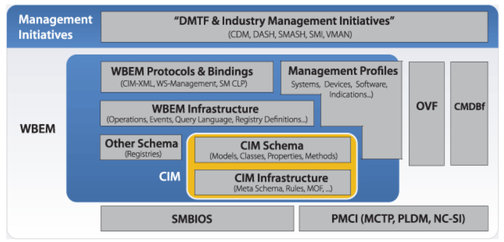
SMASH & DASH Technology Overview

Jeff Hilland & members of the Server Desktop & Mobile Platforms WG
sdmpwg@dmtf.org

DMTF Management Initiatives

SMASH & DASH are two of the four DMTF Management Initiatives
- SMASH – Systems Management Architecture for Server Hardware
- DASH – Desktop and mobile Architecture for System Hardware
- CDM – Common Diagnostics Model
- VMAN – Virtualization Management

DMTF Technology Diagram



What is SMASH?

SMASH Stands for Systems Management Architecture for Server Hardware
- SMASH is a suite of specifications that deliver industry standard protocols and profiles to unify the management of the data center.
  - Vendor independent
  - Platform neutral
  - Independent of machine state

The SMASH specifications utilize the CIM data model and industry standard transports and security mechanisms.
- Align out-of-service with in-service manageability.
- Align in-band with out-of-band manageability.
- Customer Driven

SMASH consists of:
- Architecture White Paper
- SM CLP
- WS-Management
- SMASH Implementation Requirements Specification
- Profiles.

1.0 Standard completed Dec, 2006
- www.dmtf.org/standards/smash

Made public at Manageability Developers Conference

2.0 Standard completed Sep 2007
- Made public at Intel Developers Forum

What is DASH?

DASH Stands for Desktop and mobile Architecture for System Hardware
- Web services based programmatic interface for desktop to mobile environment, including bladed PCs.
- Utilizes the CIM Data Model, leveraging the SMASH Profiles & Architecture gives this effort a head start.
- Tackling tough issues like standardized Eventing.
- First revision maps to ASF functionality plus inventory and account management.

DASH consists of:
- Architecture White Paper
- WS-Management
- DASH Implementation Requirements Specification
- Profiles.

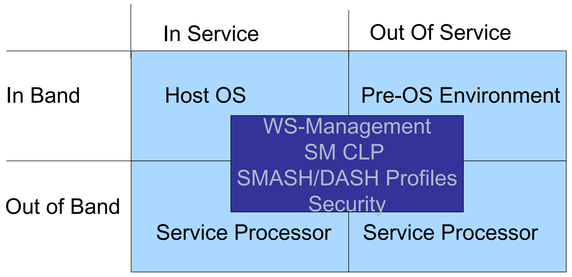
1.0 completed Apr, 2007
- www.dmtf.org/standards/dash

Made public at Microsoft Management Summit (MMS), 2007

1.1 completed Dec 2007
- Made public at MDC 2007.

Platform Manageability Alignment

DMTF is driving a consistent interface and view, regardless of machine state or access method.



Industry is aligning around key elements:
- Protocols (Transport) – WS-Management & SM CLP
- Profiles (Data Model) – SMASH, DASH & SMI-S Profiles

Management data from the profiles is wrapped in the protocols using standard encoding schemes to achieve interoperability

SMASH & DASH Architectural Models

Management Models included in white papers
- In-Band/In-Service, Out-of-band/Out-of-Service Model
  - SMASH & DASH Similar
- Manageability Access Point Model
  - Common transport/protocol is WS-Man.
    SMASH also has SM CLP.
- Operational Model
  - Job oriented for certain functions (implementation dependent)
- Session Capabilities
  - Concurrent session support (implementation dependent)
- Resource Handling
  - Transient nature of resources
- Security Model
  - Consistent across implementations

WS Management

WS Management is a SOAP based protocol designed as a programmatic interface

Web Services based set of specs that map SOAP to the CIM Schema

DISCOVER the presence of management resources and navigate between them

GET, PUT, CREATE, and DELETE individual management resources, such as settings and dynamic values

ENUMERATE the contents of containers and collections, such as large tables and Logs

SUBSCRIBE to events emitted by managed resources

EXECUTE specific management methods with strongly typed input and output parameters

SM CLP

SM CLP (Server Management Command Line Protocol) is
Designed for a human (primary) or a script (secondary)
- Working over, but not limited to, industry standard transports
  - Telnet & SSHv2
- Exposes CIM data model in a "human friendly" fashion through simple commands
  - SM ME Addressing Spec turns CIM containment into command targets like "system1\fan1"

NOT a full featured programming interface
- Because it is a lightweight communication mechanism with some semantics were intentionally left out.
- Therefore, a programmatic interface is still required for some operations
- But input and output are fully machine-parsable.

BUT all of the Hardware Operations (provisioning, allocation, configuration, inventory, state change, security) can be done with the CLP.
Either by a human, script or program
- Because there is a grammar that defines input and XSD defined output.
- Very light weight implementations can be done.

Profiles

A profile is a specified subset of the CIM Schema elements that describe a standard implementation for interoperability and conformance verification
- The CIM Specification defines the language and methodology for describing management data. Schemas provides the actual model descriptions.

