Systems Management Architecture for Mobile and Desktop Hardware White Paper

Version 1.0.0a

Status: Informational

Publication Date: February, 2007

DSP2014

Version 1.0.01i
Publication Date: February, 2007
DSP2014
Status: Informational

Abstract

The Desktop and mobile Architecture for System Hardware (DASH) is a DMTF Management Initiative that represents a suite of specifications which standardize the manageability interfaces for mobile and desktop hardware. The DASH suite of specifications defines the interfaces for management in the form of protocols and profiles for representing mobile and desktop hardware.

This document is an architectural white paper and describes the concepts used in managing mobile and desktop platforms which adhere to the DASH Implementation Requirements [2].

Acknowledgments

The following persons were instrumental in the development of this white paper:
Bob Blair – AMD, Jon Hass – Dell, Jeff Hilland – HP, David Hines, - Intel, Hemal Shah - Broadcom.

Table of Contents

List of Figures

1 Introduction

This document is an introduction into the architectural framework required for managing desktop and mobile systems hardware in the enterprise environment. This document lays forth the basic principles required for understanding and implementing the DMTF Web Services for Management (WS-Management) interface as applied to this environment. The framework is composed of technologies defined in multiple standard specifications, including the WS-Man Specification [1], the DASH Implementation Requirements Specification [2], and a variety of profiles (Section 7) which are applicable to this environment.

The focus of this architecture is to enable the management of desktop and mobile computing resources in a standard manner across any Manageability Access Point implementation, independent of operating system state.

1.1 Target Audience

The intended target audience for this document is readers interested in understanding management through Web Services of desktop systems, mobile systems, thin clients and bladed PCs as well as desktop and mobile systems management architecture in general.

1.2 Related Documents

[1] DSP0226, Web Services for Management (WS-Management), Version 1.0, 2006-03-14.

[2] DSP0232, Desktop and mobile Systems Management (DASH) Implementation Requirements, Version 1.0.

[3] DSP2001, SMASH CLP White Paper, Version 1.1, 2006-12-12.

[4] DSP0227, WS-Management CIM Binding Specification Preliminary, Version 1.0.0b. 2006-08-09.

[5] DSP0230, WS-CIM Mapping Specification Preliminary, Version 1.0.0c, 2006-08-09.

[6] Hypertext Transfer Protocol -- HTTP 1.1, RFC 2616, IETF, June 1999.

[7] HTTP over TLS 1.0, RFC 2818, IETF, May 2000.

[8] HTTP Authentication: Basic and Digest Access Authentication, RFC 2617, IETF, June 1999.

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

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

[11] DSP1054, Indications Profile, Version 1.0.

[12] DSP0136, Alert Standard Format (ASF) Specification, Version 2.0.

[13] DSP1033, DMTF Profile Registration Profile, Version 1.0.

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

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

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

[17] CIM Schema, Version 2.15.0.

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

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	An autonomous profile defines an autonomous and self-contained management domain. This includes profiles that are standalone, or have relationships to other profiles
Common Information Model	The DMTF Common Information Model (CIM) is an 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.
CIM Profile	A profile is 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 MAP.
Common Information Model Object Manager	A CIM-capable implementation.
Component Profile	A component profile 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 of a packet.
Extensible Markup Language	Extensible Markup Language (XML) is a simple, very flexible text format 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	The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which 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.

Term	Definition
In-Band	Management that operates with the support of hardware components that are critical to and used by the operating system
In-Service	Management that operates with the support of software components that run concurrently and are dependent on the operating system.
Internet Protocol	The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.
IP Security	A suite of protocols for securing Internet Protocol (IP) communications.
Manageability Access Point (MAP)	A collection of services of 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 which 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 systems 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
Remote Management and Control Protocol	A protocol used for client control and discovery functions.
SOAP	A lightweight protocol intended for exchanging structured information in a decentralized, distributed environment.
Transmission Control Protocol	The Transmission Control Protocol (TCP) is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications.
Transport	The layers of the communication stack responsible for reliable transportation of commands and message from the Client to the MAP
Transport Layer Security	The TLS protocol 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.

Term	Definition
User	The set of human users and Management Clients which interact with the Transport Client in order to manage a Managed System through a Manageability Access Point. Human users include Administrators, Operators, and Read-Only Users.
WS-CIM Mapping	A specification that provides the normative rules and recommendations that describe the structure of the XML Schema, WSDL fragments and metadata fragments corresponding to the elements of CIM models, and the representation of CIM instances as XML instance documents.
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.
Web Services	A Web service is 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	WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, it 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-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-Transfer	A general SOAP-based protocol for accessing XML representations of Web service-based resources.

161 1.4 Acronyms and Abbreviations

Term	Definition
ASF	Alert Standard Format
CIM	Common Information Model
CIMOM	Common Information Model Object Manager
DASH	Desktop and mobile Architecture for Systems Hardware
ESP	Encapsulating Security Payload
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over TLS
IP	Internet Protocol

Term	Definition
IPSec	IP Security
MAP	Manageability Access Point
RMCP	Remote Management and Control Protocol
SMASH	Systems Management Architecture for Server Hardware
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TCP/IP	See TCP and IP
TLS	Transport Layer Security
XML	Extensible Markup Language

2 Architecture Overview

Desktop and mobile systems management in today's enterprise environments is comprised of a disparate set of tools and applications which administrators can use to manage the multitude of networked desktop and mobile computers. In many cases, these tools are specialized and adapted to each individual environment, installation and product in the environment.

Currently, the CIM Schema provides a feature-rich systems management environment. In its current form, it also places a burden on those vendors attempting to implement the CIM Schema and CIM-XML Protocol to support systems hardware management. This has resulted in lack of interoperability and acceptance of solutions in the desktop and mobile systems hardware management solution space, 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 system's representation and management of the system.

The Desktop and mobile Architecture for System Hardware (DASH) Management Initiative supports a suite of specifications which include architectural semantics, industry standard protocols and a set of profiles to standardize the management of desktop and mobile systems independent of machine state, operating platform or vendor. By creating industry standard protocols, interoperability is facilitated over the network and the syntax and semantics of those protocols are facilitated to be interoperable by products which adhere to those standards. Because it is based on the CIM Schema, the DASH Management Initiative (hereafter referred to as DASH) leverages the richness of CIM. By creating industry standard profiles, the richness of the CIM Schema can be applied in a consistent manner by all vendors.

Extra emphasis has been placed in the development of DASH to enable lightweight implementations which are architecturally consistent. This has been done to enable a full spectrum of implementations without sacrificing the richness of the CIM heritage. This includes software-only solutions and small footprint firmware solutions. Emphasis has been placed on ensuring that these implementations will be interoperable, independent of implementation, CPU architecture, chipset solutions, vendor or operating environment.

2.1 Principal Goals

One goal of DASH is to enable the same interfaces independent of system state. To this end, a Service Model is referenced in Section 2.2 to illustrate that, independent of Service Access Point or operating system state, the same protocols can be used for systems management.

Another goal of DASH is to enable the same tools, syntax, semantics and interfaces to work across a full range of products – traditional desktop systems, mobile and laptop computers, bladed PCs as well as "thin clients". Therefore, we have encompassed considerations for these products in our initial architecture and plan to include support for them in the on-going profile development effort.

2.2 Service Model

Fundamental to the DASH is the underlying goal to unify the experience achieved through out-of-band mechanisms with those available via the operating system. To achieve this goal, DASH has adopted the Service Model as Described in the SMASH White Paper [3]. The definitions, terms and model for In-Band, Out-of-Band, In-Service and Out-of-Service documented in the SMASH White Paper [3] apply to DASH.

