – stand alone systems, rack-mounted servers, blades, Telco servers, and partitionable as well as virtual and redundant servers. Therefore, we have considered these products in our initial architecture and will include support for them in the ongoing profile development effort.

2.2 Service Model

Fundamental to the SMASH is the underlying goal to unify the experience achieved through out-of-band mechanisms with those available through the operating system. To achieve this goal, the SMASH contains a model to describe these terms (In-Band, Out-of-Band, In-Service, Out-of-Service) and to relate them to server management today.

2.2.1 In-Band versus Out-of-Band

A key concept in understanding the Service Model is an understanding of the terms In-Band and Out-of-Band and how they are used within the context of Server Management.

In-Band Management operates with the support of hardware components that are critical to and used by the operating system. An example would be a general purpose NIC available through the operating system.

Out-of-Band Management operates with hardware resources and components that are independent of the operating system. These resources are dedicated to systems management and allow management of system hardware components independent of their state. Typically, they are also available when the operating system is available and can interact with the operating system. An example would be a service processor or baseboard management controller.

2.2.2 In-Service versus Out-of-Service

Dependency on the operating system service state is described by the terms “In-Service” and “Out-of-Service”.

In-Service management operates with the support of software components that run concurrently and are dependent on the operating system. This is often provided through a service or process within the operating system.

Out-of-Service management operates with the support of software components that require the operating environment to be put out-of-service and the system to be placed into an alternate management environment. In this state, the operating system is not available.

2.2.3 Combined Service Model

By combining the operating system service dependency with the management access method (“In-Band”/“Out-of-Band”), we can achieve the SMASH Service Model matrix shown in Figure 1. This service model is useful in understanding what is meant by unifying the In-Service/Out-of-Service and In-Band/Out-of-Band management experience. This illustration should help vendors of manageability components, software, and solutions to understand the goal and deliverables encompassed by the SMASH. Included in the Service Matrix are examples of solutions for that part of the matrix.

In Figure 1 the horizontal axis represents the OS-Dependency and refers to the state of the normal operating system at the managed endpoint. The vertical axis represents the physical location of the Manageability Access Point. Note that Service Processor is terminologically equivalent to a firmware- or software-based management controller or service.

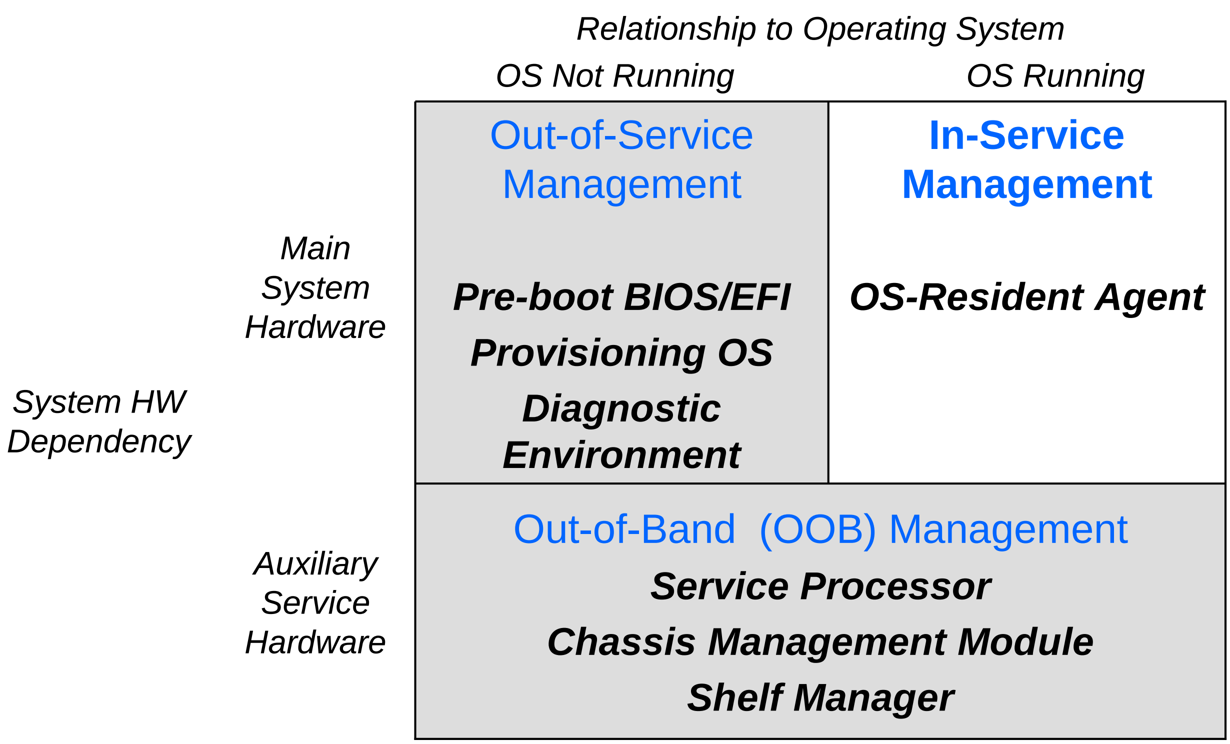

Figure 1 – Service Model

2.3 Management Protocol Selection

Two common paradigms for performing system management are the use of command line tools (either through manual invocation or driven by scripts) and the deployment of system management software applications. SMASH addresses both of these management paradigms through the inclusion of a command line protocol and a programmatic access protocol.

A command line protocol defines human oriented text messages exchanged over a network. A command line protocol may be layered on top of a transport client to provide a command line interface for management. A programmatic access protocol is one that is optimized for use by software applications to communicate with one another.

SMASH specifies the Server Management Command Line Protocol (SM CLP) and Web Services for Management Protocol (WS-Man) as the protocols a MAP may support for management. The SM CLP is a command line protocol that enables basic human user and script driven management. WS-Man is a programmatic access protocol that enables system management applications.

A MAP may support either or both of the protocols. While not required, it is expected that a MAP supporting both management protocols will expose the same set of Managed Elements through both protocols.

Irrespective of the protocols supported by a MAP, the basic architecture and the profiles that may be selected remain the same. Sections 6 and 7 detail the architectural elements that support the SM CLP and WS-Man protocol directly.

3 Architecture Definition

To provide server management standardization, it is necessary to develop an abstract model that describes server management regardless of the actual implementation. This abstract model is necessary to provide a common vocabulary and a common base of understanding. It is also used to illustrate the access points where interoperability is guaranteed as well as to show semantically visible components and interfaces.

The goal of the architecture is also to describe server management in abstract terms regardless of server type, topology, or framework. This means that the architecture must meet the following criteria:

- be implementation-agnostic
- span the spectrum from small stand-alone servers to large partitionable servers
- encompass topologies such as blades and racks, as well as unique segments such as industry standard servers, telecommunications, and mission critical high-end servers

3.1 Architectural Model

This section introduces the overall SMASH Model (see Figure 2). The terms used in this model are defined in the following sections. The dotted lines in this model indicate the protocols and transports that are externally visible. These are the communication interfaces between the Manageability Access Point (MAP) and the Client and represent data that flows across the network, for example. The solid lines indicate semantically visible interfaces. The packets, transports, and interfaces are not externally visible, but the fact that they are separate components with their own semantics is visible. The functional implications that are noticeable by the Client need to be accounted for to have a complete model.