A profile contains
- Required and Conditional CIM Element Properties and Methods
- Class & Instance Diagrams
- Profile Usage Guide and Profile Registration Profile compliance

DMTF is producing Profiles
- Strong desire to have common set of profiles to extent possible
- Synergy with SMASH and SMI efforts
- Definition of optional elements to support scaling from desktop and mobile platforms up to stand-alone, modular and partitionable servers

SMASH Profiles

High-level Profiles
1. CLP Service
2. Base Server
3. Modular System
4. Service Processor
5. Physical Asset
6. Boot Control
7. SM CLP Admin Domain
8. SMASH Collection
9. CPU
10. System Memory
11. Fan
12. LED
13. Power Supply
14. Power State Management
15. Record Log
16. Sensor
17. Watchdog
18. Host Hardware Raid (Reference)
19. OS Status
20. PCI Device
21. Software Update
22. Software Inventory
23. Host LAN Network Port
24. IP Interface
25. Ethernet Port
26. DHCP Client
27. DNS Client
28. SSH Service
29. Telnet Service
30. Role-Based Authorization
31. Simple Identity Management
32. Shared Device Management
33. Pass-Through Module
34. Device Tray
35. Text Console Redirection
36. KVM Redirection
37. Profile Registration
38. Computer System
39. Indications

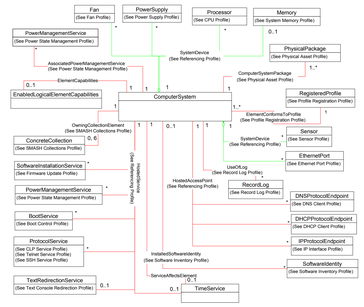
* Underlining indicates SMASH 2.0 Profile, Blue is autonomous profile

DASH Profiles

High-level Profile
1. Base Desktop
2. Physical Asset
3. Boot Control
4. CPU
5. System Memory
6. Fan
7. Power Supply
8. Power State Management
9. Sensor
10. Battery
11. BIOS Management
12. Opaque Data
13. OS Status
14. Software Update
15. Software Inventory
16. Host LAN Network Port
17. IP Interface
18. Ethernet Port
19. DHCP Client
20. DNS Client
21. Role-Based Authorization
22. Simple Identity Management
23. Text Console Redirection
24. KVM Redirection
25. Media Redirection
26. USB Redirection
27. Profile Registration
28. Computer System
29. Indications
30. WiFi

* Underlining is DASH 1.1 profile, Blue is autonomous profile

Base Server Profile



Connectivity

Embedded DASH implementations operate on well known TCP Ports (repurposed ASF TCP Ports)
- One for HTTP
  - Could be configured to support WS-Discovery only
- One for HTTPS
  - It is recommended to support both TLS Security Profiles (see next page)

Embedded SMASH implementations do not use the repurposed ASF ports and use the normal HTTP ports

Discovery

When discussing discovery it is important to divide the discussion into 3 broad groups, namely
- Network Addressable End-Point Discovery
- Classification (Type Discovery)
- Service Discovery

These broad groups can be further broken down with each layer of discovery providing more information including:
- The existence of the Network Addressable End-Point.
- The type of device (classification)
- The services (capabilities) of the device as a whole
- The device in the context of topology (e.g. a MAP in a Client Machine)

WS-Identify is supported as part of the standard.

Security

HTTP 1.1 is the required transport
Two classes of SMASH & DASH defined WS-Management security levels: (See next slide for class details)
- DMWG Class A – HTTP Only: Digest/Basic Auth
- DMWG Class B – HTTPS or IPSec: TLS1, TLS2 or IPSec

A SMASH or DASH implementation must be compliant with at least one of Security Class levels
A SMASH or DASH implementation should be Class B compliant for privacy/confidentiality and additional security

---

Three roles are defined for DASH & SMASH:
- Administrator – Mandatory for SMASH & DASH
- Operator – Optional for SMASH & DASH
- Read Only – Mandatory for SMASH, Optional for DASH

Indications

Two major categories of Indications for the CIM Model
- Alert Indications
  - Message ID/string oriented class design
  - The underlying event and its data may or may not be modeled in the CIM class hierarchy
  - Includes handles pointing to the alerting Managed Element
- Lifecycle Indications
  - Generated based on changes in instantiated objects
  - Indication class includes the object instances and handles pointing to the objects
  - For changes in existing objects, the indication class also include the object instance before the change
  - Predominant approach used by SNIA

SMASH & DASH Support Alert Indications
- Platform Event Message Registry – DSP8007
  - Standardized message ID's and message strings
  - Publish recommended "Perceived Severity" mappings
- Included in DASH 1.0 and SMASH 2.0 Specifications
- Published Recommended Message Registry Mappings
  - Recommended PET Frame Values Mapping Spec

Next Steps

Updates to SMASH & DASH Forthcoming
- Improved Discovery
  - SLP template reference included in specs
  - Support for CIM_RegisteredSpecification class
- View Class Support
- Endpoint correlation