



1

2

3

System Management Architecture for Server Hardware White Paper

4

5

Version 2.0.0

6

Status: Informational

7

Publication Date: 2007-08-25

8

DSP2001

9 Copyright © 2007 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

10 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
11 management and interoperability. Members and non-members may reproduce DMTF specifications and documents
12 for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be
13 revised from time to time, the particular version and release date should always be noted.

14 Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights,
15 including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard
16 as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party
17 patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights,
18 owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal
19 theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's
20 reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no
21 liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any
22 patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is
23 withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the
24 standard from any and all claims of infringement by a patent owner for such implementations.

25 For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent
26 may relate to or impact implementations of DMTF standards, visit
27 <http://www.dmtf.org/about/policies/disclosures.php>.

29
30
31
32
33
34

System Management Architecture for Server Hardware Whitepaper
Version 2.0.0
Publication Date: 2007-08-25
DSP2001
Status: Informational

35 **Abstract**

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

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

41 **Acknowledgments**

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

Table of Contents

48	Abstract	3
49	Acknowledgments.....	3
50	1 Introduction	7
51	1.1 Target Audience.....	7
52	1.2 Related Documents	7
53	1.3 Terminology	8
54	1.4 Acronyms and Abbreviations	11
55	2 Overview	12
56	2.1 Principal Goals	12
57	2.2 Service Model	12
58	2.2.1 In-Band versus Out-of-Band.....	13
59	2.2.2 In-Service versus Out-of-Service.....	13
60	2.2.3 Combined Service Model	13
61	2.3 Management Protocol Selection.....	14
62	3 Architecture Definition.....	15
63	3.1 Architectural Model	15
64	3.2 Client	17
65	3.2.1 User.....	18
66	3.2.2 Transport Client	18
67	3.3 Manageability Access Point	18
68	3.3.1 Management Service Infrastructure.....	19
69	3.3.2 Client Object Manager Adapter	19
70	3.3.3 External Authentication, Authorization, Audit Service.....	20
71	3.4 Managed System	21
72	3.4.1 Managed Element	21
73	4 Operation Model.....	22
74	4.1 MAP Responsibilities.....	22
75	4.2 Operation Handoff.....	23
76	4.3 Operation Queue	23
77	4.4 Multi-Session Capability.....	24
78	4.5 Resource Handling	24
79	5 Profiles.....	26
80	6 SM CLP Protocol Support.....	29
81	6.1 Target Addressing.....	29
82	6.1.1 Addressing Architecture	29
83	6.1.2 UFcTs and UFiTs	29
84	6.1.3 Target Addressing in the CLP	29
85	6.2 Transport Considerations.....	30
86	7 Programmatic Access Protocol Support	31
87	7.1 WS-Management Protocol	31
88	7.1.1 Overview.....	31
89	7.1.2 WS-Management – CIM Binding.....	33
90	7.2 Transport Protocol.....	33
91	7.3 Authentication Mechanisms.....	34
92	7.4 Eventing	34
93	7.4.1 Overview.....	35

94	7.4.2 Alert Indications	36
95	7.4.3 CIM Modeling of Events	36
96	7.4.4 Standardized Message Content	37
97	8 Authentication, Authorization, and Auditing.....	38
98	8.1 User Account Management	38
99	8.2 Audit.....	39
100	8.3 MAP Management	39
101	9 Discovery.....	40
102	9.1 Service and Access Point Discovery	40
103	9.2 Service Capabilities Discovery	40
104	9.2.1 SM CLP Capabilities	40
105	9.2.2 WS-Man Capabilities.....	40
106	9.3 Managed Element Discovery	41
107	10 Conclusion.....	42
108		

109 **List of Figures**

110	Figure 1 – Service Model	14
111	Figure 2 – SMASH Model.....	16
112	Figure 3 – Example MAP Implementation Architecture	17
113	Figure 4 – SMASH Protocol Stack	32
114	Figure 5 – Indication Activity Diagram.....	35
115	Figure 6 – Event Indication Subscription	36
116	Figure 7 – Standard Message Usage	37
117		

118 1 Introduction

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

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

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

134 1.1 Target Audience

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

138 1.2 Related Documents

- 139 [1] *Common Information Model (CIM) Schema*, V2.14, December, 2006,
140 www.dmtf.org/spec/cim.html.
- 141 [2] DSP0215, *SM Managed Element Addressing Specification*, V1.0.0, 2005, DMTF SMASH,
142 www.dmtf.org/standards/smash.
- 143 [3] DSP0217, *SMASH Implementation Requirements*, V1.0.0, 2006, DMTF SMASH,
144 www.dmtf.org/standards/smash.
- 145 [4] DSP0214, *Server Management Command Line Protocol Specification*, V1.0.0, 2005,
146 DMTF SMASH, www.dmtf.org/standards/smash.
- 147 [5] DSP0216, *SM CLP to CIM Mapping Specification*, V1.0.0, 2006, DMTF SMASH,
148 www.dmtf.org/standards/smash.
- 149 [6] *Posix Utility Conventions*, The Open Group Base Specifications Issue 6, IEEE Std 1003.1,
150 2004 Edition, www.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap12.html.
- 151 [7] DSP0226, *Web Services for Management (WS-Management)*, V1.0, 2006-03-14.
- 152 [8] DSP0227, *WS-Management CIM Binding Specification Preliminary*, V1.0.0b, 2006-08-09.
- 153 [9] DSP0230, *WS-CIM Mapping Specification Preliminary*, V1.0.0c, 2006-08-09.

