Storage Management Technical Specification, Simple Storage Management

Version 0, Revision 25

Abstract: The Simple Storage Management model—SSM for short—is designed to make SMI-S more efficient and easy to use, especially for entry-level work.

Publication of this Working Draft for review and comment has been approved by the SMI Technical Steering Group (TSG). This draft represents a “best effort” attempt by the TSG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a “work in progress.” Suggestion for revision should be directed to http://www.snia.org/feedback/.

Working Draft

May 21, 2015

Foreword

Complexity often seems like a lurking and somehow unavoidable evil, growing like a cancer in the solutions to complicated problems. Software systems are particularly susceptible to this. Large portions of modern systems are so complicated that no human fully comprehends them, and as patch is laid on top of patch, they become so fragile that fixing problems can cost more than tolerating them.

But hope springs eternal, and inspired by this, the SNIA Storage Management Initiative (SMI) membership has undertaken to find a simpler way.

Fools ignore complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it.
-- Alan Perlis

When designing systems, aspire to brilliance. Eschew cleverness.
-- Alan Yoder

Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.
-- Antoine de Saint-Exupéry

The Simple Storage Management model—SSM for short—is designed to make SMI-S more efficient and easy to use, especially for entry-level work. No existing work is invalidated by this work. Rather, it provides a limited number of "flattened" classes that can be used for basic storage management. Our goal is to allow new clients and providers to satisfy all their storage management needs using this new model. But if and when more is needed, a mechanism is provided to enable clients to navigate to the larger CIM schema structure of SMI-S that has been built over time.

Special Thanks

The SNIA wishes to thank Huawei Technologies and WBEM Solutions Inc. for exceptional commitment to and support of this work. Many other companies have provided input and help as well; they are gratefully called out in the Acknowledgements section.

Revision History

Date	Revision	TRAC tickets incorporated and other changes
23 Nov 2014	initial	Subversion source code repository established
15 Dec 2014	1.0 Draft 1	
16 Jan 2015	0.18	Tickets # 26, 3, 2, 7, 8, 31, 20, 21, 29, 41, 42, 30, 28, 13, 9, 14, 24, 35, 11, 10, 1, 12, 4, 5 Document generating tool mof2html completed Word macros for document generation completed
28 Jan 2015	0.19	Fix diagrams messed up by a formatting change
6 Feb 2015	0.20	Misc. editing
2 Mar 2015	0.21	Edited to comply with ISO guidelines, added new SNIA front matter
30 Mar 2015	0.22	Make authorization component of Security section optional, remove all features besides blocks and physical system, in preparation for 1.0 release, correct out-of-date material.
30 Mar 2015	0.23	Accept all changes to get a baseline for future revisions.
15 Apr 2015	0.24	Editorial pass, update MOF listing, remove comments and hyperlinks.
21 May 2015	0.25	Extensive rework of infrastructure classes and corresponding text.

Intended Audience

This document is intended for use by individuals and companies engaged in developing, deploying, and promoting interoperable multi-vendor storage systems and networks through the Storage Networking Industry Association (SNIA) organization.

Usage

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Portions of the CIM Schema are used in this document with the permission of the Distributed Management Task Force (DMTF). The CIM classes that are documented have been developed and reviewed by both the SNIA and DMTF Technical Working Groups. However, the schema is still in development and review in the DMTF Working Groups and Technical Committee, and subject to change.

About the Storage Networking Industry Association (SNIA)

SNIA Web Site

Current SNIA practice is to make updates and other information available through the SNIA web site at

http://www.snia.org

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at

http://www.snia.org/feedback/.

Acknowledgements

The SNIA SMI Technical Steering Group, which developed and reviewed this specification, would like to recognize the significant contributions made by the following members:

Organization Represented	Name of Representative
EMC Corporation	Tony Fiorentino
	Mike Hadavi
	Peter Lamanna
	George Ericson
Hitachi Data Systems	Steve Quinn
Huawei Technologies	Alan Yoder
Individual Contributor	Mike Walker
Inova	Karl Schopmeyer
Intellimagic	Wim Oudshoorn
	Brett Allison
	Tom Koelman
	Daan Smit
Microsoft	Jeff Goldner
	Hector Linares
NetApp	Don Deel
Red Hat	Tony Asleson
SolarWinds	Chandra Reddy
WBEM Solutions, Inc. (WS)	Jim Davis

Contents

List of Figures

List of Tables

1. Introduction

A number of criticisms have been leveled at SMI-S over the years. Some examples are