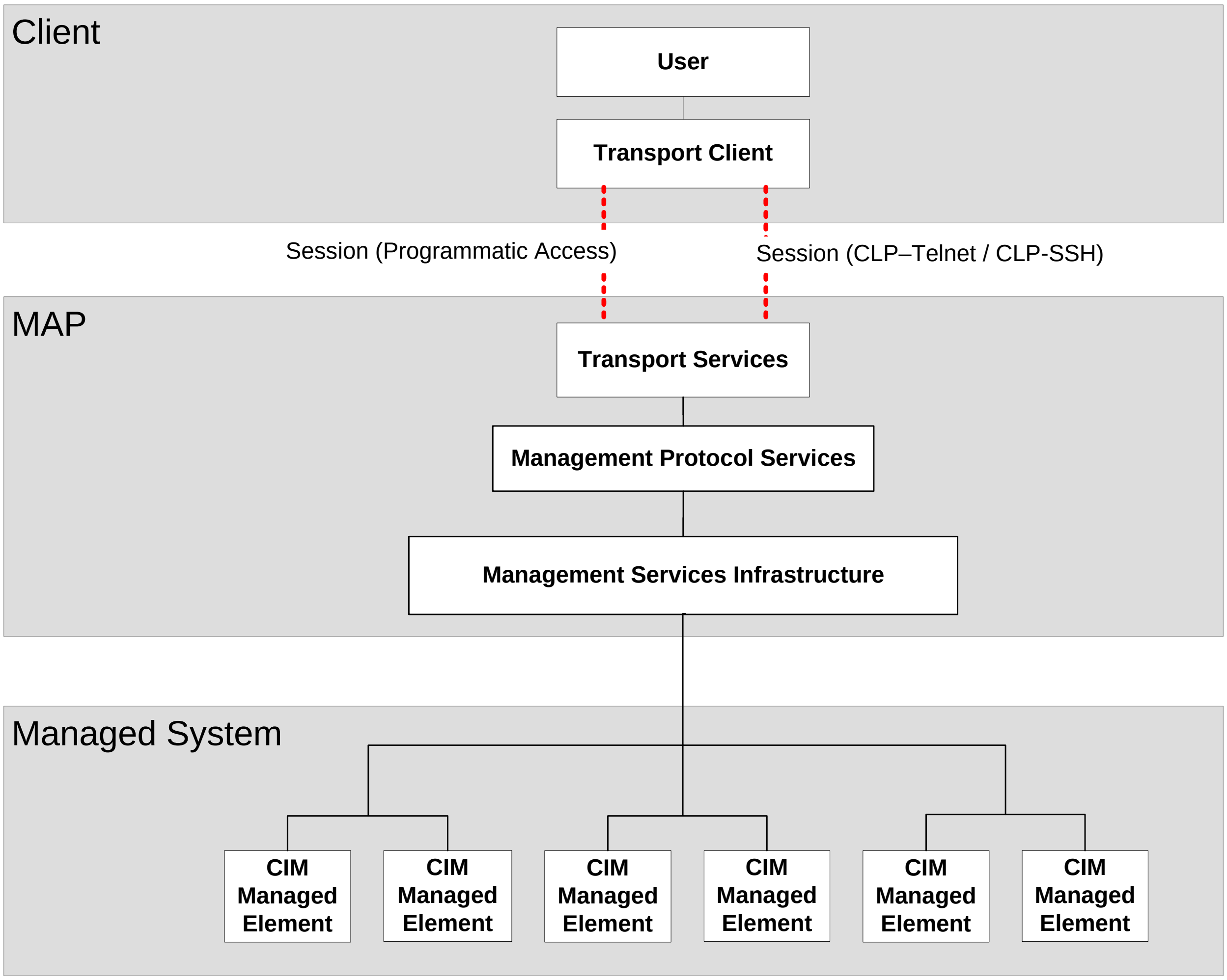

Figure 2 – SMASH Model

Figure 3 shows an example implementation with an emphasis on components within the MAP that are noticeable when implemented within a WBEM context. While the entities described are not required to exist as independent entities, their existence can be determined by the syntax and semantics of the interface between the MAP and the Client. This figure expands on the architecture model, exposing the detailed, identifiable portions of the Client and the MAP. This includes the Transports and a detailed User model to indicate support by the SM CLP of both a direct human Administrator and a Management Client. It also indicates that Authentication, Authorization, and Audit components exist within the map and, therefore, are expected to be accessible through the protocols. In addition, the Operation Invocation Engine and the Target Address Scheme Resolution Services indicate that both the operations within the MAP and the addressing and discovery within the MAP are distinct with their own operational semantics. Note that while only one Managed System is shown, managing multiple Managed Systems from one MAP is supported by the SMASH.

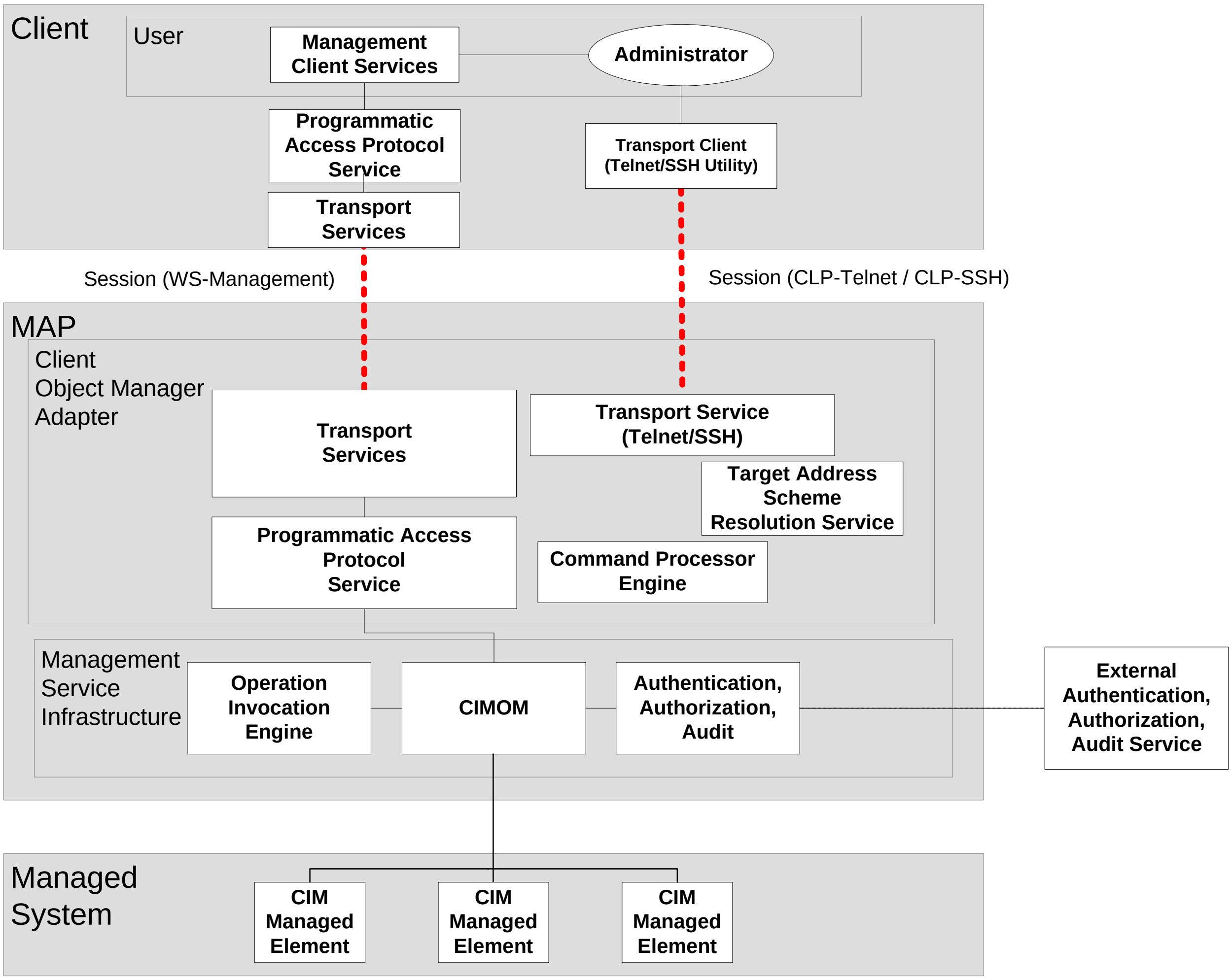

Figure 3 – Example MAP Implementation Architecture

3.2 Client

A Client is a logical component that manages a system through a Manageability Access Point (MAP). A Client may run on a management station or other system.

A Client is responsible for:

- providing an interface to the functionality provided by the MAP in a form consistent with the SMASH
- accessing a MAP using one of the SMASH defined management protocol specifications. This entails interacting with the MAP through the following process:
  – initiating a session with a MAP
  – transmitting protocol-specific messages to the MAP
  – receiving protocol-specific output messages from the MAP

3.2.1 User

A User in this model is either a user of the SM CLP or of the WS-Management interface. An SM CLP User in this model represents an instance of a Client that transmits and receives CLP compliant messages. The CLP is part of the SMASH. It is intended to either be a human or script interacting with a terminal service such as Telnet or SSHv2. For more information on the CLP, see [4]. A user of the WS-Management interface is a management client service that transmits and receives WS-Management messages.

3.2.1.1 Management Client

A Management Client represents a program of some type, such as a script or application that initiated management requests to the Transport Client and handles responses from the Transport Client. Interaction between the Management Client and the Transport Client is in the form of SM CLP or WS-Management messages. Interaction between the Administrator and the Management Client is outside the scope of this document.