- 154 [10] RFC2616, *Hypertext Transfer Protocol – HTTP 1.1*, IETF, June 1999.
- 155 [11] RFC2818, *HTTP over TLS 1.0*, IETF, May 2000.
- 156 [12] RFC2617, *HTTP Authentication: Basic and Digest Access Authentication*, IETF, June
157 1999.
- 158 [13] RFC2246, *The TLS Protocol, Version 1.0*, IETF, January 1999.
- 159 [14] RFC1157, *A Simple Network Management Protocol (SNMP)*, IETF, May 1990.
- 160 [15] DSP1054, *Indications Profile, V1.0*.
- 161 [16] DSP1033, *DMTF Profile Registration Profile, V1.0*.
- 162 [17] RFC4301, *Security Architecture for the Internet Protocol*, IETF, December 2005.
- 163 [18] RFC4303, *IP Encapsulating Security Payload (ESP)*, IETF, December 2005.
- 164 [19] RFC4305, *Cryptographic Algorithm Implementation Requirements for Encapsulating
165 Security Payload (ESP) and Authentication Header (AH)*, IETF, December 2005.
- 166 [20] CIM Schema, Version 2.15.0.
- 167 [21] Web Services Architecture, W3C Working Group, Note 11, February 2004.
- 168 [22] RFC2609, *Service Location Protocol, Version 2*, IETF, June 1999.

169 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
SM CLP	Server Management Command Line Protocol
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
TCP/IP	See TCP <i>and</i> 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

171 **2 Overview**

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

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

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

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

198 **2.1 Principal Goals**

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

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

208 **2.2 Service Model**

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

213 **2.2.1 In-Band versus Out-of-Band**

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

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

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

224 **2.2.2 In-Service versus Out-of-Service**

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

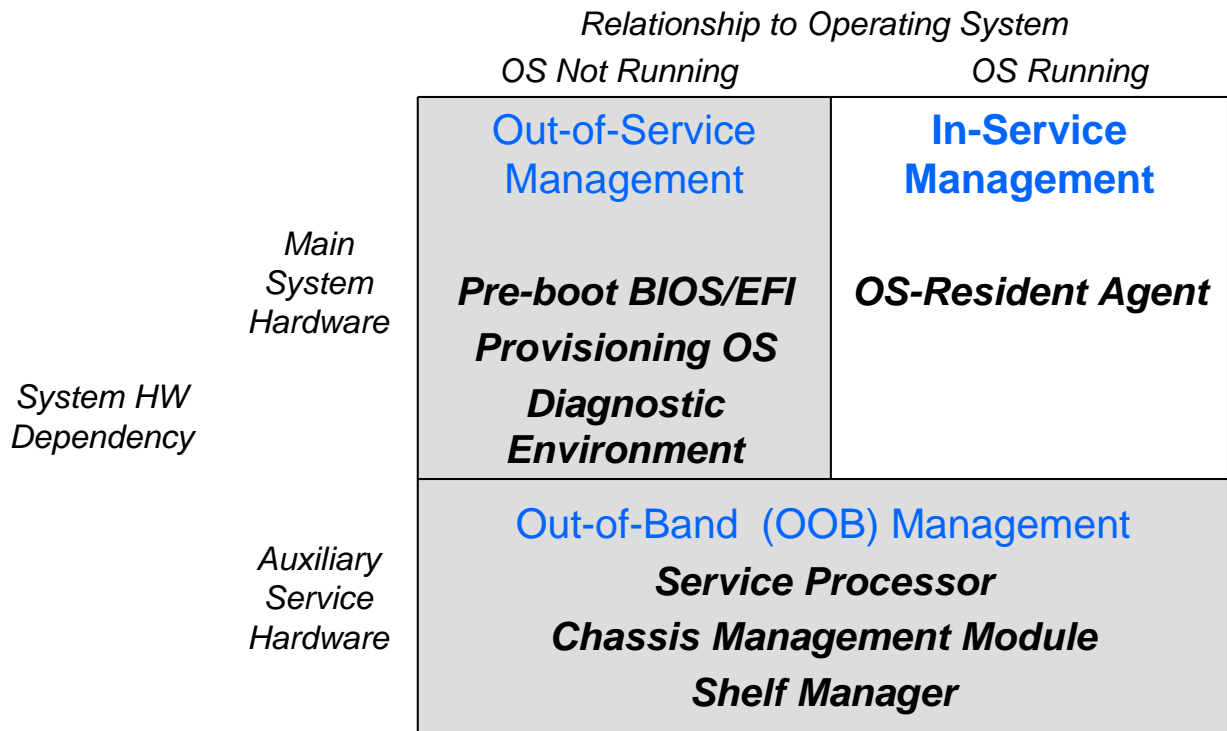
227 In-Service management operates with the support of software components that run concurrently
228 and are dependent on the operating system. This is often provided through a service or process
229 within the operating system.