3 Desktop and Mobile Management Architecture Model

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

The goal of the architecture is also to describe systems management in abstract terms for all desktop and mobile systems. This means it is implementation agnostic and spans the spectrum of the supported platforms.

3.1 Architectural Model

This section introduces the overall DASH Architecture Model (see Figure 1). 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 which are noticeable by the Client need to be accounted for in order to have a complete model.

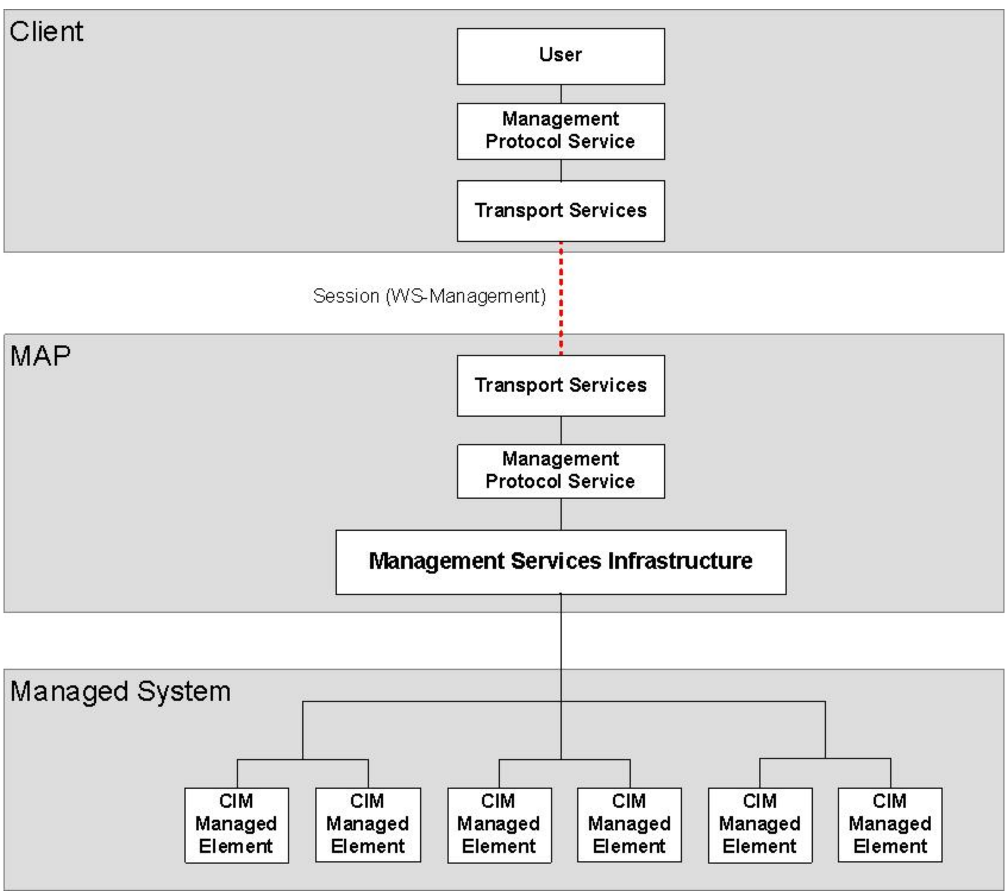

Figure 1 - DASH Management Initiative Architecture Model

Figure 2 depicts an example implementation that emphasizes the components within the MAP which are noticeable when implemented within a WBEM context. While the entities described are not required to exist as independent entities, their existence is evident from 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 DASH of a human Administrator interacting with Management Client Services. It also includes Authentication, Authorization and Audit components within the MAP that are expected to be accessible through the protocols. In addition, the Operation Invocation Engine indicates that the operations 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 DASH.

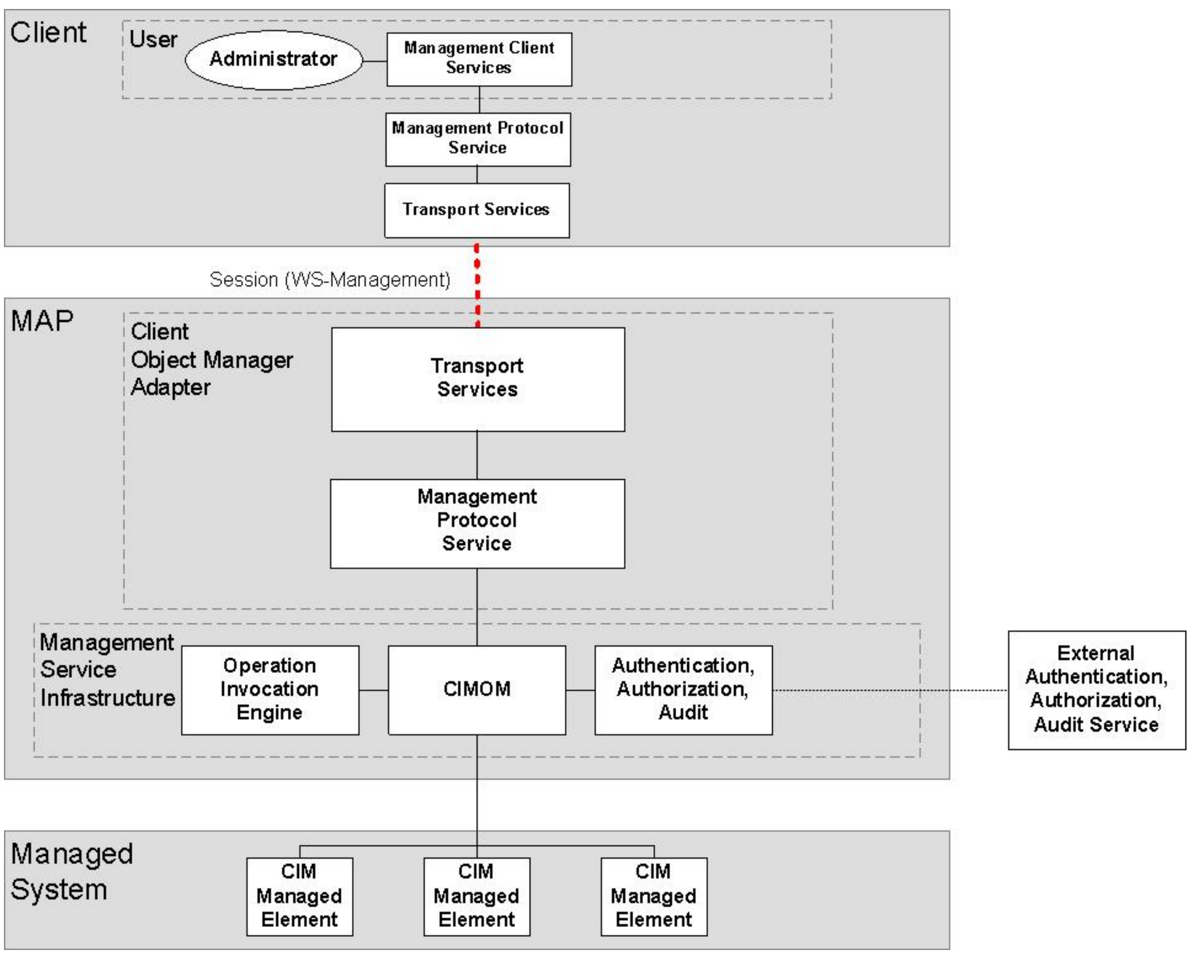

Figure 2 - Example MAP Implementation Architecture

The following sections describe the components found in Figure 1 and Figure 2.