3.2.1.2 Administrator

The Administrator represents the human interacting with either the Management Client or directly with the Transport Client. If using the SM CLP, the interaction between the Administrator and the Transport Client is in the form of SM CLP messages.

3.2.2 Transport Client

The Transport Client represents the endpoint of the transport and lower layer protocols with which the User interacts. It initiates and maintains the transport session with the Transport Service in the MAP. This includes the transport session establishment and authorization. If using the SM CLP, authentication is expected to take place either during or after Transport session establishment but before SM CLP Session establishment, as indicated later in this specification.

The SM CLP specification contains mappings for SSHv2 and Telnet, but other transports are possible. *SMASH Implementation Requirements* [3] contains mappings for HTTP and HTTPS. Other transports are not precluded but are outside the scope of SMASH.

3.3 Manageability Access Point

The Manageability Access Point (MAP) is a network-accessible service for managing a Managed System. A MAP can be instantiated by a Management Process, a Management Processor, a Service Processor, or a Service Process.

The MAP is responsible for the following tasks:

- managing the Session between the MAP and the Client. The MAP is considered the endpoint for the transport protocol.
- interpreting the incoming protocol-specific messages and seeing that a response is transmitted.
- returning protocol-specific output messages to the Client containing status and result data.

The MAP fulfils these responsibilities by utilizing components contained within the MAP. Note that the interface between the Managed Elements (ME) and the MAP is outside the scope of the SMASH. The interfaces within the MAP are outside the scope of the SMASH.

The MAP contains the following major components, which are discussed in the following sections:

- A Client Object Manager Adapter, which adapts the CLP Messages into CIM operations that the Management Service Infrastructure can act upon.
- The Management Service Infrastructure, which provides management access to the instrumentation of the Managed Systems.

3.3.1 Management Service Infrastructure

The Management Service Infrastructure is a logical entity that contains the MAP's core services, which implement a CIM Server. It is primarily comprised of the functions described in the following sections.

3.3.1.1 CIMOM

The CIM Object Manager is the component of the Management Service Infrastructure that handles the interaction between the Client Object Manager Adapter and the Providers. It supports services such as the Operation Invocation Engine and the Authentication, Authorization, and Audit components.

3.3.1.2 Operation Invocation Engine

The Operation Invocation Engine is responsible for understanding the management requests and tracking the initiation, interim status, and completion of operations resulting from those requests on Managed Elements. A major component of the Operation Invocation Engine is the Operation Queue. This is the queue of all the operations submitted to the MAP. Operations are discussed in more detail in section 4.

3.3.1.3 Authentication, Authorization, Audit

This entity is responsible for coordinating the authentication, authorization, and auditing within the MAP. This includes coordination of transport session establishment, local account information, and the access permission required for MAP operations. This entity is also responsible for coordinating audit information of the operations and tasks taking place within the MAP. Note that this service is internal to the MAP and does not include any external service components or coordination.

3.3.2 Client Object Manager Adapter

This component represents the collection of entities required to process the commands and responses and, as required by the messages, interact with the Management Service Infrastructure to accomplish the requests and produce responses. The Client Object Manager Adapter consists of the Transport Service and Command Processor Engine. If the protocol in use is WS-Management, it also includes the Management Protocol Service. If the protocol in use is the SM CLP, it includes the CLP Service and Target Address Scheme Resolution Service.

3.3.2.1 Transport Service

The Transport Service represents the transports and lower layer protocols on which the CLP resides. This includes the transport session establishment and authorization. Authentication is expected to take place either during or after Transport session establishment but before CLP Session establishment, as described in 6.2.

The Transport Service also represents the entity that encrypts and decrypts the data stream. This operation happens as part of the transport mechanism in this architecture. For instance, SSHv2 has encryption mechanisms.

3.3.2.2 CLP Service

The CLP Service represents the endpoint of the CLP within the MAP. Commands are received here and turned into internal operations within the MAP. This entity is responsible for receiving messages and transmitting responses that are compliant with the *SM CLP Specification* [4].

The interface between the CLP Service and the Management Service Infrastructure is implementation dependent; therefore, the interface itself is outside the scope of the SMASH.

3.3.2.3 Management Protocol Service

The Management Protocol Service represents the endpoint of the Management Protocol within the MAP. The Management Protocol for SMASH is WS-Management. WS-Management messages are received here and turned into internal operations within the MAP. This entity is responsible for receiving messages and transmitting responses that are compliant with the *WS Management Specification* [7].

The interface between the Management Protocol Service and the Management Service Infrastructure is implementation dependent; therefore, the interface itself is outside the scope of SMASH.

3.3.2.4 Command Processor Engine

The Command Processor Engine represents the entity that parses incoming commands and handles responses of the CLP. It is responsible for ensuring that the SM CLP messages are compliant with the grammar in the *SM CLP Specification* [4].

3.3.2.5 Target Address Scheme Resolution Service

The Target Address Scheme Resolution Service is responsible for discovering and enumerating the Managed Elements within the local domain, maintaining the addressing and naming structure of the local domain, and coordinating this information with the operation invocation engine. This Service is required to implement and adhere to the rules and grammar specified in the *Server Management Managed Element Addressing Specification* [2].

3.3.3 External Authentication, Authorization, Audit Service

The External Authentication, Authorization, Audit Service represents the entity that establishes and coordinates the authentication, authorization, and auditing information outside of the MAP. Examples of services that it may coordinate are keys, certificates, user accounts, passwords, and privileges. The instantiation of any global Authentication, Authorization, Audit Service is outside the current scope of the SMASH. In addition, the interface between the MAP and the Security

Service is outside the current scope of the SMASH. Note that this service is distinct from the Authentication, Authorization, Audit component (see 3.3.1.3) of the MAP because it is an external service and not contained within the MAP.

3.4 Managed System

A Managed System is a collection of Managed Elements that comprise a Computer System for which the MAP has management responsibilities. The Managed System may sometimes be referred to as a host, node, server, or platform. A Managed System could represent multiple types of systems, such as stand-alone, rack, blade, or virtual systems.

One or more Managed Element and/or Resources, or collections thereof, may be managed by a single MAP. Consequently, there may be multiple servers in a Managed System. More than one Managed System may be within the domain of any MAP.

Each Managed Element within the Managed System could contain subcomponents, sub-targets, or resources within that individual Managed Element.

3.4.1 Managed Element

Managed Elements are the targets, components, resources, collections, or logical entities within a Managed System that the operations will manipulate.

Specific interfaces for Managed Element access are outside the scope of the SMASH.

4 Operation Model

This section contains information relevant to operation handling within the MAP. It covers MAP responsibilities, operation handoff, queue depth issues, multi-session support issues, operation visibility, and resource handling.

For a complete understanding of operation handling, the reader should be familiar with the CIM_Job (Core Schema) and CIM_JobQueue classes. Though the terms *operation* and *job* are synonymous with respect to this specification, in the MAP operation model the term *operation* is often used.

4.1 MAP Responsibilities

The Manageability Access Point (MAP) has several responsibilities to the Client. Some of these may appear intuitive to some readers, but for purposes of clarity they are described here.

MAPs are responsible for managing the elements for which they claim responsibility. This does not imply that MAPs actually execute the method or modify the property included in the operation, but MAPs are the focal point of the interaction and are responsible for tracking the operation.

The MAP is responsible for ensuring that a command is syntactically correct. The MAP may pass the parsing of a command to further levels within the MAP or System, but it is the MAP that ensures that the implementation complies with the protocol.

The MAP is responsible for handling commands, messages, and operations. The MAP may delegate the actual operation, but it is responsible for handling commands and messages, turning them into jobs or operations, tracking operations, and manipulating the operations (including completing, canceling, removing, or logging).

The MAP is responsible for determining if the specified ME is in the scope of the MAP. Operations that target MEs that are not within the MAP's scope should result in the appropriate error syndrome.

The MAP is responsible for determining if access to the ME is allowed. This includes, but is not limited to, authorization determination (to ensure that the user account and access right combination will allow access to the ME) and determination that the ME is in a state where the operation can be initiated.

The MAP is also responsible for determining if the operation or property modification is properly formed and conveyed for a particular Managed Element and if the operation or property modification is a valid request. It is the MAP's responsibility to ensure that any such request takes place as indicated.

The MAP is responsible for maintaining any session context required. Because the MAP contains the connection with the transport, the MAP maintains the following information:

- any session-related information, such as current default target
- option settings, such as language, locale, or output format

For protocols that do not maintain session state or do not allow connections to persist, the MAP does not need to maintain session information.