- Excessive complexity
- Specification bloat
- A class hierarchy ("propagation graph") that is too deep
- Too many associations to traverse
- A long steep learning curve
- Domination of spec development by server vendors
- Focus on what equipment can do, rather than on what IT needs

The increasing adoption of SMI-S by Microsoft[1] has helped at least to address the issue of server vendor-centricity, but other problems remain. Microsoft has published a document set outlining the CIM classes and associations that it requires—and their correct use—for successful management by their Server 2012 storage management console. Overall, roughly 85 classes and 40 associations are required, with several hundred pages of profile specification text to back them up. This is a tiny fraction of the number of classes in the CIM schema, and about a third of the SMI-S spec. Yet, less than 20 discrete physical and logical entities are discovered and managed by the Microsoft client software. A naïve person—trained in Computer Science but not steeped in the metaphysics of CIM—may wonder why so many classes and associations are required to manage so few system component types. This book attempts to address this issue.

[1] All trademarks herein are the property of their respective owners.

2. Terms, normative references and use cases

2.1. Terms

This specification uses terms from Clause 3 of the SMI-S Architecture book. The definitions therein shall be considered normative for this book. Additional terms follow.

2.1.1. Container

A physical or logical entity into which data may be stored.

Numerous conversations in various committees have demonstrated a need for this seemingly obvious definition. Example containers in SSM are StorageSystems, StorageVolumes, StoragePools, FileSystems and Shares, and ObjectStores.

2.1.2. SMI

The Storage Management Initiative (SMI) is an Initiative within the SNIA umbrella organization. More information can be found at

http://www.snia.org/forums/smi

2.1.3. SMI-S

The Storage Management Initiative's Specification (for storage management).

This specification (SSM) is built as an extension to SMI-S. SMI-S documents may be found at

http://www.snia.org/tech_activities/standards/curr_standards/smi

2.2. Normative references

The following references are normative for this specification.

DMTF DSP0200	CIM-XML
DMTF DSP0210	CIM-RS 2.0 or newer
DMTF DSP0223	Generic Operations 2.0 or newer
DMTF DSP0212	Filter Query Language (FQL)
DMTF DSP8016	WBEM Operations Message Registry
IETF RFC 2608	Service Location Protocol (SLP), v2
IETF RFC 3721	iSCSI Naming and Discovery
IETF RFC 3722	String Profile for iSCSI Names
IETF RFC 5424	Syslog
IETF RFC 4122	UUID URN Namespace
SNIA TLS spec	Normative guidance on the implementations and use of TLS http://www.snia.org/tech_activities/standards/curr_standards/tls

2.3. Use cases

A central tenet of this work is that functionality and design must be driven by use cases and IT workflows. Table 1 lists the use cases that have been identified as applicable to basic storage management. The center column indicates which version of SSM introduces support for the use case.

Table 1 Use cases

Area	Release target	Use cases / workflows
Pool operations	1.0	1. Get the canned (predefined) settings available for pools
	1.0	2. Create a single storage pool from available storage elements
	1.0	3. Create multiple storage pools from available storage elements, possibly with different service levels
	1.0	4. List all the pools of various types
	1.0	5. Modify some pool parameters
	1.0	6. Erase the data in a pool
	1.0	7. Decompose an aggregate pool into its constituent parts
Container operations. A container is a pool, filesystem, object store, volume, file directory, or object container ("bucket").	1.0	8. Get the canned (predefined) settings available for each container type
	1.0 (partial)	9. Create volumes, filesystems or object stores on one or more pools
	1.0	10. Modify some container parameters
		11. Erase the data in a container
	1.0	12. Delete a container

Area	Release target	Use cases / workflows
Container export to hosts	1.0	13. Map volumes to hosts, NodeWWNs or PortWWNs
	1.0	14. Unmap volumes
	2.0	15. Export shares from filesystems
	2.0	16. Set, modify and delete ACLs on shares
	2.0	17. Unexport shares
	…	18. Create access points to object stores
	1.0	19. Support named volume groups in order to correlate SSM volume groups with volume groups exposed by vendor tools
Multipathing	1.0	20. Use all available paths between server and host
	…	21. Manage multipathing
Performance	2.0	22. Get performance stats for storage systems, pools, volumes, filesystems, filesystem containers, object stores or object store containers
	2.0	23. Get performance data for ports and links
	1.0	24. Get information on space usage for systems and containers