3.2 Client

A Client is a logical component that manages a system via 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 DASH Implementation Requirements [2].
- Accessing a MAP using the DASH defined management protocol. 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.
  - Terminating a session with a MAP.

3.2.1 User

The User in this model represents an instance of a Management Client Services and an Administrator.

3.2.1.1 Management Client Services

A Management Client Services represents a program of some type, such as an 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 WS-Management messages. Interaction between the Administrator and the Management Client Services is outside the scope of this document.

3.2.1.2 Administrator

This represents the human interacting with the Management Client.

3.2.2 Transport Services

The Transport Services in the 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 transport session establishment, authentication, and authorization.

The DASH Implementation Requirements Specification [3] contains mappings for HTTP and HTTPS. Other transports are not precluded but are outside of the scope of DASH.

3.3 MAP

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:

- 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 of the scope of DASH. The interfaces within the MAP are outside of the scope of DASH.

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

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

3.3.1 Management Service Infrastructure

The Management Service Infrastructure is a logical entity that contains the core services set of the MAP that implement a CIM Server. It is primarily comprised of the functions described below.

3.3.1.1 CIMOM

The Common Information Model Object Manager (CIMOM) represents the components 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 of the operations submitted to the MAP. Operations are discussed in more detail in Section 4.1.

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. It also is responsible for coordination of audit information of the operations and tasks taking place within the MAP. Note that this is a service internal to the MAP and interaction or coordination with any external service components is outside the scope of this architecture.

3.3.2 Client Object Manager Adapter

This represents the collection of entities required to process the WS-Management messages and ensure responses are generated and, as required by the messages, interact with the Management Service Infrastructure to accomplish the requests and produce the information contained in the responses. It consists of the Transport Service and the Management Protocol Service.

3.3.2.1 Transport Services

This represents the transports and lower layer protocols over which the Management Protocol Service is carried. This includes transport session establishment, authorization, and authentication.

It also represents the entity which encrypts/decrypts the data stream. This happens as part of the transport mechanism in this architecture. The two defined transport services for DASH are HTTP [6] and HTTPS [7].

Note that the DASH Implementation Requirements Specification [2] is the definitive reference for requirements on the Transport Service.

3.3.2.2 Management Protocol Service

This represents the endpoint of the Management Protocol within the MAP. The Management Protocol for DASH is WS-Management. WS-Management messages will be received here and turned into internal operations within the MAP. This entity is responsible for receiving messages and transmitting responses which are compliant with the WS Management Specification [1].

The interface between the Management Protocol Service and the Management Service Infrastructure is implementation-dependent and thus the interface itself is out-of-scope of DASH.

3.3.3 External Authentication, Authorization, Audit Service

The External Authentication, Authorization, Audit Service represents the entity which 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 of the current scope of DASH. In addition, the interface between the MAP and the Security Service is outside of the current scope of the DASH. Note that this is distinct from the Authentication, Authorization, Audit component of the MAP itself since (see Section 3.3.1.3) 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, system, or platform. Managed System types include desktop, workstation, laptop, tablet, thin client, bladed, and virtual systems.

One or more Managed Elements and/or Resources – or collections thereof – are managed by a single MAP. There may also be more than one Managed System within the domain of a 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, physical or logical entities within a Managed System which the operations will manipulate.

Direct interfaces for Managed Element access are outside of the scope of DASH.

4 Management Models

This section contains the models which are useful in understanding DASH.

4.1 Operation Model

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

It is important to understand that in the MAP operation model, the term operation is often used. In CIM, operations correspond to property accesses using intrinsic methods and extrinsic method invocations. The reader should understand the class CIM_ConcreteJob (Core Schema), which can be used to make operations visible to management clients.

4.1.1 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 included here.

MAPs are responsible for managing the elements for which they claim responsibility. This does not imply that they will actually execute the method or modify the property included in the operation, but MAPs are responsible for dispatching, tracking, ensuring the completion of, and delivering the results of the operation.

The MAP is responsible for ensuring the message is syntactically correct. It may pass the parsing to one of its subcomponents or another system component, but it is the MAP that has the responsibility for ensuring that the implementation complies with the protocol.

The MAP is responsible for operation handling. It may delegate the actual operation but it is responsible for handling 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 its scope. Operations which target MEs which 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 valid for this 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 ensures that the request is properly formed and conveyed, but relies on the feedback from the ME for the assessment of operation validity.

4.2 Operation Handoff

Operations within the MAP are not directly visible to the Client. The fact that they exist, are initiated, can be cancelled, can complete and can be deleted can be made visible by the implementation if it supports CIM_ConcreteJob, which is returned when a CIM method will complete asynchronously. In addition, their status can be retrieved.

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

When operations are modeled in CIM, they have identifiers. The CIM_ConcreteJob class is used to represent operations, so the identifier is that of a CIM_ConcreteJob instance. The term Job ID represents the identifier of that CIM_ConcreteJob instance. The status of the job can be retrieved with a command or message using the Job ID. The MAP keeps track of all active operations.

When an operation modeled by CIM_ConcreteJob is complete, the properties of the instance of CIM_ConcreteJob determine if the instance persists or is immediately recycled. Specifically, the TimeBeforeRemoval property in CIM_ConcreteJob is used to determine the amount of time that the instance persists.

Operations which result in a Job being spawned are able to handle a cancellation request. Sometimes the response to the cancellation will be an error, such as in the case of 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.

The Client can then determine the status of the operation and whether or not the operation is complete. This can be done through a query operation on the operation queue using the Job ID. 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

In the architecture, the MAP implements an Operation Service which logically contains an Operation Queue. This is a FIFO queue which contains all of the operations to be processed within the MAP. All current sessions submit operations to this single queue. The MAP provides access to the capabilities of this queue and the profiles. The properties of the Operation Queue are expected to vary from implementation.

Ordering is with respect to operation initiation and is implied by the queue. Ordering of operation initiation is guaranteed but no such guarantee is made on operation completion.

The MAP's operation queue depth varies from MAP to MAP. The minimum acceptable operation queue depth is equal to one operation or message. Some implementations may support multiple outstanding operations; others may not. Should the queue become full, the MAP is responsible for communicating this resource constrained condition.

Implementations that support asynchronous operation completion support the class CIM_ConcreteJob, which provides detailed information about the operation, including status. A reference to an instance of CIM_ConcreteJob is returned by a CIM method when it will complete asynchronously. A Client that receives such a reference can use it to query this information.

4.4 Multi-session capabilities

An important aspect of MAP operations management is to be able to support simultaneous management 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 management sessions. Therefore, DASH supports multiple concurrent management sessions.

The number of ports offered to transports from the Management Services Core for each protocol supported is one per transport 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 multi-session capabilities is the ability for operations to be visible independent of the transport that initiated them. This implies that there is one global operations (job) queue per MAP. The MAP is responsible for routing the results of operations to the appropriate session. But if the command or message spawns a job, then any session should be able to discover the details about the job in question, by querying the job using its 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.

5 Protocol Support

DASH 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 DASH.

5.1 Management Protocol

DASH supports the Web Services for Management Protocol, as defined in the WS-Management Specification [1], as the management protocol for transporting DASH 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 DASH is shown in Figure 3. The WS Management stack is based on the Web Services. The network and physical layers are the two bottommost 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, and HTTP 1.1, which provides user authentication and request-response semantics. TCP and HTTP 1.1 are required by DASH. TLS support is conditional on support for security profiles that require it. Section 9 describes DASH security profiles in more detail.

At the next layer, SOAP/XML messaging is handled. The security profiles specified in the DASH Implementation Requirements Specification [2] 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. The DASH profiles are mapped over the WS Management protocol stack using the WS Management CIM Binding [4] (which is defined in terms of WS-CIM [5].