If the SM CLP is used, the MAP is responsible for maintaining the local UFiT address space, including any aliases or OEM extensions. The MAP is responsible for ensuring the creation of the address space of Managed Element instances and mediating commands and messages into operations on those elements.

4.2 Operation Handoff

Operations within the MAP are not directly visible to the Client. However, the fact that operations exist, are initiated, can be cancelled, can complete, and can be deleted is visible to the Client. In addition, the Client can retrieve the status of an operation.

Operations can be created only by using commands or messages. The MAP exposes one and only one identifiable, traceable operation for any single, valid command. If an implementation spawns multiple activities to process a single command or message, then all of the activities are related to the single job identifier created when the operation was initiated; it is the responsibility of the MAP to track the multiple activities and relate them to the single operation.

All operations have identifiers. The CIM_ConcreteJob class is used to represent operations, so the identifier is that of a CIM_ConcreteJob instance. The term Operation ID (OPID) or Job ID is used interchangeably to represent the identifier of that CIM_ConcreteJob instance. Note that OPIDs are returned when the operation is spawned, regardless of the duration of the operation. The status of the operation can be retrieved with a command or message using the OPID. The MAP must keep track of all active operations.

When an operation is complete, the settings for the operation determine if that instance represented by the OPID persists or is recycled immediately. The TimeBeforeRemoval property from the CIM_ConcreteJob class determines the amount of time that an operation persists in the operation queue.

All operations must be able to handle a cancellation request. Sometimes the response to the cancellation will be an error, such as in either of the following cases:

- an operation that cannot be undone,
- an operation that has already taken place or that cannot be stopped part of the way through, such as turning the power off or resetting a system

Any operation that takes longer than the typical command-response time is run asynchronously and an operation identifier is returned. The Client can then determine the status of the operation and whether or not the operation is complete. This process can be done through a query operation on the operation queue using the OPID. The operation queue can also be queried to find out the maximum operation queue depth or if the queue is full.

4.3 Operation Queue

The architecture contains an operation Service within the MAP that logically contains an operation queue. This is a FIFO queue that contains all of the operations to be processed within the MAP. All current sessions submit operations to this single queue. The operation queue is modeled using CIM_JobQueue. The properties of the operation queue are expected to vary depending on implementation.

Ordering is with respect to command initiation and is implied by the queue. Ordering of operation initiation is guaranteed within a session, but no such guarantee is made between sessions.

The MAP's operation queue depth varies from MAP to MAP. The minimum acceptable operation queue depth is equal to one operation. Some implementations may support multiple outstanding operations on a single session; others may not. Should the queue become full, the MAP is responsible for communicating this resource-constrained condition distinct from other error conditions through the use of error codes. For instance, an error that indicates that a resource is busy is distinct from one that indicates that the job queue is full.

Through the modeling of the operation queue within the MAP, the MAP must be able to indicate to the Client the maximum operation queue depth supported, as well as the number of current outstanding operations.

Detailed information for individual operations on the operation queue, such as what is available through CIM_ConcreteJob, can be obtained through the MAP by directing queries at individual operations.

4.4 Multi-Session Capability

An important aspect of MAP operations management is to be able to support simultaneous sessions through the MAP. Implementations are not required to support more than one session simultaneously. However, implementations are expected to exist that support many simultaneous sessions. Therefore, the SMASH supports multiple concurrent sessions.

The number of ports offered to transports from the Management Services Core for each protocol supported must be at least one per protocol supported. The MAP utilizes the error syndromes of the transport and subsequent layers when handling out-of-resource conditions (such as no more ports available), attempting to connect to the wrong port, or not supporting the requested transport.

Another aspect of the multi-session capability is that operations are visible regardless of the transport that initiated them. This capability implies that there is one global operations (job) queue per MAP, and the MAP is responsible for routing the results of operations to the appropriate session. But if the command or message spawns an operation, then any session should be able to discover the details about the operation in question by querying the operation using the OP ID. This is helpful for a number of reasons. For example, if an operation is spawned, the Client may disconnect and then query the status of that operation at a later time, provided the Client has retained or can discover the identifier for that operation.

4.5 Resource Handling

The SMASH contains mechanisms that enable resource handling.

In the SMASH, the manipulation of resources in the server is limited to treating the server as a collection of Managed Elements. This approach allows the MAP to create and modify configurations of the system, as well as establishing boot order.

The administration and configuration of complex systems, such as those with shared resources, often requires the locking of an ME to manage the ME or to ensure that the ME is assigned to one and only one system. Direct support of these mechanisms is not included in this version of the architecture. Because direct support is not required, the mechanism for handling resource locking is outside the scope of this specification.

5 Profiles

DMTF Management Profiles provide the information model definitions for manageability content and architecture models for mapping computer hardware in a way that is consistent between different implementations. These profiles combine to ensure that implementations supporting the management of similar components provide a consistent representation of the components. Individual implementations support the profiles that are appropriate for the hardware and software configurations they manage.

CLP implementations are dependent on the underlying modeling of system components. To achieve an interoperable CLP, the information models utilized are required to be consistent across implementations.

The SMASH identifies a subset of DMTF Management Profiles that are appropriate for its targeted management domain. The following is a list of DMTF Management Profiles that are included in the SMASH CLP architecture, including a brief description of the functionality provided by each. Because implementations select the DMTF Management Profiles that are appropriate for their environment, not all profiles are supported by all implementations.

- DSP1004, the *Base Server Profile* is a top-level profile that provides the ability to manage server systems.
- DSP1030, the *Battery Profile* provides the ability to manage batteries of a managed system.
- DSP1012, the *Boot Control Profile* provides the ability to manage boot configurations of a system.
- DSP1018, the *Service Processor Profile* provides the ability to manage service processors, chassis managers, and other dedicated management controllers.
- DSP1005, the *CLP Service Profile* provides the ability to manage an implementation of the SM CLP architecture.
- DSP1022, the *CPU Profile* provides inventory, status, and state information for processors of a managed system.
- DSP1019, the *Device Tray Profile* provides the ability to manage shared media trays in a modular system.
- DSP1037, the *DHCP Client Profile* provides the ability to manage the DHCP client configuration of a managed system.
- DSP1038, the *DNS Client Profile* provides the ability to manage the DNS client configuration of a managed system.
- DSP1014, the *Ethernet Port Profile* provides inventory, status, and state information for the Ethernet interfaces of a managed system.
- DSP1013, the *Fan Profile* provides inventory, status, and state information for fans of a managed system.
- DSP1054, the *Indications Profile* provides the ability for a client to subscribe to indications produced by a MAP.
- DSP1074, the *Indicator LED Profile* provides the ability to manage the LEDs of a managed system.

- DSP1036, the *IP Interface Profile* provides the ability to manage the configuration of IP interfaces of a managed system.
- DSP1076, the *KVM Redirection Profile* provides the ability to configure the KVM redirection infrastructure of a managed system.
- DSP1008, the *Modular System Profile* provides the ability to manage modular enclosures and contained components.
- DSP1029, the *OS Status Profile* provides the ability to determine basic status information about installed and running operating systems.
- DSP1020, the *Pass-Through Module Profile* provides inventory, status, and state information for pass-through modules of a managed system.
- DSP1075, the *PCI Device Profile* provides inventory, status, and state information about PCI devices in a managed system.
- DSP1011, the *Physical Asset Profile* provides the ability to report physical asset information including capacity and FRU information for components installed in a managed system.
- DSP1027, the *Power State Management Profile* provides the ability to query and manage the power state on a managed system.
- DSP1015, the *Power Supply Profile* provides inventory, status, and state information for power supplies of a managed system.
- DSP1010, the *Record Log Profile* provides the ability to retrieve error and event log information for managed systems.
- DSP1039, the *Role Based Authorization Profile* provides the ability to manage rights granted to security principals through role membership.
- DSP1009, the *Sensors Profile* provides the ability to query sensor status and state information for component and system sensors.
- DSP1021, the *Shared Device Management Profile* provides the ability to control access to shared devices in a modular system.
- DSP1034, the *Simple Identity Management Profile* provides support for basic account management, including account creation and deletion.
- DSP1007, the *SM CLP Admin Domain Profile* is used to model the administrative domain of an SM CLP implementation.
- DSP1006, the *SMASH Collections Profile* provides support for collecting settings, capabilities, and other Managed Elements to simplify management access through an SM CLP implementation.
- DSP1023, the *Software Inventory Profile* provides the ability to view the firmware, device drivers, BIOS, and other software installed on a system and its components. It also provides the ability to view the software available for installation on a system and its components.
- DSP1025, the *Software Update Profile* provides the ability to perform software installation, upgrades, and downgrades on a system and its components.