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

233 **2.2.3 Combined Service Model**

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

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



245

246

Figure 1 – Service Model

247 **2.3 Management Protocol Selection**

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

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

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

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

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

267 **3 Architecture Definition**

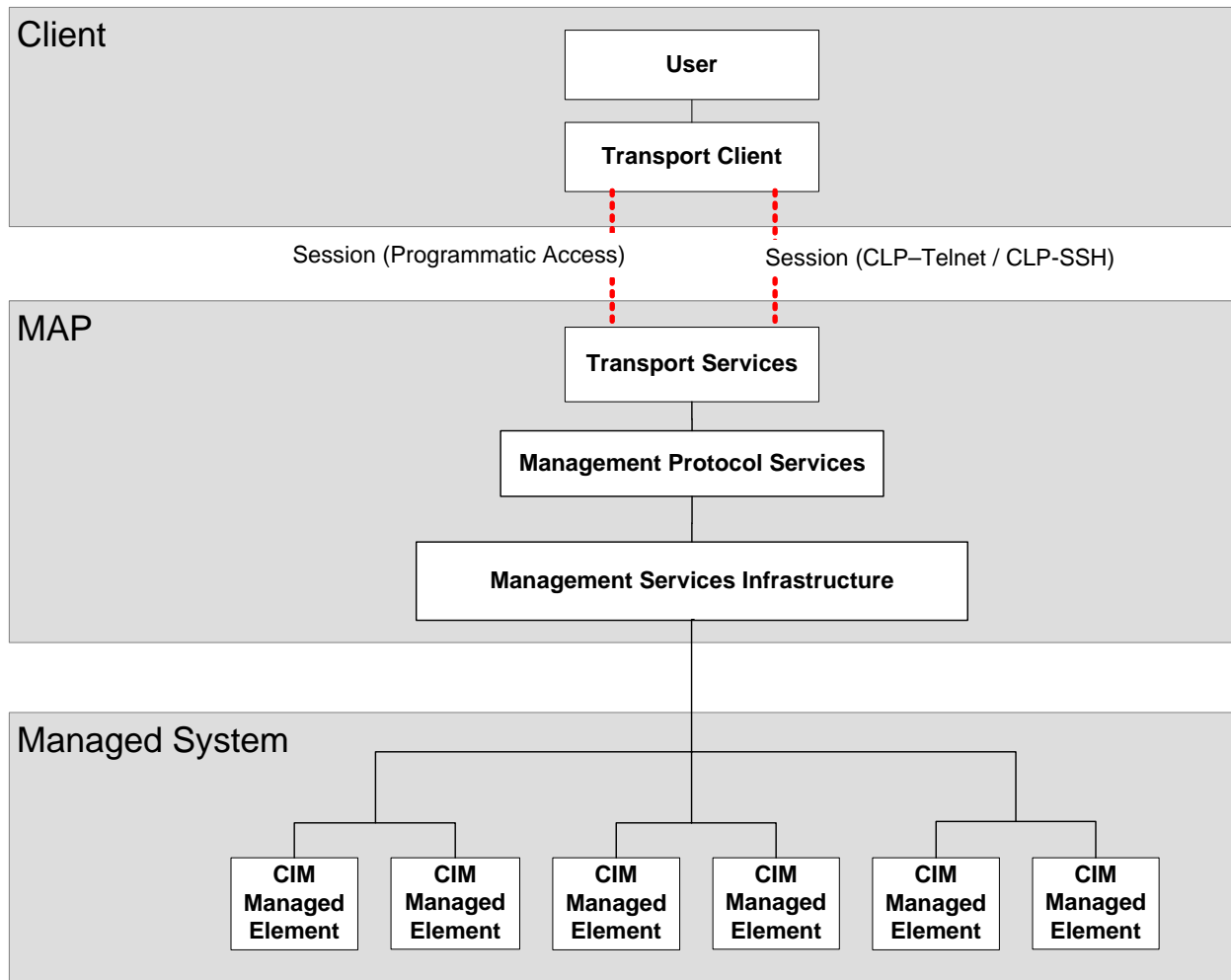
268 To provide server management standardization, it is necessary to develop an abstract model that
269 describes server management regardless of the actual implementation. This abstract model is
270 necessary to provide a common vocabulary and a common base of understanding. It is also used
271 to illustrate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