Area	Release target	Use cases / workflows
Replication	2.0	25. Copy a volume, file, container, filesystem, object or object store to the same storage system or a remote storage system
	2.0	26. Synchronously mirror a volume, file, container, filesystem, object or object store to the same storage system or a remote storage system
	2.0	27. Asynchronously replicate a volume, file, container, filesystem, object or object store to the same storage system or a remote storage system
	2.0	28. Synchronize a volume, file, container, filesystem, object or object store to the same storage system or a remote storage system (syncing an iPod to an iMac is a familiar example).
	2.0	29. Suspend replication or mirroring (e.g. to allow a database to flush)
	2.0	30. Resume replication or mirroring
	2.0	31. Break (detach) a replication or mirror relationship
	2.0	32. Reattach a replication or mirror relationship
Long running operations	1.0	33. Find out how the jobs running on the system were invoked
	1.0	34. Get the current status of the jobs running on the system
	1.0	35. Get the final status and return values of completed jobs
Events	1.0	36. Use lifecycle events to keep client-side databases current
	1.0	37. Use typed events from a message registry to alert admins to conditions requiring administrative action
Navigation	1.0	38. Provide links to the SMI-S representation of systems, pools, volumes, disk drives, etc. so that more complex management tasks may be performed there.

Area	Release target	Use cases / workflows
Security	1.0 2.0 2.0	39. Use TLS and HTTP for secure authentication 40. Use roles to separate management functions 41. Enable secure multi-tenancy by placing ACLs on systems and pools.

Use cases in Table 1 that are not introduced in version 1.0 are planned as release drivers for upcoming releases. Much of the design work on these has been done and kept in reserve; SSM v1.0 is purposely kept small to avoid confusion as the developer community begins the prototyping and adoption process.

3. The SSM Model

3.1. Overall design

SSM is part of a larger effort to simplify and unite several management initiatives under one umbrella, enabling vendors to offer complete data center management solutions without having to rely on a mishmash of annoyingly similar yet different standards.

As part of this effort, a new schema hierarchy has been defined. The WBEM schema, which contains classes pertaining to all management initiatives, resides at the top of this hierarchy. SSM is the first domain-specific management initiative to conform to this design.

The WBEM_Server schema contains classes pertaining to the WBEM infrastructure, i.e. CIMOMs such as Pegasus. Other initiatives to manage servers and networks, perhaps simplified forms of SMASH and DASH, are envisioned but not yet design complete.

3.2. Naming

The system specified herein is called the Simple Storage Management (SSM) model. This model drastically flattens the SMI-S class hierarchy into a small number of classes with many properties. It shares some concepts with the View class model but is designed to radically simplify the class hierarchy.

The model is contained in its own schema. All classes not in the WBEM schema belong to the `SSM` schema, e.g. `SSM_StorageServer`. The convention in this specification, as in SMI-S, is to omit the "`SSM_`" schema prefix, and refer to classes without the prefix. E.g. `SSM_StorageServer` becomes StorageServer. The reader is cautioned that there are a couple name conflicts with the CIM schema. When we talk about a StorageVolume or StoragePool in this book, it always means `SSM_StorageVolume` and `SSM_StoragePool`, respectively. If `CIM_StorageVolume` or any other class in a different schema needs to be mentioned, it is explicitly spelled out.

3.3. Design considerations

There are several requirements and design principles that have guided the development of the WBEM and SSM models.

- We need to use or at a minimum allow use of the existing infrastructure, e.g. CIM-XML, CIM v2, CIM-RS.
- We want a 90% solution[2] that new providers and clients can use to get going. When the full richness of the existing CIM schema is needed, clients can move back up to the scoping system and traverse to the existing CIM model.
- Model the use cases, not the implementations. Functionality should be focused on what clients want, rather than what the server can or could do. Especially try to avoid describing how the server does what it does.

[2] To be clear, we want a 10% solution that addresses 90% of the problem.

- Rather than torturously attempting to prevent edge conditions—leading to large specs—attempt instead to make dealing with them easy (for the implementer). This applies, for example, to rare error conditions that may be vendor specific. In cases like these, instead of trying to specify everything in some "interoperable" format—inevitably full of vendor extensions—it may be best to simply provide a way for the client to get human readable text in front of an admin who can then decide what to do.
- Storage Servers usually contain the most expensive CPUs in the room. To the extent possible, use a "smart client/dumb server" model.

3.4. Security considerations

Security considerations include transport level security, authentication and authorization, and *roles* which define the type of operations various types of users are permitted to perform.

3.4.1. Transport security

TLS transport security is required for both clients and servers. All implementations shall conform to the most current version of the SNIA TLS Security Specification, which can be found at

http://www.snia.org/tech_activities/standards/curr_standards/tls

The TLS specification is updated when new vulnerabilities are found. New SSM implementations shall support the latest specification within six months of its publication announcement.