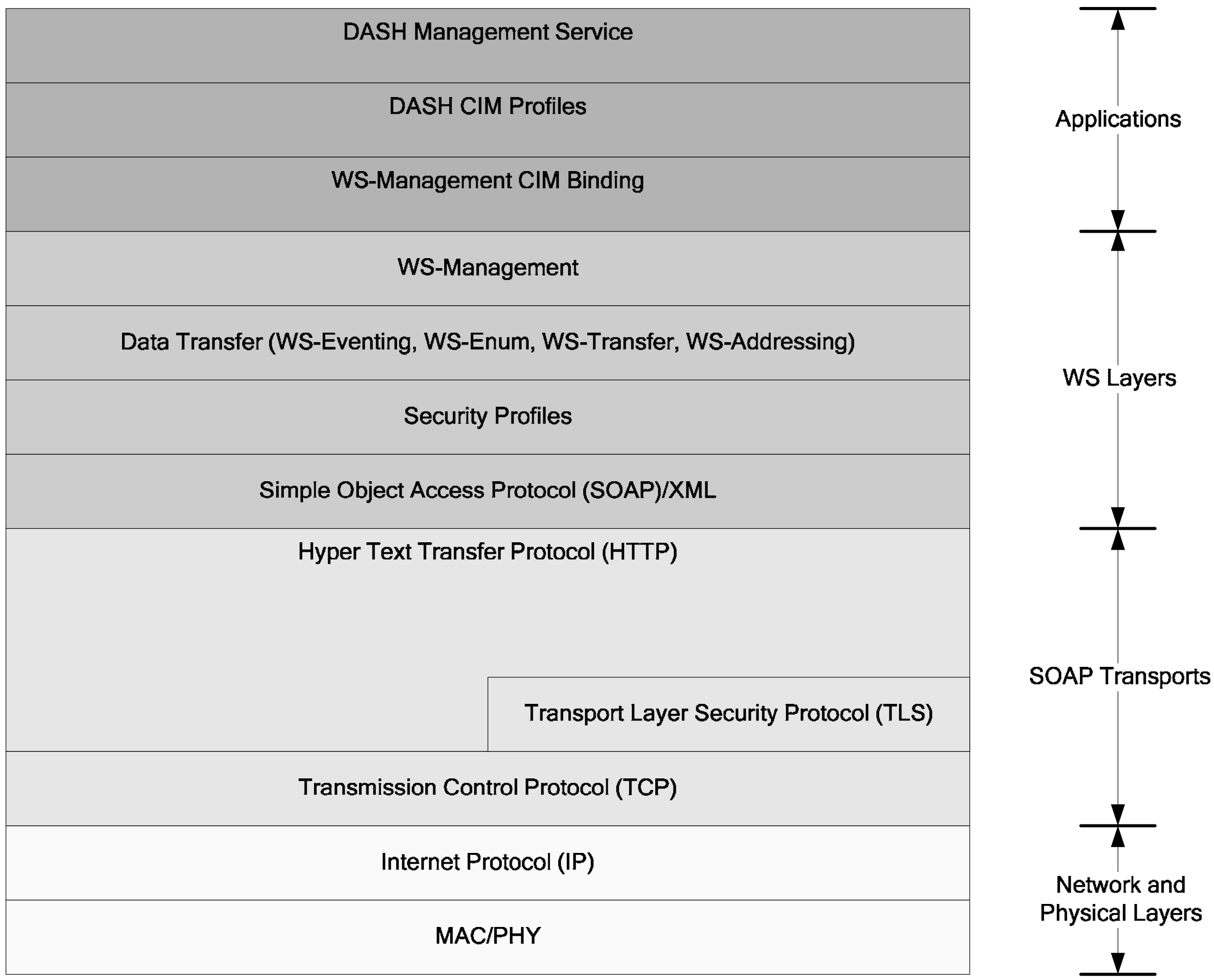

Figure 3 – DASH Protocol Stack

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 the DASH Architecture).

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 operation 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. WS-Management in addition 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 con-

sumer 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:* to initiate an enumeration and receive an enumeration context.
   b. *Pull:* to pull a sequence of elements of a resource.
   c. *Release*: to release an enumeration context (graceful).

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.

5.2 Transport Protocol

The WS-Management protocol is transport-independent but it specifies HTTP 1.1 [6] and HTTPS [7] as the common transports for the interoperability.

DASH 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 2-way authentication in the form of basic and digest authentication (RFC 2617) [8]. 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 (RFC 2246) [9] 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 [9].

5.3 WS-Management – CIM Binding

The WS-Management CIM Binding specification defines the binding between the Web Services representation of CIM (defined in the WS-CIM Mapping Specification [5]) and WS-Management. This binding encompasses:

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

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

6 Eventing

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

DASH targets the use of WBEM-based event notification mechanisms in conjunction with greater standardization of event message content. Traditionally, Simple Network Management Protocol (SNMP) [10] 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 DASH Implementation Requirements Specification [2], in conjunction with WS-Management [1], WS-Management CIM Bindings [4], Profiles and related Message Registry specifications, defines a new level of Web Services based event management and notification.

6.1 Eventing Overview

The CIM model contains alert indication class designs that represent events (described below). The DASH 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 4 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.

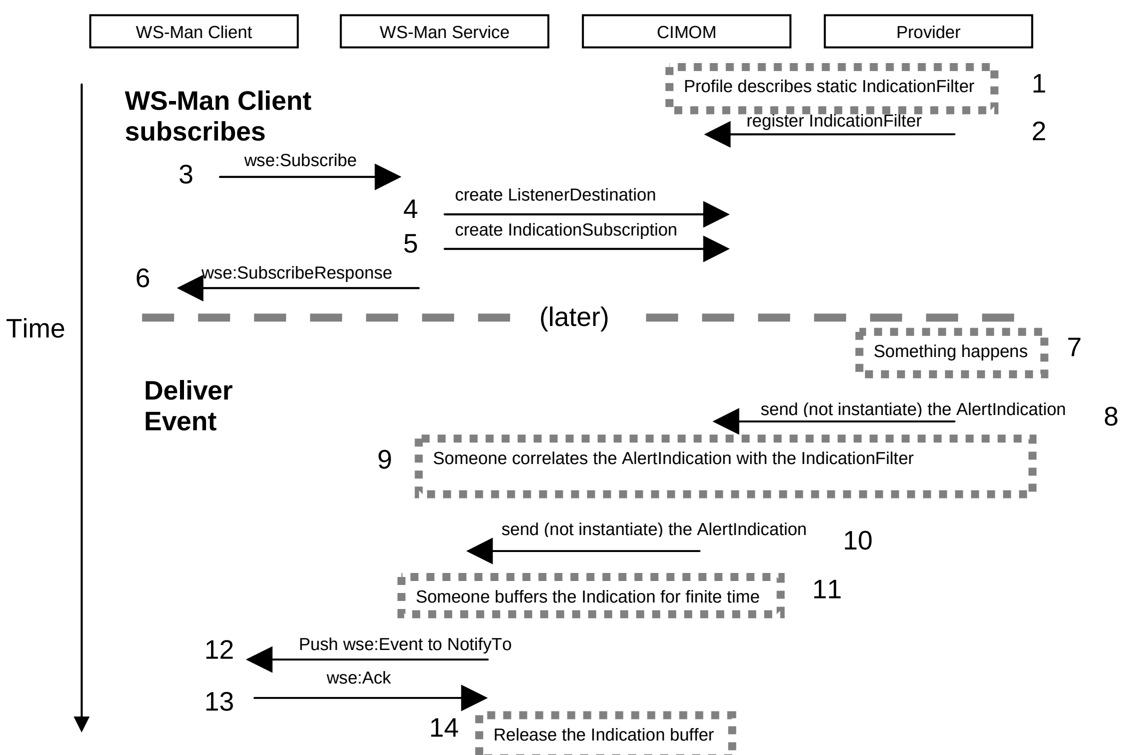

Figure 4 – Indication Activity Diagram

The first sequence of events in Figure 4 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 [4]. 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.