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

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

280 **3.1 Architectural Model**

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

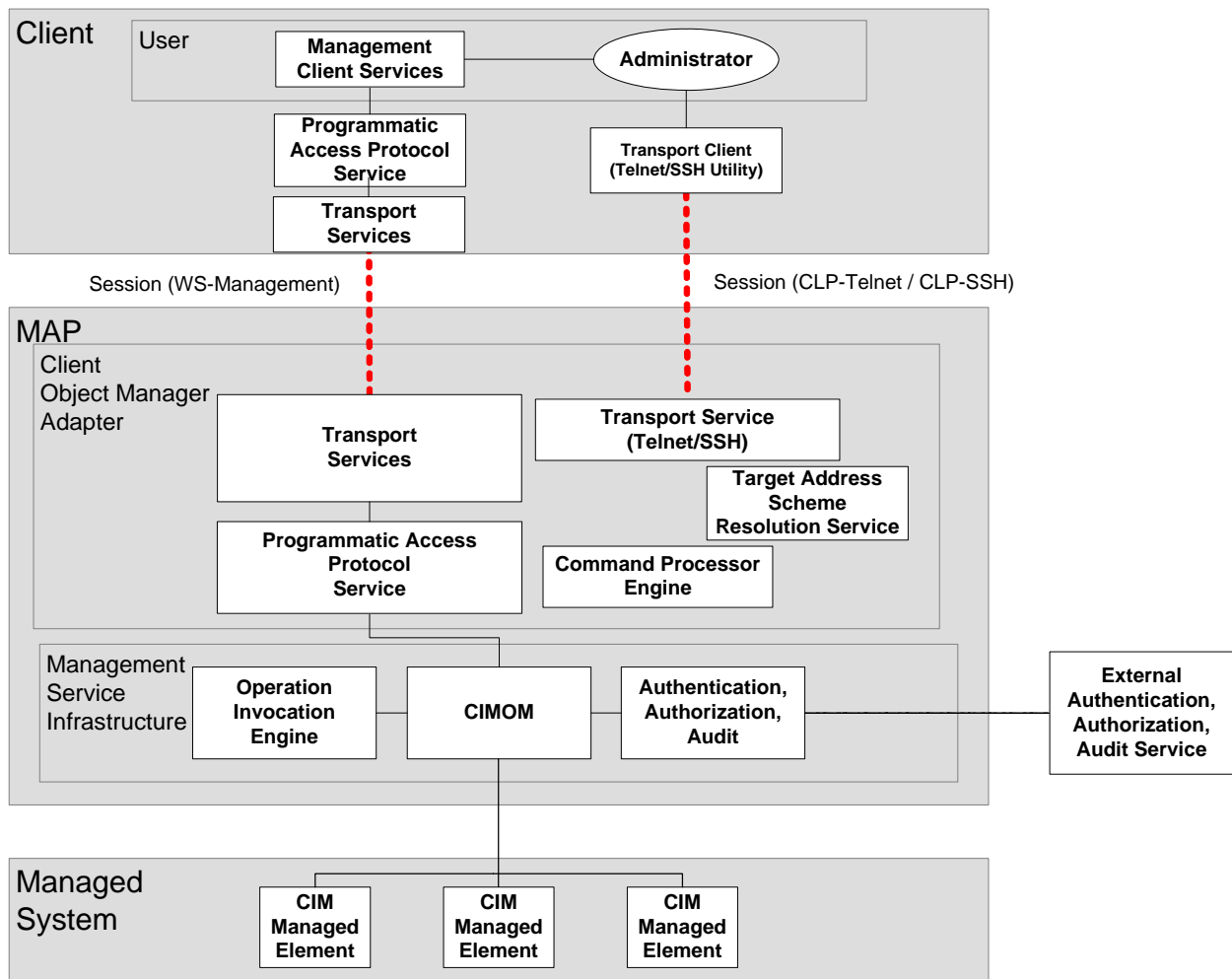


289

290

Figure 2 – SMASH Model

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



304

305

Figure 3 – Example MAP Implementation Architecture

306 **3.2 Client**

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

309 A Client is responsible for:

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

317 **3.2.1 User**

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

324 **3.2.1.1 Management Client**

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

330 **3.2.1.2 Administrator**

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

334 **3.2.2 Transport Client**

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

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

343 **3.3 Manageability Access Point**

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

347 The MAP is responsible for the following tasks:

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

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

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

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

363 **3.3.1 Management Service Infrastructure**

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

367 **3.3.1.1 CIMOM**

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

372 **3.3.1.2 Operation Invocation Engine**

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

378 **3.3.1.3 Authentication, Authorization, Audit**

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

385 **3.3.2 Client Object Manager Adapter**

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

392 **3.3.2.1 Transport Service**

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

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

400 **3.3.2.2 CLP Service**

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

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

406 **3.3.2.3 Management Protocol Service**

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

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

415 **3.3.2.4 Command Processor Engine**

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

419 **3.3.2.5 Target Address Scheme Resolution Service**

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

425 **3.3.3 External Authentication, Authorization, Audit Service**

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

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

434 **3.4 Managed System**

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

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

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

444 **3.4.1 Managed Element**

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

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

448 **4 Operation Model**

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

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

456 **4.1 MAP Responsibilities**

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

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

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

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

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

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

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

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

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

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

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

491 **4.2 Operation Handoff**

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

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

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

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

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

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

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

520 **4.3 Operation Queue**

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

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

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

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

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

541 **4.4 Multi-Session Capability**

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

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

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

559 **4.5 Resource Handling**

560 The SMASH contains mechanisms that enable resource handling.

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

564 The administration and configuration of complex systems, such as those with shared resources,
565 often requires the locking of an ME to manage the ME or to ensure that the ME is assigned to
566 one and only one system. Direct support of these mechanisms is not included in this version of

567 the architecture. Because direct support is not required, the mechanism for handling resource
568 locking is outside the scope of this specification.

569 5 Profiles

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

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

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

- 584 • [DSP1004](#), the *Base Server Profile* is a top-level profile that provides the ability to
585 manage server systems.
- 586 • [DSP1030](#), the *Battery Profile* provides the ability to manage batteries of a managed
587 system.
- 588 • [DSP1012](#), the *Boot Control Profile* provides the ability to manage boot configurations of
589 a system.
- 590 • [DSP1018](#), the *Service Processor Profile* provides the ability to manage service
591 processors, chassis managers, and other dedicated management controllers.
- 592 • [DSP1005](#), the *CLP Service Profile* provides the ability to manage an implementation of
593 the SM CLP architecture.
- 594 • [DSP1022](#), the *CPU Profile* provides inventory, status, and state information for
595 processors of a managed system.
- 596 • [DSP1019](#), the *Device Tray Profile* provides the ability to manage shared media trays in a
597 modular system.
- 598 • [DSP1037](#), the *DHCP Client Profile* provides the ability to manage the DHCP client
599 configuration of a managed system.
- 600 • [DSP1038](#), the *DNS Client Profile* provides the ability to manage the DNS client
601 configuration of a managed system.
- 602 • [DSP1014](#), the *Ethernet Port Profile* provides inventory, status, and state information for
603 the Ethernet interfaces of a managed system.
- 604 • [DSP1013](#), the *Fan Profile* provides inventory, status, and state information for fans of a
605 managed system.
- 606 • [DSP1054](#), the *Indications Profile* provides the ability for a client to subscribe to
607 indications produced by a MAP.
- 608 • [DSP1074](#), the *Indicator LED Profile* provides the ability to manage the LEDs of a
609 managed system.