- DSP1017, the *SSH Service Profile* provides the ability to manage the configuration of an SSH service and client sessions.
- DSP1026, the *System Memory Profile* provides inventory, status, and state information for the main system memory of a managed system.
- DSP1016, the *Telnet Service Profile* provides the ability to manage the configuration of a Telnet service and client sessions.
- DSP1024, the *Text Console Redirection Profile* provides the ability to start and stop text console redirection over the interfaces of a managed system.
- DSP1040, the *Watchdog Profile* provides the ability to manage watchdog timers of a managed system.

6 SM CLP Protocol Support

This section provides an overview of the SM CLP within SMASH.

6.1 Target Addressing

The primary goal of the target addressing scheme is to provide an easy-to-use way to address CIM objects accurately.

The target address term of the CLP syntax in this architecture is extensible. Addressing for version 1.0.0 is fully described in the *Server Management Managed Element Addressing Specification* [2].

The addressing scheme provides a unique target for CLP commands. The scheme is finite for parsing target names and unique for unambiguous access to associated instance information needed to support association traversal rooted at the MAP AdminDomain instance.

6.1.1 Addressing Architecture

The addressing rules are applied to the CIM aggregation and association relationships to ensure that each fully qualified instance name is unique. This is accomplished by requiring that an instance name be unique within its immediate container. The specific containers that Managed Elements are allowed to be in is defined fully in the *Server Management Managed Element Addressing Specification* [2].

The addressing rules, specified in the *Server Management Managed Element Addressing Specification* [2], contain the detail necessary to fully understand the formulation of addresses and valid Target names for the CLP. This section contains a brief overview of the addressing architecture.

6.1.2 UFcTs and UFiTs

A User Friendly class Tag (UFcT) convention is defined to simplify long complex CIM class names without compromising object references, class properties, associations, or behavior. This provides a more user friendly experience for the Client (human end user). UFcTs are simple synonyms of specific CIM classes used in Server Management Profiles.

A User Friendly instance Tag (UFiT) is formed by taking a User Friendly class Tag and combining it with a non-negative integer suffix.

UFcTs are used to represent CIM classes. UFiTs are used to represent a specific Managed Element.

UFiTs are then combined in a manner similar to a file directory structure to form a User Friendly instance Path (UFiP). This structure is based on the collection of, associations between, and aggregations of Managed Elements.

6.1.3 Target Addressing in the CLP

The Server Management Command Line Protocol will accept UFiTs that are formed into a UFiP. The SM CLP also accepts other target address constructs, such as those used to select all instances of a class. MAPs support a number of standard, default UFiTs that are consistent with the SMASH addressing rules contained in the *SM Managed Element Addressing Specification* [2].

6.2 Transport Considerations

Implementations of SMASH may support Telnet or SSHv2 as the transport for the CLP. The detailed requirements for each transport protocol are detailed in the CLP specification [4]. Information on the exact specifications supported as well as any other information required to implement the CLP over these specific transports is contained in the SM CLP specification. The architectural model described in 3.1 shows how these transports are included in the architecture.

Some transports contain their own authentication mechanisms, such as key-exchange in SSHv2. Others rely on an intermediate authentication mechanism. If the transport supplies an authentication mechanism, it should equate to a user configured in the MAP, which will then be used for the session's authorization information. If another authentication mechanism is used, such as in the case of Telnet, the logon mechanism is expected to be user based, so the user name and password used to authenticate the Telnet session can be used to determine authorization of the commands of the CLP. For instance, key exchanges equate to user names and passwords. The user name and password used to authenticate the connection, or the user name and password associated with the key information, are the user name and password used to determine authorization of the commands of the CLP. Regardless, the CLP Service expects authentication to be performed before a session is established between the CLP and the Client. The CLP Session established is expected to pass a user account name as described in 8.1 to the MAP for use in authorizing commands.

For transports that do not contain an adequate encryption protocol, it is recommended that they be layered upon a protocol that supports strong encryption. It should be apparent to the reader that the vulnerability of the MAP is equivalent to the vulnerability of the transport protocols. Thus, to prevent intrusion the MAP should support secure transports. In the case of Telnet, any mapping of Telnet over a protocol such as TLS or SSL is outside the scope of this specification and SMASH. SSHv2 includes automatically negotiated encryption, so any layering is not required because encryption is inherent in the protocol.

7 Programmatic Access Protocol Support

SMASH uses a CIM-based data model for representing managed resources and services. The Management Services infrastructure and protocols are used to exchange the management information in a platform-independent and resource-neutral way. This is done by encapsulating CIM operations in a management protocol, which (in turn) is encapsulated in a transport protocol. This section describes the management protocol and transport protocol selected by SMASH.

7.1 WS-Management Protocol

This section provides an overview of the use of the WS-Management Protocol in SMASH.

7.1.1 Overview

SMASH supports the Web Services for Management Protocol, as defined in the *WS-Management Specification* [7], as the management protocol for transporting SMASH messages. WS-Management is a specification of a core set of Web Services to expose a common set of system management operations. The specification comprises the abilities to:

- Discover and navigate management resources.
- Manipulate management resources (create, destroy, rename, get, put).
- Enumerate the content of containers or collections (logs or tables).
- Subscribe/unsubscribe to events.
- Execute specific management methods.

The WS-Management protocol stack for SMASH is shown in Figure 4. The WS-Management stack is based on the Web Services. The network and physical layers are the two bottom layers in the stack.

The transport layers that carry SOAP messages are next in the stack. These layers include

- TCP, which provides reliable, stream-oriented data transport
- TLS, which provides various security attributes
- HTTP 1.1, which provides user authentication and request-response semantics

TCP and HTTP 1.1 are required by SMASH. TLS support is conditional on support for security profiles that require it.

At the next layer, SOAP/XML messaging is handled. The security profiles specified in *SMASH Implementation Requirements Specification* [3] define the security mechanisms required. Above the SOAP/XML layer is the data transfer layer, which is based on multiple Web Services specifications. These are WS-transfer, WS-Enumeration, and WS-Eventing for transferring the management information. The top three layers represent the WS-Management applications. SMASH profiles are mapped over the WS-Management protocol stack using the *WS Management CIM Binding Specification* [8] (which is defined in terms of WS-CIM [9]).