6.2 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 other identifying information in the form of MessageArgs. These 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.

6.3 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 DSP1054 – Indications Profile [11]. Figure 5 represents the actions and resultant representation of an event indication subscription. For a detailed explanation of the classes, please refer to DSP1054 – Indications Profile [11].

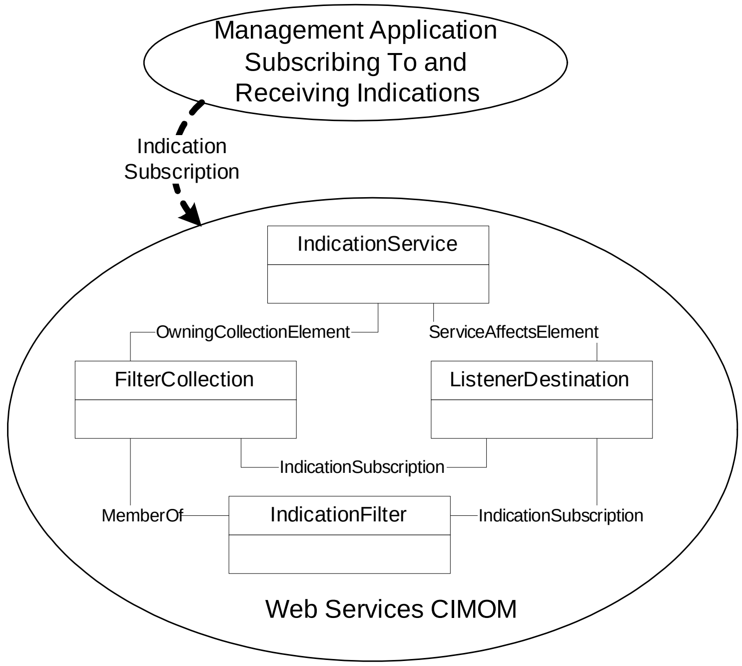

Figure 5 – Event Indication Subscription

6.4 Standardized Message Content

In order to foster greater interoperability between different implementations of management instrumentation and the applications that subscribe for 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. DASH 1.0 uses message registries for the Message IDs, perceived Severity and interpretations of MessageArgs for each MessageID.

7 Profiles

This section discusses the topics of profiles. A brief overview of the purpose of profiles is included. Profiles specify standard support for manageability features, the list of which is in Section 7.2. The autonomous and component profiles are listed in the DASH Implementation Requirements Specification [2], but have been listed for convenience in Section 7.3.

7.1 Overview

DMTF profiles provide the object model definitions for manageability content and architecture models for mapping computer hardware to Common Information Model (CIM) object classes in a way that is consistent between different implementations. These autonomous and component profiles combine to ensure that individual implementations will contain the same object information as appropriate based on their hardware configuration and the elements they manage.

Autonomous and component profiles describe the classes and associations that are used to model a target desktop or mobile system and its manageable elements for DASH. These profiles combine to ensure that all CIM representations of the system are implemented in a consistent fashion across multiple vendor offerings and architectures. The profiles lay out the standard CIM-based modeling approaches defined for managed system elements. Profiles include object and association behavioral definitions that specify how system components are to be modeled in order to produce consistent implementations. Another benefit of profiles is that they effectively prune the many classes, associations, methods and properties in the CIM Schema to a base consensus model.

The use of the categories of "Mandatory", “Conditional” and "Optional" for classes, associations, properties and methods draws the distinction, both for the Manageability Access Point (MAP) Web Services implementation and the Client, as to what must be supported and what can be expected with respect to interoperability. This results in not only consistent implementations but sets expectations on the levels of support within the industry.

7.2 DMWG Targeted Manageability Features

The following is the list of manageability features targeted in the 1.0 version of the DASH Implementation Requirements Specification [2]. These features are represented by using the profiles listed in Section 7.3.

- Power Control
- Boot Control
- WS-Eventing Push Indications (functionally equivalent to PET alerts)
- Correlatable System ID
- Firmware Version information
- Hardware information
  - Chassis model/serial, CPU, Memory, Fan, Power Supply, Sensor
- Login and UserID credentials as well as Roles and Privileges

7.3 DMWG 1.0 Profiles

This section contains the list of autonomous and component profiles in the DASH Implementation Requirements Specification [2]. They have been listed for convenience in this section along with a description.

Profile Name	Description	Manageability Feature
Base Desktop Mobile	Autonomous profile for describing desktop or mobile systems	Hardware Information, Correlatable System ID
Physical Asset	Physical component, chassis, card, FRU representation	Hardware Information
Boot Control	Boot sequence representation and configuration	Boot Control
Power State Management	System power state representation and control	Power Control
Software Inventory	Representation of software/firmware identification and version information	Firmware Version Information
CPU	Processor representation and configuration	Hardware Information
System Memory	System memory representation	
Fan	Fan status and component representation	
Power Supply	Power supply status and component representation	
Sensor	Sensor status and component representation	
Role Based Authorization	Role and privilege representation and management	Login and UserID credentials as well as Roles and Privileges
Simple Identity Management	User identity representation and management	
Indications	Subscription, listener destination, event filter and indication representation and management	WS-Eventing Push Indications (functionally equivalent to PET alerts)

8 Discovery

8.1 Discovery Overview

Management clients make use of a variety of discoverable information about managed systems. These pieces of information are typically accumulated across multiple discovery stages. The following is a list of stages involved in discovering managed systems and their management capabilities:

1. Network Endpoint Discovery Stage
2. Management Access Point Discovery Stage
3. Management Capabilities Discovery Stage

Each of these stages is described in more detail in this section.

8.2 Network Endpoint Discovery Stage

A Client may enumerate the participants in a network by finding the endpoints based upon network layer. When done, this step provides a list of network addresses for use in subsequent phases.

Because it is supported by all IP network stacks, ICMP Echo Request/Reply is one of the more common methods of network endpoint discovery.

DASH mandates that implementations support these methods. They are critical in discovering DASH Management Access Points (MAPs).

8.3 Management Access Point (MAP) Discovery Stage

The MAP discovery phase involves discovering DASH MAPs in managed systems. It can be done either pursuant to or in lieu of network endpoint discovery.

DASH-compliant MAPs support the following two-phase process for MAP discovery:

Phase 1: RMCP Presence Ping/Pong. This provides information about the management protocol(s) supported by the MAP. This can be done in a unicast, broadcast, or multicast fashion, as described below.

Phase 2: WS-Management Identify Method. This method provides detailed information about the WS-Management service, but it assumes a priori knowledge of the MAP's network address, and hence is not sufficient in and of itself as a discovery mechanism.

These steps are summarized in Figure 6 and described in more detail below.