- 610 • [DSP1036](#), the *IP Interface Profile* provides the ability to manage the configuration of IP
611 interfaces of a managed system.
- 612 • [DSP1076](#), the *KVM Redirection Profile* provides the ability to configure the KVM
613 redirection infrastructure of a managed system.
- 614 • [DSP1008](#), the *Modular System Profile* provides the ability to manage modular enclosures
615 and contained components.
- 616 • [DSP1029](#), the *OS Status Profile* provides the ability to determine basic status information
617 about installed and running operating systems.
- 618 • [DSP1020](#), the *Pass-Through Module Profile* provides inventory, status, and state
619 information for pass-through modules of a managed system.
- 620 • [DSP1075](#), the *PCI Device Profile* provides inventory, status, and state information about
621 PCI devices in a managed system.
- 622 • [DSP1011](#), the *Physical Asset Profile* provides the ability to report physical asset
623 information including capacity and FRU information for components installed in a
624 managed system.
- 625 • [DSP1027](#), the *Power State Management Profile* provides the ability to query and manage
626 the power state on a managed system.
- 627 • [DSP1015](#), the *Power Supply Profile* provides inventory, status, and state information for
628 power supplies of a managed system.
- 629 • [DSP1010](#), the *Record Log Profile* provides the ability to retrieve error and event log
630 information for managed systems.
- 631 • [DSP1039](#), the *Role Based Authorization Profile* provides the ability to manage rights
632 granted to security principals through role membership.
- 633 • [DSP1009](#), the *Sensors Profile* provides the ability to query sensor status and state
634 information for component and system sensors.
- 635 • [DSP1021](#), the *Shared Device Management Profile* provides the ability to control access
636 to shared devices in a modular system.
- 637 • [DSP1034](#), the *Simple Identity Management Profile* provides support for basic account
638 management, including account creation and deletion.
- 639 • [DSP1007](#), the *SM CLP Admin Domain Profile* is used to model the administrative domain
640 of an SM CLP implementation.
- 641 • [DSP1006](#), the *SMASH Collections Profile* provides support for collecting settings,
642 capabilities, and other Managed Elements to simplify management access through an SM
643 CLP implementation.
- 644 • [DSP1023](#), the *Software Inventory Profile* provides the ability to view the firmware,
645 device drivers, BIOS, and other software installed on a system and its components. It also
646 provides the ability to view the software available for installation on a system and its
647 components.
- 648 • [DSP1025](#), the *Software Update Profile* provides the ability to perform software
649 installation, upgrades, and downgrades on a system and its components.

- 650
- 651
- [DSP1017](#), the *SSH Service Profile* provides the ability to manage the configuration of an SSH service and client sessions.
- 652
- 653
- [DSP1026](#), the *System Memory Profile* provides inventory, status, and state information for the main system memory of a managed system.
- 654
- 655
- [DSP1016](#), the *Telnet Service Profile* provides the ability to manage the configuration of a Telnet service and client sessions.
- 656
- 657
- [DSP1024](#), the *Text Console Redirection Profile* provides the ability to start and stop text console redirection over the interfaces of a managed system.
- 658
- 659
- [DSP1040](#), the *Watchdog Profile* provides the ability to manage watchdog timers of a managed system.
- 660

661 **6 SM CLP Protocol Support**

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

663 **6.1 Target Addressing**

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

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

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

672 **6.1.1 Addressing Architecture**

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

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

682 **6.1.2 UFcTs and UFiTs**

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

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

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

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

694 **6.1.3 Target Addressing in the CLP**

695 The Server Management Command Line Protocol will accept UFiTs that are formed into a UFiP.
696 The SM CLP also accepts other target address constructs, such as those used to select all
697 instances of a class. MAPs support a number of standard, default UFiTs that are consistent with
698 the SMASH addressing rules contained in the *SM Managed Element Addressing Specification*
699 [2].