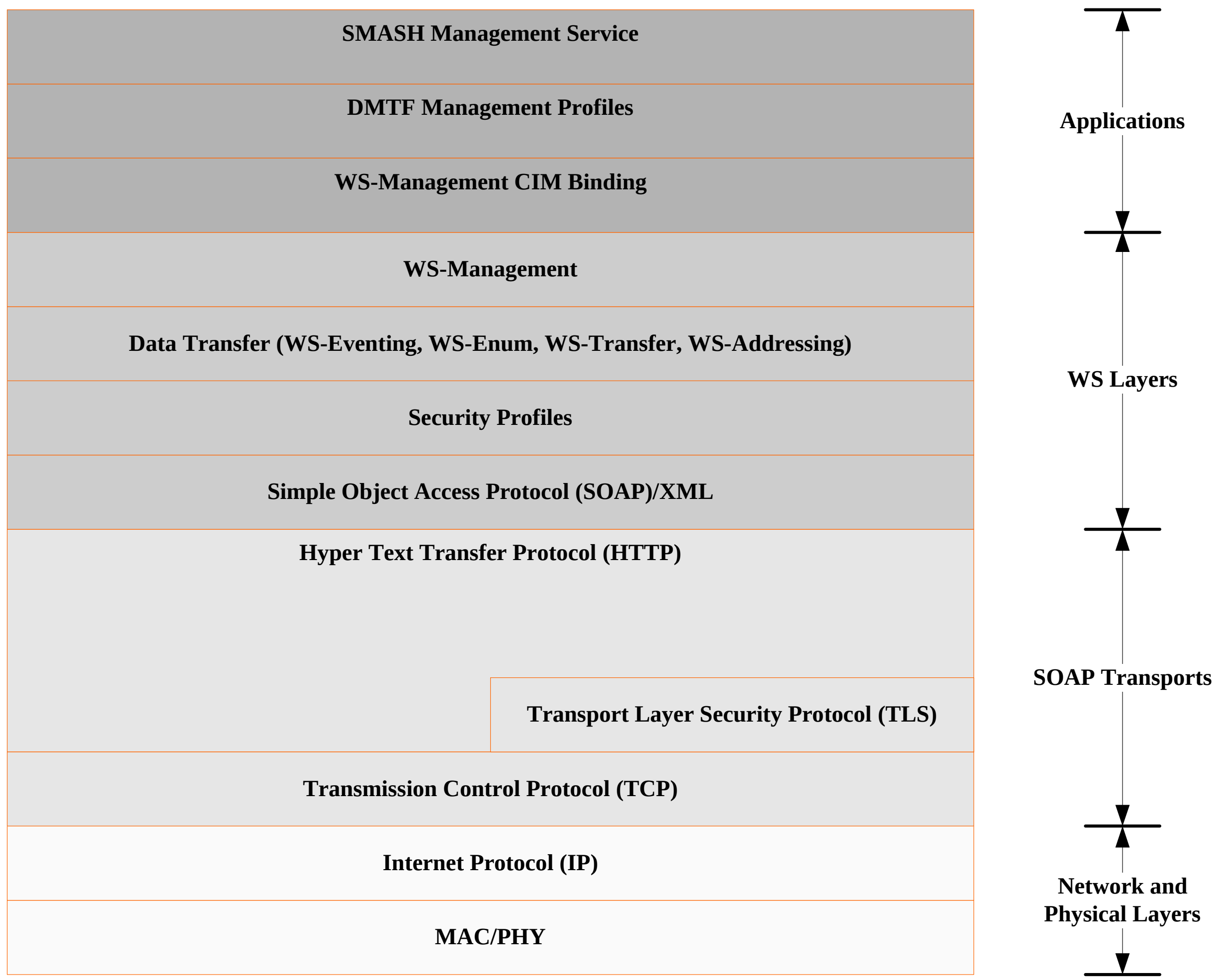

Figure 4 – SMASH Protocol Stack

WS-Management defines a default addressing model based on WS-Addressing. WS-Addressing defines a reference format using EndPointReference (EPR) that uses a ReferenceParameter field to identify specific elements (ResourceURI and SelectorSet). WS-Addressing is used to identify and access resources (CIM objects in SMASH).

The three data transfer models used by WS-management are briefly described below:

1. WS-Transfer: defines a mechanism for acquiring XML-based representations of entities. It defines the following resource operations using SOAP messages.
   a) *Get*: is used to fetch a one-time snapshot representation of a resource.
   b) *Put*: is used to update a resource by providing a replacement representation.
   c) *Create*: is used to create a resource and provide its initial representation.
   d) *Delete*: is used to delete a resource.
   e) In addition, WS-Management defines the rename operation and fragment-level transfer for fragment-level access of resources.
2. WS-Enumeration: is a SOAP-based protocol for enumeration. Using this protocol, the data source can provide a session abstraction called the enumeration context. The

consumer can then request XML element information over a span of one of more SOAP messages using the enumeration context. The enumeration context is represented as XML data. The following operations (defined as SOAP request/response messages) are supported using this model[1]:

a) *Enumerate:* is used to initiate an enumeration and receive an enumeration context.

b) *Pull:* is used to pull a sequence of elements of a resource.

c) *Release*: is used to release an enumeration context gracefully.

3. WS-Eventing: is a SOAP-based protocol for one Web service to register interest and receive messages about events from another Web service. The operations supported by WS-Eventing include *Subscribe, Renew, GetStatus, Unsubscribe,* and *SubscriptionEnd*. WS-management defines heartbeats as pseudo-events. WS-Management also defines a bookmark mechanism for keeping a pointer to a location in the logical event stream. The delivery modes defined for events are: Push, Push with Acknowledgement (PushWithAck), Batched, and Pull.

7.1.2 WS-Management – CIM Binding

The *WS-Management CIM Binding Specification* [8] defines the binding between the Web Services representation of CIM (defined in the WS-CIM Mapping Specification [9]) and WS-Management. This binding encompasses:

- WS-Addressing based addressing to identify and access CIM objects that are accessed over the protocol.
- Retrieving and updating instances of a class using WS-Transfer.
- Enumerating instances of classes using WS-Enumeration.
- Invoking an extrinsic method using action URIs and messages.
- Performing generic operations using WS-Management equivalent operations.

7.2 Transport Protocol

The WS-Management protocol is transport-independent, but it specifies HTTP 1.1 [10] and HTTPS [11] as the common transports for interoperability.

SMASH uses HTTP 1.1 as the SOAP transport for WS-Management. HTTP 1.1 is consistent with existing transports used by the Web servers and Web Services. HTTP 1.1 is widely supported, deployed, tested, and enhanced. HTTP provides two-way authentication in the form of basic and digest authentication (RFC2617) [12]. HTTP digest authentication exchanges are confidential, but HTTP does not provide general-purpose confidentiality. There is a well known SOAP binding for HTTP. Transport Layer Security (TLS) 1.0 (RFC2246) [13] can be used to add encryption, message integrity, message origin authentication, and anti-replay services to HTTP-based communications. HTTPS supports HTTP communications over TLS [13].

The preferred ports for SMASH are the IANA defined HTTP and HTTPS ports.

[1] The WS-Enumeration operations *Renew*, *GetStatus*, and *EnumerationEnd* are omitted here because their use is not recommended by the WS-Management specification.

7.3 Authentication Mechanisms

The three types of authentication mechanisms considered are as follows:

1. **Machine-level authentication**

   This authentication type is used to authenticate the machine that is accessing the service. The machine-level authentication uses machine-level credentials (such as keys and certificates) for the authentication. The machine-level authentication does not authenticate a particular user or a user session.

2. **User-level authentication**

   This authentication type is used to authenticate a particular user. It is typically based on the usernames/passwords, and it may involve a third party that provides an identity for the user. User-level authentication is performed on a per-operation basis. However, this authentication is typically visible to the user only on the first operation; the user's credentials are cached for use in subsequent operations.

3. **Third-party authentication**

   This authentication type is typically an out-of-band authentication mechanism in which a third party is used to verify the user credentials. The credentials used for the authentication are issued by the third party (such as a certificate authority). The user provides these credentials during the authentication process. The third party is involved in authenticating a user. (The third party verifies user credentials.) Typically, the third-party authentication is performed using a separate channel, and it does not involve the managed system. Typically, the authentication is asserted in the credential, and the MAP authenticates the credential.

SMASH requires user-level authentication support at minimum. The machine-level authentication is optional.

Note: If class B security compliance is needed, then the machine-level authentication is required for the defined security profiles. The third-party authentication is optional. A SMASH implementation can choose to support multiple levels of authentication.

7.4 Eventing

This section provides an overview of the SMASH model for delivery of events. This model encompasses a definition of alert indications, methods for subscribing to and delivering alert indications, and a standard alert indication message format.

SMASH targets the use of WBEM-based event notification mechanisms in conjunction with greater standardization of event message content. Traditionally, Simple Network Management Protocol (SNMP) [14] network messages have been used to communicate event related information from the Managed System to a listener console or application. With the advent of CIM-based management interfaces, more robust event delivery and more granular control of event message traffic is enabled. The *SMASH Implementation Requirements Specification* [3], in conjunction with WS-Management [7], WS-Management CIM Bindings [8], Profiles and related Message Registry specifications, defines a new level of Web Services-based event management and notification.

7.4.1 Overview

The CIM model contains alert indication class designs that represent events (7.4.2). The SMASH approach to event management combines the WS-Eventing event subscription model, specific requirements for generating alert event indications, and a standardization of event alert indication message content. Figure 5 provides an example of the sequence of activities that take place when instrumentation generates an indication filter, an application subscribes to the indication filter, and the instrumentation generates an indication based on an underlying event.