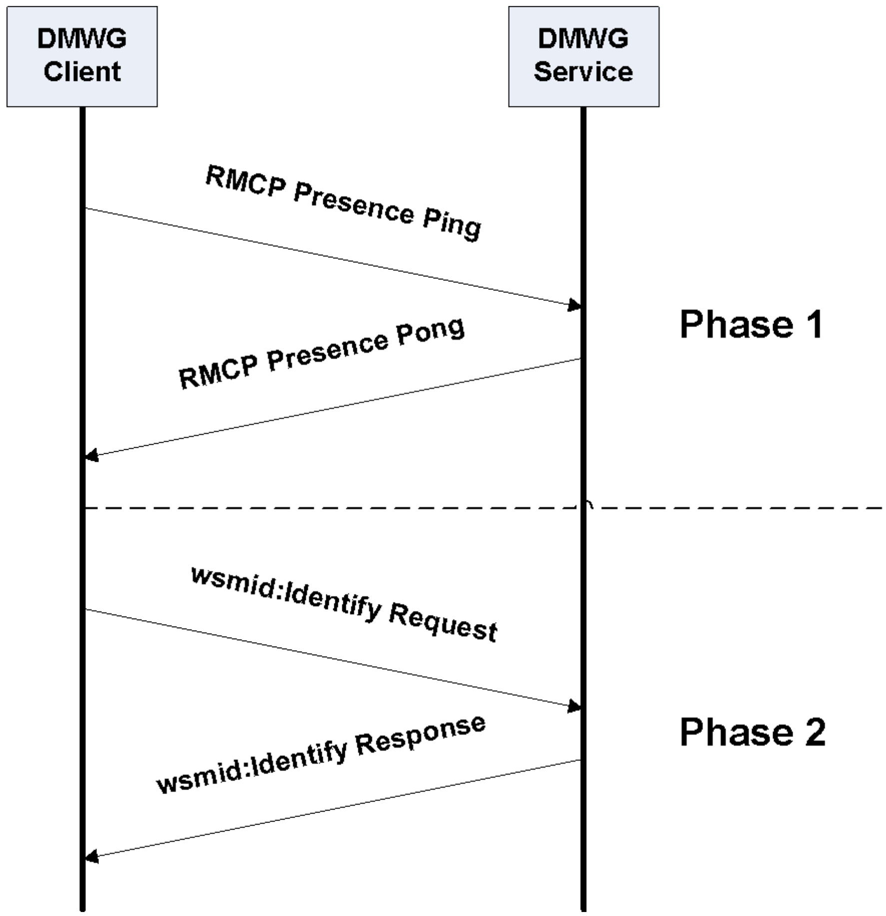

Figure 6 –Two-Phase Management Access Point Discovery

8.3.1 RMCP Presence Ping/Pong

Presence Ping is an RMCP command defined in ASF [12]. It involves a request-response message exchange initiated by a management client (Ping) and completed by a management service (Pong).

DASH implementations support this command on the asf-rmcp well-known UDP port (623). Support of Presence Ping/Pong on the asf-secure-rmcp well-known UDP port (664) is not recommended for a DASH implementation discovery.

The DASH Implementation Requirements Specification [2] defines the ports used for the Ping/Pong for phase 1 discovery. It also indicates the exact format of the Pong to determine if the endpoint supports an out-of-band[2] WS-Management service. An existing bit in the Supported Entities Field identifies support of ASF [12]. One of the key advantages of this method is that it can be used in a heterogeneous environment to discover multiple types of management services.

Because the Presence Ping command is sent to a UDP port, it can be sent to broadcast and multicast addresses as well as unicast addresses. The RMCP Presence Ping/Pong supports the following models:

[2] The network endpoint may also support an in-band WS-Management service. Because the RMCP port was defined to describe out-of-band management services, it is not used to advertise support for in-band services.

1. Broadcast – A single Presence Ping message is sent to either the local or a network-directed broadcast address. All network endpoints that support RMCP Presence Ping/Pong respond with a Presence Pong message. Enterprise network policy may limit the applicability of this approach, in which case, one of the other methods should be used. Network-directed broadcast in particular is frequently disabled in enterprise networks.
2. Unicast sweep – A separate Presence Ping message is sent to each IP address in a range of IP addresses. Each network endpoint that supports RMCP Presence Ping/Pong responds with a Presence Pong message. This approach should always be coordinated with any enterprise security policies designed to prevent Denial of Service (DoS) and other attacks that exhibit similar behavior.
3. Multicast – A single Presence Ping message is sent to a multicast group address. All network endpoints in the group that support RMCP Presence Ping/Pong respond with a Presence Pong message. The group may be defined expressly for discovery purposes, or may be more general-purpose in nature. The routers in the enterprise network need to have multicast delivery enabled, and the group needs to be established and managed.

8.3.2 WS-Management Identify Method

The Identify method is defined in WS-Management [1]. If Phase 1 indicates that the network endpoint supports an out-of-band WS-Management service, the management client can subsequently send the *Identify* message to the DMTF registered TCP port 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.

A DASH MAP supports the Identify method on each registered access port that it supports. See the DASH Implementation Requirements Specification [2] regarding DMWG registered access ports.

DASH defines extension elements as children of the *IdentifyResponse* element in addition to the child element defined in WS-Management [1]. For details of these elements, see the DASH Implementation Requirements Specification [2].

8.3.3 Enumeration of Management Capabilities Stage

The DMTF Profiles Registration Profile [13] specifies methods for enumerating the management capabilities of a CIM-based management access point in a scalable manner. DASH Implementations support the Profile Registration Profile and therefore provide a mechanism for enumerating the set of related management capabilities that is independent of the number of CIM instances supported by the management access point.

9 Security

Security is very important for systems management operations. DASH defines several aspects of security including transport level security, roles and authorizations, user account management, and authentication mechanisms. The transport level security provides machine-level authentication and encryption of payloads contained within the transport messages. The user-level authentication and authorization mechanisms provide the second level of authentication and authorizations for operations allowed for the specific roles.

9.1 Transport Considerations

DASH requires HTTP 1.1 [6] as the transport for the management protocol WS-Management [1]. The security at the transport or network layer provides the message integrity, data origin authentication, and encryption of transport messages. The transport or network layer security mechanisms protect the management protocol messages, management operations, and CIM-based resources/data accessed using the management protocol. DASH defines two classes of security as described below for security at the transport layer and layers below it.

DASH defines two security classes for HTTP 1.1 transport.

1. Class A: The security class A requires HTTP digest authentication for the user authentication. For this class, no encryption capabilities are required.

2. Class B: This class defines three security profiles that are based on either TLS or IPsec with specifically selected modes and cryptographic algorithms. For the class B compliance, the support for at least one the security profiles below is required. The definitions of the three security profiles defined for class B are as below.

    a. HTTP_TLS_1: For this security profile, TLS 1.0 is required for both authentication and encryption. This profile provides two-level authentication and encryption capabilities. The user-level authentication is provided by the HTTP digest authentication. The machine-level authentication is provided by the TLS server and client certificates (X.509 based) where the implementation of the client certification is optional. The encryption capabilities are provided by TLS 1.0. The required cipher suite for this security profile is TLS_RSA_WITH_AES_128_CBC_SHA.

    b. HTTP_TLS_2: For this security profile, TLS 1.0 is required for both authentication and encryption. This profile is based on providing two-level authentication and encryption capabilities. The user-level authentication is provided by the HTTP basic authentication. The machine-level authentication is provided by the TLS server and client certificates (X.509 based) where the implementation of the client certification is optional. The encryption capabilities are provided by TLS 1.0. The required cipher suite for this security profile is TLS_RSA_WITH_AES_128_CBC_SHA. The only difference between HTTP_TLS_1 and HTTP_TLS_2 is the mechanism used for user authentication (HTTP digest authentication for HTTP_TLS_1 security profile and HTTP basic authentication for HTTP_TLS_2 security profile). For HTTP_TLS_2 security profile, HTTP basic authentication used in conjunction with TLS avoids the transmission of credentials in clear text.

    c. HTTP_IPSEC: This security profile is based on combining HTTP 1.1 over IPsec with the HTTP digest authentication. The user-level authentication is provided by the HTTP digest authentication. While, the machine-level authentication and encryption is provided at the IPsec layer below the transport layer. For this security profile, IPsec ESP transport mode is required. This security profile requires the implementation of one of the cipher suites mentioned below:

i. AES-GCM (key size: 128 bits, ICV or Digest length: 16 bytes)

ii. AES-CBC (Key size: 128 bits) with HMAC-SHA1-96

A DASH implementation is required to support at least one of the above security classes. It is recommended that a DASH implementation be Class B compliant for privacy/confidentiality and additional security.