700 **6.2 Transport Considerations**

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

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

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

726

727 **7 Programmatic Access Protocol Support**

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

734 **7.1 WS-Management Protocol**

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

736 **7.1.1 Overview**

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

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

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

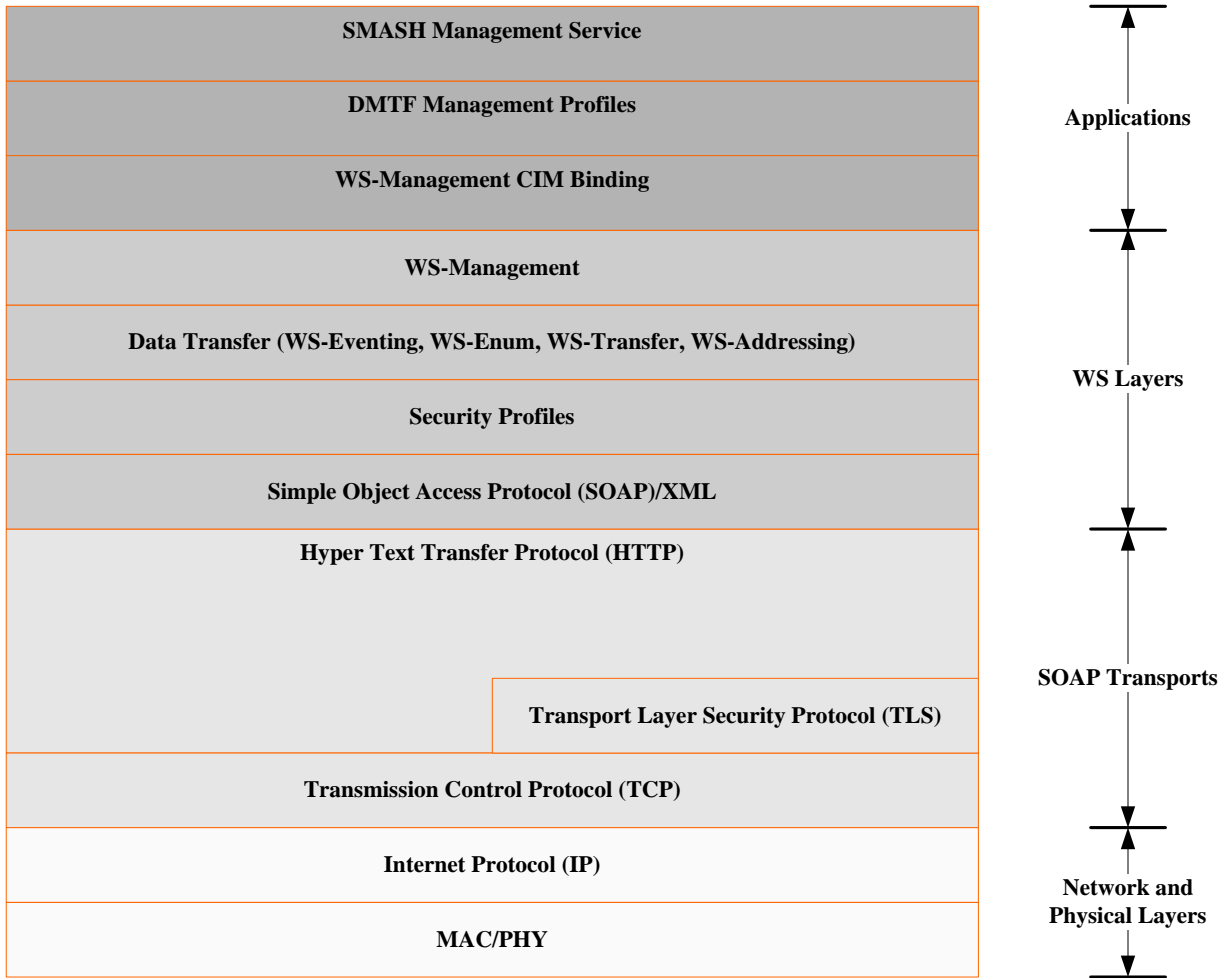
749 The transport layers that carry SOAP messages are next in the stack. These layers include

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

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

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

762



763

764

Figure 4 – SMASH Protocol Stack

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

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

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

780 consumer can then request XML element information over a span of one or more SOAP
781 messages using the enumeration context. The enumeration context is represented as
782 XML data. The following operations (defined as SOAP request/response messages) are
783 supported using this model¹:

- 784 a) *Enumerate*: is used to initiate an enumeration and receive an enumeration context.
785 b) *Pull*: is used to pull a sequence of elements of a resource.
786 c) *Release*: is used to release an enumeration context gracefully.
- 787 3. WS-Eventing: is a SOAP-based protocol for one Web service to register interest and
788 receive messages about events from another Web service. The operations supported by
789 WS-Eventing include *Subscribe*, *Renew*, *GetStatus*, *Unsubscribe*, and *SubscriptionEnd*.
790 WS-management defines heartbeats as pseudo-events. WS-Management also defines a
791 bookmark mechanism for keeping a pointer to a location in the logical event stream.
792 The delivery modes defined for events are: Push, Push with Acknowledgement
793 (PushWithAck), Batched, and Pull.

794 7.1.2 WS-Management – CIM Binding

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

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

804 7.2 Transport Protocol

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

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

815 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.