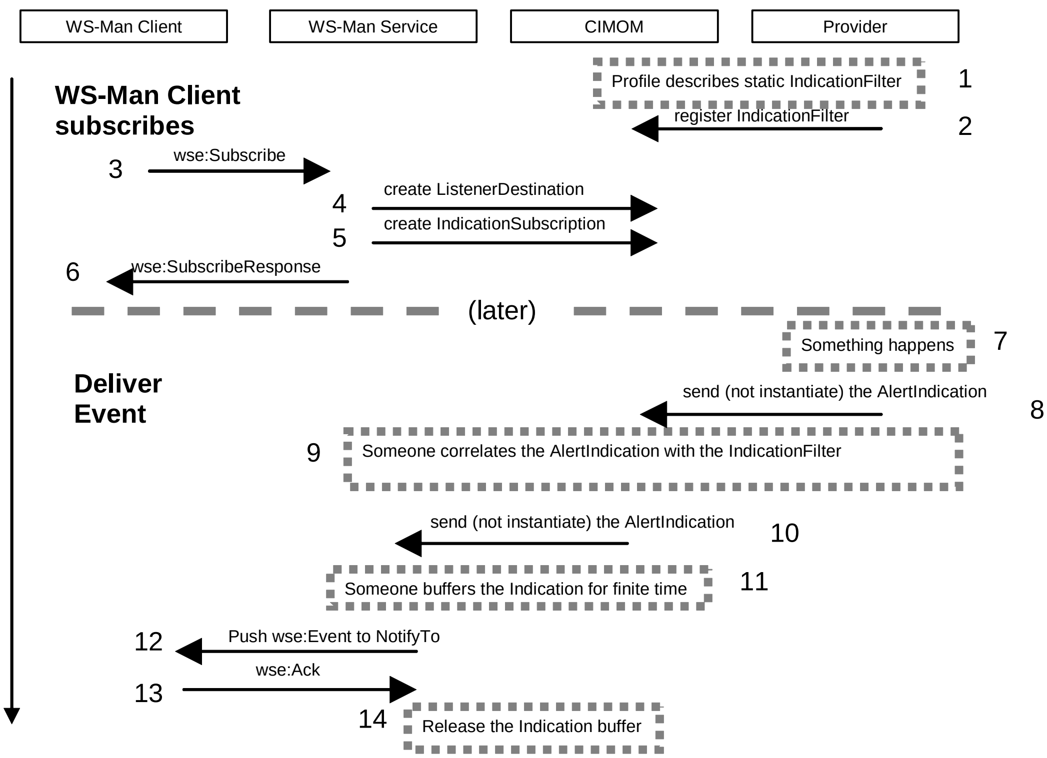

Figure 5 – Indication Activity Diagram

The first sequence of events in Figure 5 provides an example of how instrumentation indicates that it would make a filter available. In step 1, the provider has a static description of at least one IndicationFilter, for which support was probably created when the provider was developed. In step 2, the provider indicates to the CIMOM that it has an Indication Filter by registering the IndicationFilter with the CIMOM. Now the CIMOM adds this information into the repository.

When a WS-Management-based Client subscribes to an indication, it sends a WS-Management Subscribe message to the implementation (step 3). The WS-Management service, in turn, creates the ListenerDestination and IndicationSubscription instances in the CIMOM (steps 4 and 5) to represent the client and creates the appropriate associations. This information is then returned to the Client in the SubscribeResponse message (step 6).

When an event occurs (step 7), the instrumentation has the responsibility of communicating the event to applications that have subscribed to that particular information. The WS-Eventing approach to communicating event information involves generating an instance of the appropriate CIM Indication Class and sending the instance information, along with other information, as the payload of an event delivery message to subscribing listeners. Specifics of the CIM-to-event delivery message mapping are defined in the *WS-Management CIM Binding Specification* [8]. A synopsis of that process is as follows: when an event occurs (step 7), the provider sends the AlertIndication to the CIMOM (step 8). Then one of the implementation components correlates the AlertIndication from the provider with the IndicationFilter from the Client (step 9). Then the CIMOM sends the AlertIndication to the WS-Management Service (step 10). The service then pushes the Event (step 12) to the Client, which acknowledges the message (step 13), resulting in the Indication buffer being released (step 14). Note that the instance of the indication will be buffered for a finite amount of time by the MAP, implying that the Client should acknowledge the receipt of the message in an expedient fashion.

7.4.2 Alert Indications

The content of an Alert Indication consists of a Message ID/string-oriented class design. The content includes handles pointing to the alerting Managed Element and includes support for specifying recommended actions. The content includes a Message ID, which correlates to a Message Registry entry. The content may also include other identifying information in the form of MessageArgs, which will be indicated in the Message Registry as well. Note that the underlying event and its data may or may not be modeled in the CIM class hierarchy representing the managed system.

7.4.3 CIM Modeling of Events

The CIM event notification model is a subscription-based approach to configuring event indication delivery. The MAP represents the subscription, listener destination and event filters as defined in the *Indications Profile* [15]. Figure 6 represents the actions and resultant representation of an event indication subscription. For a detailed explanation of the classes, please refer to the *Indications Profile* [15].

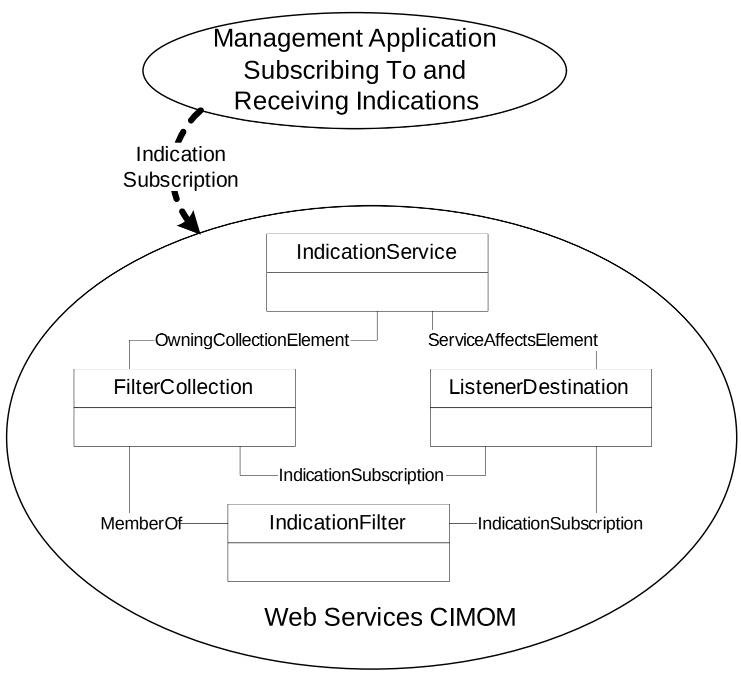

Figure 6 – Event Indication Subscription

7.4.4 Standardized Message Content

To foster greater interoperability between different implementations of management instrumentation and the applications that subscribe to and receive events, a set of standardized event message content has been defined. The event message content is specified in XML documents according to the DMTF Message Registry Schema. Message Registry entries consist of definitions for a message ID, message string, message arguments, perceived severity, and defining organization. Each Message in a registry represents a particular event type. SMASH 1.0 uses message registries for the Message IDs, perceived Severity and interpretations of MessageArgs for each MessageID.