9.2 Roles and Authorization

The access control for the managed resources and management operations is an important aspect of the secure management for DASH. The authorization and access control is based on the roles assigned and privileges associated with the user accounts.

DASH defines the following operational roles.

1. Read-Only User. This is a role that allows a user to only perform query and read operations on the managed elements. The read-only user is not allowed to modify data/properties, settings, and setting data. For the managed elements and objects, the read-only user can neither change the state of the managed elements/objects nor create/delete object instances or properties. The read-only user can not perform user account management.

2. Operator. This is a role that allows a user to perform read, write, and execute operations on the managed elements. An operator is allowed to change data/properties, settings, and setting data as well as change the state of the managed elements. The operator is not allowed to create/delete object instances or properties through direct manipulation of object instances or properties. Another restriction that applies to an operator role is the inability to perform user account management.

3. Administrator. This role is a superset of operator role with the additional capability to create/delete object instances or properties and perform user account management. Similar to a user with the operator role, a user with the administrator role is allowed to perform read, write, and execute operations; change data/properties, settings, and setting data; and change state of the managed elements. The administrator can perform user account management (create/delete/modify user accounts) if supported by the implementation.

A DASH compliant service is required to support the administrator role. In addition, an implementation may support the operator and/or read-only user roles. DASH does not define any functionality-based roles but an implementation is free to support them. Also, DASH does not require a separate role for auditing the DASH operations.

9.3 User Account Management

The user account management is another important security aspect of the DASH architecture. The authentication and authorization mechanisms are tied with the user account management. DASH supports one or two-levels of authentications. The authorization model supported by DASH is role based.

Each user has the ability to modify his or her credentials. But only the administrator is allowed to perform account management for all users. The following are the operations supported for user account management.

1. Create an account
2. Delete an account
3. Enable an account
4. Disable an account
5. Modify the privileges of an account
6. Modify password of an account
7. Change the role of an account
8. Create a group of accounts
9. Delete a group of accounts
10. Add an account to a group
11. Remove an account from a group
12. Change the role of a group
13. Modify the privileges of a group

The modifications of privileges covers the changing of bindings between accounts/groups and roles. The privileges defined for DASH are static privileges. In other words, the bindings of privileges to roles can not be changed dynamically. For the administrator role, the following minimum set of operations is required to be supported for the user account management.

1. Create an account
2. Delete an account
3. Change the associations of roles and accounts

9.4 Authentication Mechanisms

The three different types of authentication mechanisms considered are:

1. Machine-level authentication. This is used to authenticate the machine that is accessing the service. The machine-level authentication uses machine-level credentials (keys, certificates..) for the authentication. The machine-level authentication does not authenticate a particular user or a user session.
2. User-level authentication. This is used to authenticate a particular user. It is typically based on the usernames/passwords and it may involve a third-party which 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 is typically an out-of-band authentication mechanism where the third-party is used to verify the user credentials. The credentials used for the authentication are issued by the third-party (like a certificate authority). The user provides these credentials during the authentication process. The third-party is involved in

authenticating a user (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.

DASH 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. So, any configuration for third-party authentication happens outside of the CIM profiles defined in DMWG. But, the identity can be incorporated in the model. A DASH implementation can choose to support multiple levels of authentication.

9.5 Authorization

DASH uses a role-based authorization model. The scope of the authorization is within an authenticated session. For a TLS session, the HTTP digest authentication may happen multiple times within a session. Each operation performed by the user is authorized prior to the execution of the operation. An unauthorized operation (the operation which fails the authorization) does not change any state or data of the managed resources.

10 Use Cases

These Use Cases describe representative common tasks that can be performed using DASH 1.0.

10.1 User Accesses the DASH Service as an Administrator

The user presents credentials to the DASH service using one of the DASH-mandated WS-Management mechanisms. Whether the credentials allow him to use the service as an administrator depends on steps taken earlier to establish an association between the instance of CIM_Identity that corresponds to the credentials and the instance of CIM_Role that corresponds to the Administrator role. Use cases for establishing identities, roles and privileges are described in DSP1039 (Role Based Authorization Profile). DASH-specific values for the authorization classes are described in <DASH Wrapper Spec>. A general discussion of authentication and authorization in DASH is found in section 9 above. Note that DASH 1.0-compliant services are not required to implement the Role Based Authorization Profile.

10.2 Client discovers the capabilities of the DASH Service

There are three sets of discoverable capabilities in DASH 1.0.

The first is simply whether there is a DASH service at a particular network endpoint. This can be done using the RMCP Presence Ping mechanism described in section 8.3.1 above.

The second is the set of capabilities advertised in the response to a WS-Identity query. The WS-Identity query is described in the WS-Management specification; DASH-specific capabilities are described in detail in <DASH Wrapper Spec>.

A general discussion of DASH discovery is found in section 8 above.

A third set of capabilities discoverable in DASH 1.0 is the list of supported profiles. The Profile Registration Profile (DSP1033) describes how to find the Profiles that are supported in an implementation. The “Scoping Class” described in the Profile Registration Profile is CIM_ComputerSystem for the DMTF Service. The “Central Class” is also CIM_ComputerSystem, so either the Scoping Class Methodology or the Central Class Methodology may be used to find the list of supported profiles.

10.3 PC Needs to be woken up remotely on a wired network

PCs that have Wake On LAN configured will awake on receipt of the Magic Packet. DASH provides a similar capability through the Out Of Band DASH Service. An administrator, using a SOAP client program which perhaps has been extended for DASH, formulates a query to locate the computer systems at a Management Access Point accessible through a WS-Management Service transport address. If the target computer is located, the Administrator fetches the associated CIM_PowerManagementCapabilities instance, and examines the PowerChangeCapabilities and property.

If the computer supports setting its enabled state to 2 (enabled), the Administrator executes the extrinsic method RequestStateChange() on the computer instance.

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile

Step	Actor	Action	Notes
2	Client	Navigate from the target instance of CIM_ComputerSystem to the instance of CIM_PowerManagementService using the CIM_AssociatedPowerManagementService association.	See Power State Management Profile
3	Client	Using the instance of CIM_PowerManagementService, navigate to the instance of CIM_PowerManagementCapabilities through the CIM_ElementCapabilities association.	Ditto
4	Client	If the PowerChangeCapabilities property array contains the value 3 (Power State Settable) and PowerStatesSupported contains the value 0 (On), then waking the computer is supported by the MAP.	Ditto
5	Client	Invoke the RequestPowerStateChange( ) method of the instance of CIM_PowerManagementService with the PowerState argument set to 2 (Power On).	Ditto
4	DASH Service	Using internal interfaces, effect the requested state change and return the appropriate response to the Client.	
5	Client	Examine the returned values to determine if the PC was woken.	

10.4 PC needs to be woken up remotely on a wireless network

Wake On LAN is not commonly implemented on wireless NICs because of the power requirements of keeping the radio on, but some systems can be configured to periodically wake up and listen for traffic on the wireless connections. While the wireless NIC is powered, the DASH Service can act as described in use case 10.3.