816 **7.3 Authentication Mechanisms**

817 The three types of authentication mechanisms considered are as follows:

818 1. **Machine-level authentication**

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

823 2. **User-level authentication**

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

829 3. **Third-party authentication**

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

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

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

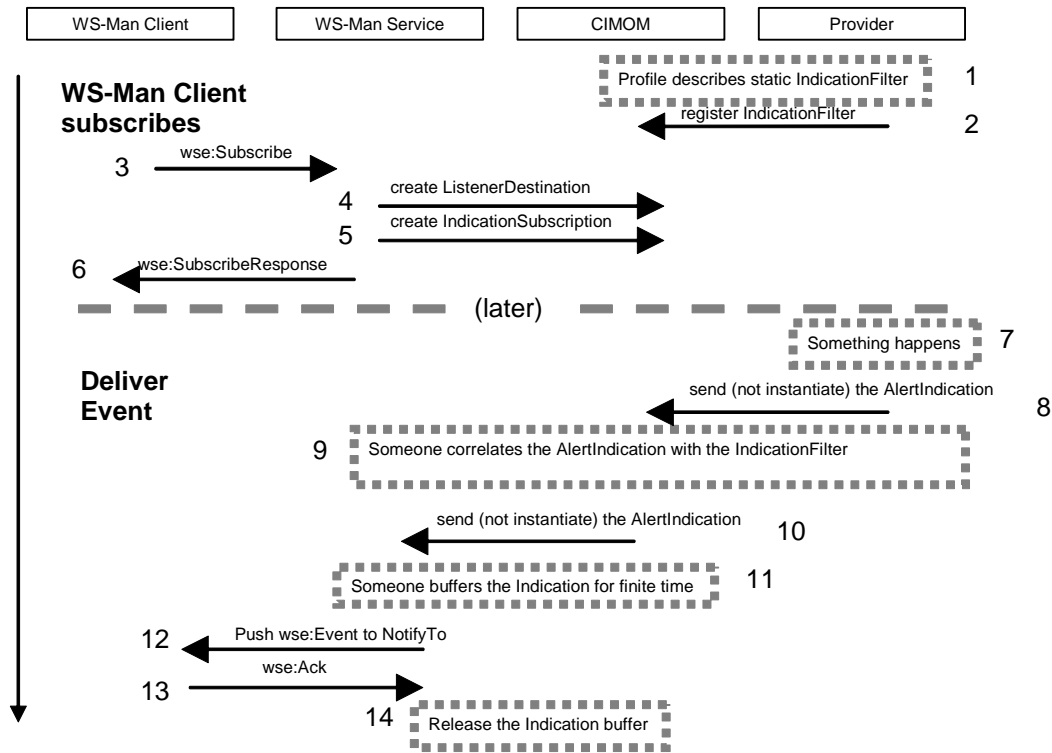
843 **7.4 Eventing**

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

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

856 **7.4.1 Overview**

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



863

864

Figure 5 – Indication Activity Diagram

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

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

875 When an event occurs (step 7), the instrumentation has the responsibility of communicating the
 876 event to applications that have subscribed to that particular information. The WS-Eventing
 877 approach to communicating event information involves generating an instance of the appropriate
 878 CIM Indication Class and sending the instance information, along with other information, as the
 879 payload of an event delivery message to subscribing listeners. Specifics of the CIM-to-event

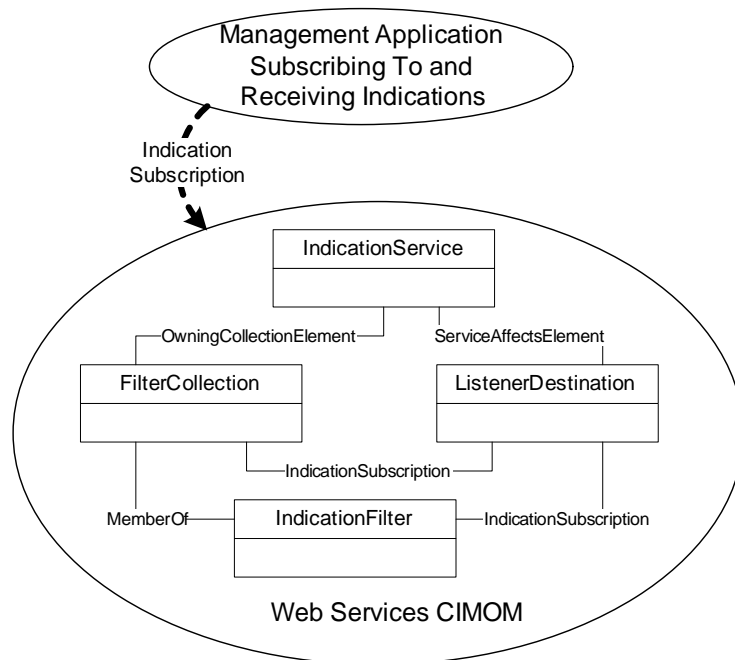
880 delivery message mapping are defined in the *WS-Management CIM Binding Specification* [8]. A
 881 synopsis of that process is as follows: when an event occurs (step 7), the provider sends the
 882 AlertIndication to the CIMOM (step 8). Then one of the implementation components correlates
 883 the AlertIndication from the provider with the IndicationFilter from the Client (step 9). Then the
 884 CIMOM sends the AlertIndication to the WS-Management Service (step 10). The service then
 885 pushes the Event (step 12) to the Client, which acknowledges the message (step 13), resulting in
 886 the Indication buffer being released (step 14). Note that the instance of the indication will be
 887 buffered for a finite amount of time by the MAP, implying that the Client should acknowledge
 888 the receipt of the message in an expedient fashion.