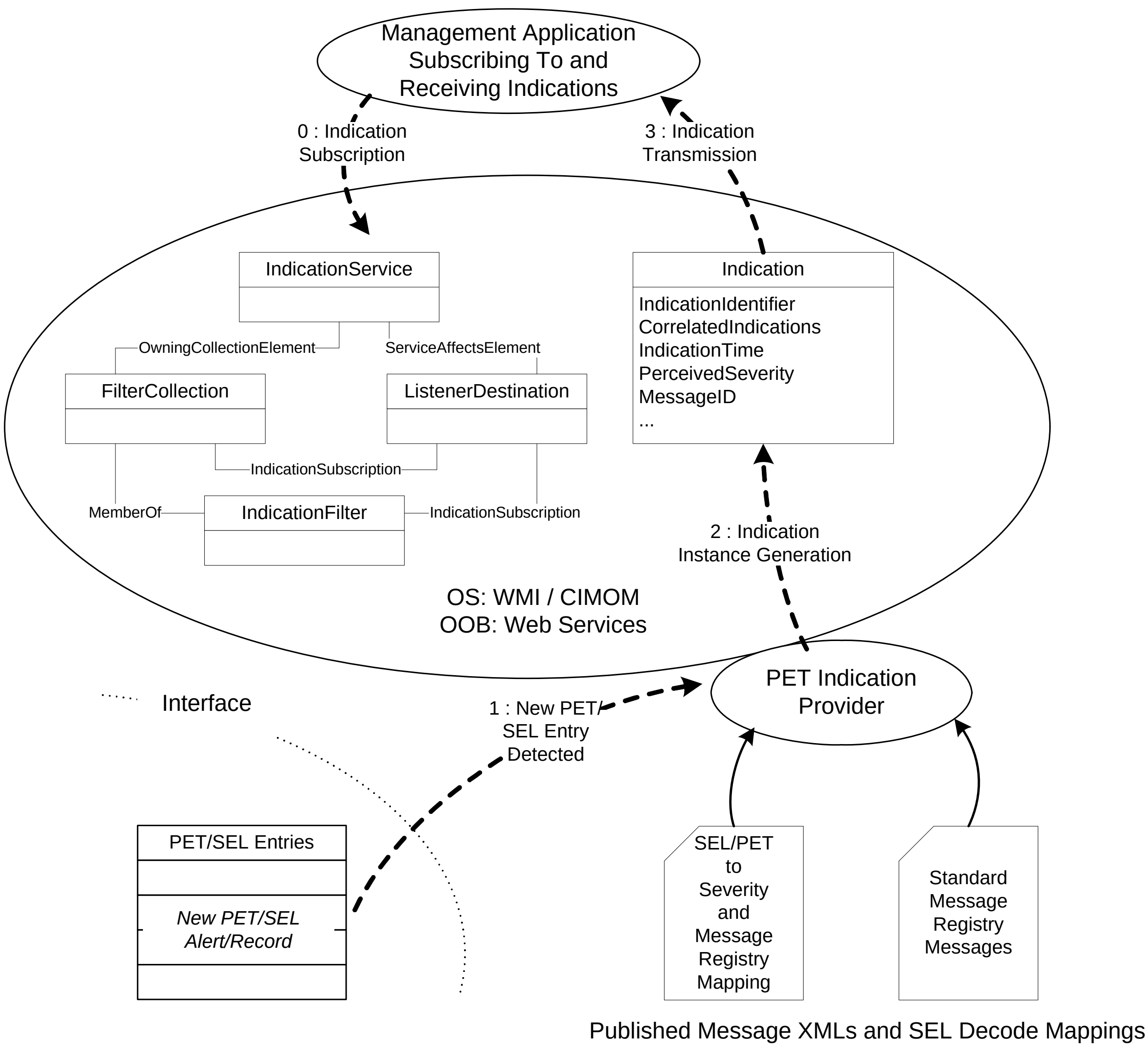

Figure 7 – Standard Message Usage

8 Authentication, Authorization, and Auditing

Security is an important consideration when providing server management. The In-Service/In-Band aspects of server management have been well explored through various standards and implementations, but the cross-section of Out-of-Band and Out-of-Service dimensions raises unique considerations.

While there are many aspects to security, it is feasible to focus only on a finite but achievable list for SMASH specifications. Specifically, these are transport considerations, logon, account properties, account management, credential management and the management of the MAP itself. Note that logon and transport considerations are protocol specific and therefore are described in the sections dedicated to each protocol.

8.1 User Account Management

User account management is an important aspect to the security of the SMASH. Because the user account used for authentication is expected to be the same account used for authorization, it is important to understand the user account model.

User accounts can be created and assigned to a user group.

Three user groups are defined in the architecture. Implementations are required to support the Read Only and Administrator groups. Implementations may support more groups or definable groups. If a user belongs to more than one group, the group with the most privileges is the group used for authorization of commands.

- Read Only – Members of this group are able to perform only read operations. This includes retrieval of data and the ability to perform non-invasive commands such as help, change default target, and change session options.
- Operator – Members of this group are able to perform read, write, and execute operations. Consequently, members of this group can query data. In addition, they can change the state of Managed Elements. They can change setting data, settings, or collections. They cannot create or delete instances or properties directly.
- Administration – Members of this group have read, write, create, delete and execute privileges, as well as all access rights. Members of this group can create, delete or modify users and assign them to groups, unless prohibited by the Authentication, Authorization, Audit Service. Members of this group can also create and delete instances, such as log records.

Currently, no per-target access control lists are defined in the architecture.

The MAP must support the methods and properties to add accounts, remove accounts, show account information, and modify accounts as follows:

- Add Account – create accounts and set their initial state and conditions.
- Remove Account – remove the account completely.
- Show Account – retrieve information associated with the account. Access to other accounts is limited to Administration accounts. Passwords can never be retrieved.
- Modify Account –change the password for the account. Accounts with Administration level can change the password or attributes for any account.

Note that all of these methods and properties are subject to the access rights granted to the user account under which the action takes place.

8.2 Audit

The SMASH supports several kinds of auditing. The MAP itself has a log that can be set to record certain types of information. The exact type of information recorded is implementation dependent.

The MAP also supports access to any logs available within the system, including retrieval of the number and identifiers for logs in the server; insertion, retrieval and removal of records (called events) in the log; and, in some cases, modification of the type of information recorded in the log.

8.3 MAP Management

The CLP and Management Protocol Services are represented as manageable services provided by the MAP. Consequently, these services are manageable, just as any other Managed Element would be.

The services may be disabled completely. The method for re-enabling the MAP is implementation dependent and is therefore outside the scope of the specifications.

Some systems may have dependencies between the MAP and the Managed System. If the MAP is dependent on the Managed System, then resetting the Managed System may result in resetting the MAP. If the system does not have a dependency between the Managed System and the MAP, then resetting the Managed System will not result in resetting the MAP. Any such dependency is implementation dependent.

Each transport and service can be enabled and disabled individually. Each service can be managed independently, allowing for customizable feature and property changes for each service.

The hardware that realizes the interface into the MAP is individually manageable. For example, in the case of an Ethernet interface, the MAC address, IP addresses and parameters, and TCP ports and parameters may all be configured as well as enabled and disabled.

Because the MAP is a container for all of the services and protocols, there are some architectural considerations to keep in mind:

- If the MAP is reset, all other services are reset as well. This implies that all sessions are dropped when the MAP is reset.
- Security information is persistent across MAP resets. Security information includes, but is not limited to, user accounts, account groups, properties, transport information and settings, service settings, and log information and records.

The initial state of the MAP and initial user account is outside the scope of the SMASH.

9 Discovery

Discovery in the SMASH may be divided into three categories of tasks that are generally defined sequentially. The first is discovery of a MAP's services and service access points. . The second is the discovery of the capabilities of the MAP. The third is the discovery of the Managed Elements that are managed by the MAP. This section discusses these three aspects of discovery.

9.1 Service and Access Point Discovery

The Service Location Protocol (SLP) is an industry-standard protocol used to advertise the availability of services on a network. The DMTF has defined an SLP template for WBEM. This template is being updated to include advertisement of WS-Man and SM CLP. A MAP that implements SMASH may support SLP to advertise the services it supports and the associated access URIs.

9.2 Service Capabilities Discovery

After discovering a MAP and its services, the second task is to discover the capabilities of the MAP itself. This section details the different approaches available for the SM CLP and WS-Man protocols.

9.2.1 SM CLP Capabilities

For the SM CLP, SMASH handles discovery of service capabilities by modeling it in profiles. The *SM CLP Profile* contains the classes describing the SM CLP services available within the MAP. To discover the SM CLP capabilities of the MAP is to simply discover the properties and methods, as well as the service access points and transports for the SM CLP. The *SM CLP Specification* and other specifications indicate how to query and alter the values of the properties for the services within the MAP.

9.2.2 WS-Man Capabilities

For WS-Man, a client can use the Identify method to query the capabilities. The Identify method is defined in WS-Management [7]. A management client can subsequently send the Identify message to the TCP port in use by WS-Man to learn the protocol version, the product vendor, and product version of the service. These are provided in the IdentifyResponse message in the wsmid:ProtocolVersion, wsmid:ProductVendor, and wsmid:ProductVersion elements, respectively. Note that the TCP port in use is either known *a priori* or extracted from the SLP response.

A SMASH MAP supports the Identify method on each registered access port that it supports. See SMASH *Implementation Requirements Specification* [3] for the complete list of registered ports.

SMASH defines extension elements as children of the IdentifyResponse element in addition to the child element defined in WS-Management [7]. For details of these elements, see SMASH *Implementation Requirements Specification* [3].

9.3 Managed Element Discovery

The final aspect of discovery is how a Client discovers which Managed Elements are managed by a particular MAP. Fortunately, this is a capability that exists in the protocols in use today. SM CLP and WS-Management provide operations to determine the profiles and Managed Elements within the management domain of the MAP. These operations are well documented in their individual specifications.

10 Conclusion

SMASH is one component in a suite of specifications that delivers the architecture, addressing methodology, profiles, Command Line Protocol, and discovery mechanisms necessary to manage the full range of current and emerging servers in enterprise environments.

The SMASH contains the models, mechanisms, and semantics for managing servers in the data center, regardless of service state. This includes the architectural, service, and operations models, and covers boot and firmware updates as well as service discovery. The profiles contain the required classes, instances, properties, and methods necessary to manage systems. The combination of the profiles with the addressing methodology determines the format of the target addressing convention for compliant systems. Together, they deliver the syntax and semantics necessary to manage servers.