10.5 PC will not boot

An administrator can be notified of a PC boot failure if he subscribes to these alert indications:

MessageID	Event
20	Preboot User password violation
23	Network Boot password violation
175	No Bootable Media
176	Non-bootable Media
177	PXE Server Not Found
178	User-timeout on boot

In DASH 1.0 can expose the following capabilities to help an administrator diagnose the problem and fix it remotely:

- Examine the Boot Control settings to see if the boot parameters, including boot source, are correct.
- Examine the Software Inventory to make sure there is an appropriate BIOS installed. If the installation tracks it, this can also be used to check that there is an OS locally installed.
- Examine the Physical Asset inventory.
- Check to see that the CPUs and memory are recognized as present and are compatible.
- Make sure that the system environmentals and power will allow a boot.
- Examine a record of indications sent from the PC if it ever was operational.

- Set the boot source to a different image using the Boot Control settings.
- Reset the system.

10.6 PC will boot, but OS hangs

An administrator can be notified when an OS fails to load by subscribing to indication message 178 (User Timeout on Boot).

The administrator has all the same tools and capabilities available to diagnose OS hangs as he does to investigate boot failures (see section 10.5). In addition, OS-managed log records may also be accessible.

10.7 Query PC assets while OS hung or absent

An administrator can query the DASH service for information about Software assets.

Depending on the implementation, some information that is available when the OS is running may not be available while an OS is absent. A number of use cases are described in detail in the Software Inventory Profile, but the following steps will return all available software information.

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Locate the packaging of the Computer by following the ComputerSystemPackage association.	See Software Inventory Profile
3	Client	Find each instance of CIM_SoftwareIdentity associated through the CIM_InstalledSoftwareIdentity association. Each instance represents a piece of software or firmware installed on the computer. To find all software whether or not it is installed, get the instance associated with the computer through the CIM_ElementSoftwareIdentity association.	Ditto
4	Client	To find information about the software, get properties like CIM_SoftwareIdentity.Name CIM_SoftwareIdentity.MajorVersion CIM_SoftwareIdentity.MinorVersion Etc.	Ditto
5	Client	Examine the returned values to determine if the FRU data was successfully retrieved.	

The administrator can also get hardware information from the DASH Service. The following steps allow him to get FRU information about the system chassis:

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Locate the packaging of the Computer by following the ComputerSystemPackage association. The Platform GUID is a property of this association.	See Physical Asset Profile
3	Client	To find if FRU information is available for the packaging, follow the ElementCapabilities of the package to its CIM_PhysicalAssetCapabilities instance. If CIM_PhysicalAssetCapabilities.FRUInfo is “TRUE”, then there is some FRU information.	Ditto
4	Client	To find FRU information, get properties like	Ditto

		CIM_PhysicalPackage.PartNumber CIM_PhysicalPackage.SeralNumber CIM_PhysicalPackage.Model etc.	
5	Client	Examine the returned values to determine if the FRU data was successfully retrieved.	

These steps allow the administrator to get FRU information about other devices:

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Follow the SystemDevice associations of the computer until the desired device is found	See Physical Asset Profile
3	Client	Follow the Realizes association of the device to locate the instance of CIM_PhysicalPackage that describes its physical aspects.	Ditto
4	Client	Follow the ElementCapabilities association of the PhysicalPackage to an instance if CIM_Capablities. If CIM_Capabilities.FRUInto is “TRUE”, then there is some FRU information	Ditto
5	Client	To find FRU information, get properties like CIM_PhysicalPackage.PartNumber CIM_PhysicalPackage.SeralNumber CIM_PhysicalPackage.Model etc.	Ditto
6	Client	Examine the returned values to determine if the FRU data was successfully retrieved.	

10.8 Detect overheat or a broken fan

An administrator who wants to be notified of events such as fan failures and high temperatures on DASH-enabled computers will use WS-Eventing support in DASH to subscribe to alert indications issued by the DASH infrastructure. After locating the target computer, the administrator creates a WS-Management Subscribe message. As illustrated in [1], the Subscribe message combines the WS-Management Subscribe Action; an End Point URI representing the computer or perhaps the CIM_NumericSensor class; and a filter that identifies the events of interest. If the administrator’s SOAP client is capable of receiving WS-Eventing messages, the target DASH implementation will send indication alerts to it as triggering events occur.

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Subscribe to threshold events using the Endpoint URI and filter as described in <DASH Wrapper Spec>.	See Sensor Profile
3	DASH Service	Watch for events that trigger the indications specified by the EPR and filter. Send matching alert indications to the SOAP end point specified in the subscription.	
4	Client	Take action as appropriate. This may be to gather more information or to shut down the computer where the event occurred.	Ditto

10.9 Query health sensors for overheat or a broken fan

To find all the fan and temperature sensors in the computer and determine which indicate a problem, an administrator uses a DASH-enabled SOAP client to find the target computer instance, then find and examine the fan sensors. If only a single sensor or a small number of known sensors are suspect, the administrator might request each of them by name.

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Locate each fan speed sensor by finding all the instances of class CIM_NumericSensor associated with the Computer through the SystemDevice association where CIM_NumericSensor.SensorType="Tachometer".	See the Sensor Profile
3	Client	To find the state of the sensor (and by implication the state of the fan), examine CIM_NumericSensor.CurrentState. If the state is not "OK", take appropriate action.	Ditto
4	Client	Examine the returned values to determine if the query operations worked.	

10.10 Detect chassis intrusion

The steps in this use case are the same as in section 10.8, except that the Administrator will watch for these alert indication messages:

MessageID	Event
1	Chassis Open
3	Drive Bay Open
6	I/O Card Area Open
8	Processor Area Open
14	Fan Area Open

10.11 Add, Remove or Edit a DASH Service User remotely.

DASH 1.0-compliant services are not required to implement user management. If they do implement user management, it is done as described in DSP1034 (Simple Identity Management Profile).

An Administrator adds an account in this way:

Step	Actor	Action	Notes
1	Client	Locate the target instance of CIM_ComputerSystem	See Base Desktop Mobile Profile
2	Client	Find an instance of CIM_AccountManagementService associated with the computer instance through CIM_HostedService. If there is no such instance, user management is not implemented.	See the Simple Identity Management Profile
3	Client	Find an instance of CIM_AccountManagementCapabilities associated with the service through CIM_ElementCapabilties	Ditto
4	Client	Examine the value of the OperationsSupported property. If it contains the value 2(Create), then adding a user is supported.	Ditto
5	Client	Create a template instance of CIM_Account	Ditto
6	Client	Execute the CreateAccount() method of the service	Ditto

To remove an account, follow the first three steps above, then

Step	Actor	Action	Notes
4	Client	Examine the value of the OperationsSupported property. If it contains the value 4 (Delete), then removing a user is supported.	See the Simple Identity Management Profile
5	Client	Execute the DeleteInstance intrinsic operation specifying the instance of CIM_Account corresponding to the user.	Ditto

11 Conclusion

DASH contains the models, mechanisms and semantics necessary to manage mobile and desktop computers in use today, independent of service state. This includes the architectural, service and operations models, and covers boot and firmware update as well as service discovery. The profiles contain the required classes, instances, properties and methods necessary to manage systems. The transport and management protocols allow implementers to determine the communication requirements for compliant systems. Discovery and security requirements described help to understand their aspects in relation to the profiles and protocols. And the use cases should help implementers understand the communications that take place in certain circumstances. All of these combine in DASH to deliver the syntax and semantics necessary to manage desktop and mobile computer systems.