889 7.4.2 Alert Indications

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

897 7.4.3 CIM Modeling of Events

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



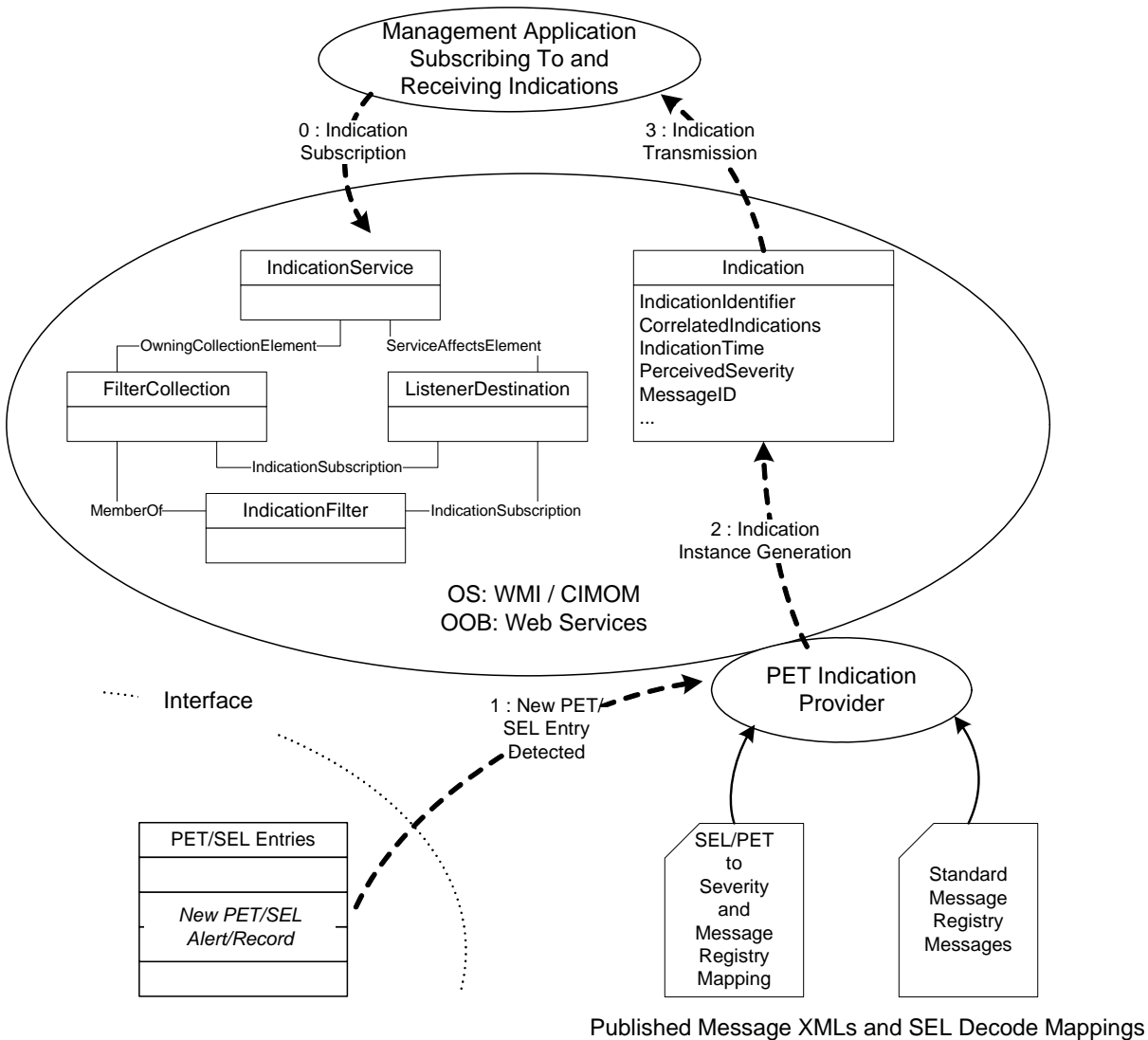
903

904

Figure 6 – Event Indication Subscription

905 **7.4.4 Standardized Message Content**

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



914

915

Figure 7 – Standard Message Usage

916 **8 Authentication, Authorization, and Auditing**

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

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

926 **8.1 User Account Management**

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

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

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

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

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

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

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

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

958 **8.2 Audit**

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

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

966 **8.3 MAP Management**

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

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

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

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

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

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

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

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

991 **9 Discovery**

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

996 **9.1 Service and Access Point Discovery**

997 The Service Location Protocol (SLP) is an industry-standard protocol used to advertise the
998 availability of services on a network. The DMTF has defined an SLP template for WBEM. This
999 template is being updated to include advertisement of WS-Man and SM CLP. A MAP that
1000 implements SMASH may support SLP to advertise the services it supports and the associated
1001 access URIs.

1002 **9.2 Service Capabilities Discovery**

1003 After discovering a MAP and its services, the second task is to discover the capabilities of the
1004 MAP itself. This section details the different approaches available for the SM CLP and WS-Man
1005 protocols.

1006 **9.2.1 SM CLP Capabilities**

1007 For the SM CLP, SMASH handles discovery of service capabilities by modeling it in profiles.
1008 The *SM CLP Profile* contains the classes describing the SM CLP services available within the
1009 MAP. To discover the SM CLP capabilities of the MAP is to simply discover the properties and
1010 methods, as well as the service access points and transports for the SM CLP. The *SM CLP*
1011 *Specification* and other specifications indicate how to query and alter the values of the properties
1012 for the services within the MAP.

1013 **9.2.2 WS-Man Capabilities**

1014 For WS-Man, a client can use the Identify method to query the capabilities. The Identify method
1015 is defined in WS-Management [7]. A management client can subsequently send the Identify
1016 message to the TCP port in use by WS-Man to learn the protocol version, the product vendor,
1017 and product version of the service. These are provided in the IdentifyResponse message in the
1018 wsmid:ProtocolVersion, wsmid:ProductVendor, and wsmid:ProductVersion elements,
1019 respectively. Note that the TCP port in use is either known *a priori* or extracted from the SLP
1020 response.

1021 A SMASH MAP supports the Identify method on each registered access port that it supports. See
1022 *SMASH Implementation Requirements Specification* [3] for the complete list of registered ports.

1023 SMASH defines extension elements as children of the IdentifyResponse element in addition to
1024 the child element defined in WS-Management [7]. For details of these elements, see *SMASH*
1025 *Implementation Requirements Specification* [3].

1026 **9.3 Managed Element Discovery**

1027 The final aspect of discovery is how a Client discovers which Managed Elements are managed
1028 by a particular MAP. Fortunately, this is a capability that exists in the protocols in use today. SM
1029 CLP and WS-Management provide operations to determine the profiles and Managed Elements
1030 within the management domain of the MAP. These operations are well documented in their
1031 individual specifications.

1032 **10 Conclusion**

1033 SMASH is one component in a suite of specifications that delivers the architecture, addressing
1034 methodology, profiles, Command Line Protocol, and discovery mechanisms necessary to manage
1035 the full range of current and emerging servers in enterprise environments.

1036 The SMASH contains the models, mechanisms, and semantics for managing servers in the data
1037 center, regardless of service state. This includes the architectural, service, and operations models,
1038 and covers boot and firmware updates as well as service discovery. The profiles contain the
1039 required classes, instances, properties, and methods necessary to manage systems. The
1040 combination of the profiles with the addressing methodology determines the format of the target
1041 addressing convention for compliant systems. Together, they deliver the syntax and semantics
1042 necessary to manage servers.

1043