3.4.2. Authentication and authorization

CIM-XML (DMTF DSP0200) and CIM-RS (DMTF DSP0210) are required by SSM. Please consult these specifications for authentication and authorization requirements.

3.4.3. Security roles

Important note: this section is optional in release 1.0 of this specification. It will be made mandatory in a future version.

SSM defines four types of users. All are administrators of some type. End users access storage through other protocols such as Fibre Channel, SMB, NFS and CDMI. In this version of SSM, the full richness of an RBAC model such as INCITS 359-2004 is not felt to be needed.

Nevertheless, SSM departs from traditional CIM usage in order to be compliant with modern security practices. At least three user accounts are required to manage an SSM implementation. The first comes pre-configured on a new system and is the security officer account. The second is an Administrator account, the third an Operator account. Finally, a Monitor account allows for read-only users that do monitoring and performance management. These four security roles in SSM work as follows.

1. Security Officers. These users have exclusive rights to manage Admin accounts. They may also manage Operator accounts and other Security Officer accounts. New systems shall ship with a single Security Officer account and password of the vendor's choosing. A first task during system setup is to change the password and create one or more Administrator accounts and possibly other Security Officer accounts.

Security Officers shall also have exclusive management rights over security logs. This includes setup and distribution of logs. Note however that although generating external copies of security logs is strongly recommended, securing those is the responsibility of the IT shop running the system.

2. Administrators. These users have exclusive rights to create and manage StoragePools. They may also manage Operator accounts. During system setup, Administrators aggregate the Basic pools on the system (see Section 4.4.4) into one or more higher level pools and assign rights to these pools to Operator accounts. Administrators are responsible for managing the physical storage on the system and its representation in CIM.

   Note: Version 1.0 of this specification does not cover authorization of Operators. Support for authorization via ACLs will be added in a future version.

3. Operators. These users are assigned management rights to one or more StoragePools, Filesystems, Volumes and ObjectStores by Administrators. From these they allocate filesystems, shares, object stores, volumes and volume mappings (as supported by the system) and manage access to them by external hosts.

4. Monitor. These users are able to read information, such as performance data, from anywhere in the system that they are given rights to.

To summarize, these four roles manage the security, physical and logical aspects of an SSM StorageServer. Support for all four roles is required.

3.4.4. Security Role management

Important note: this section is optional in release 1.0 of this specification. It will be made mandatory in a future version.

The following operational rules and steps to follow during system initialization shall be in place and followed by all compliant SSM implementations.

1. All actions by Security Officers and Administrators shall be logged via a mechanism of the vendor's choice. Support for remote syslog is recommended. As indicated by the RequiredIndications qualifier on class methods, implementations shall also issue appropriate Alerts.

2. Actions by Operators should be logged via the same mechanism(s) used in (1).

3. No action by any user except the factory-configured Security Officer account shall be permitted on a new system.

4. Upon first action of the Security Officer, he or she shall be required to change the account password, entering it twice for safety. Password strength rules are at the discretion of the vendor; it is recommended that a minimum of 8 characters, with at least two of a capital letter, special character, and a number, be required.

5. It is recommended that vendors check at this time that no other accounts have been placed on the system by outside parties.

6. The Security Officer shall not be permitted to invoke any storage management functionality. This means effectively that the only methods callable by the Security Officer are contained in the WBEMServer class, which has methods for account and log management. The mechanism for enforcing this is left to the discretion of the vendor.

7. The Security Officer shall use the WBEMServer.CreateAccount( ) method to create one or more Administrative users. In this version of SSM, all Administrators may have full control over the entire storage system. Care should be exercised to avoid conflicting workflows.

8. Upon first action of an Administrator, he or she shall be required to change their password, using the same rules as in (4).

9. Administrators may create Operator accounts. They shall not be permitted to create or otherwise manage Administrator or Security Officer accounts. Upon first action of an Operator, they shall be required to change their password, using the same rules as in (4).

10. Administrators shall be able to assign rights to specific (or all) systems and pools to specific Operators via the WBEMSterver.AssignRights( ) method. This is not modeled in SSM, except in the output of ListAccounts( ); vendors shall track and enforce Entity ownership and sandboxing in a vendor-specific way. For technical infrastructure reasons, ability to assign rights to individual filesystems, volumes and so on is not on the roadmap at this time.

11. Password change requirements are not mandated by SSM except as follows. No Security Officer account shall be required to periodically change its password (except as dictated by customer policy). Other accounts may have password change policies, at the discretion of the vendor and customer.

The above rules effectively mandate that there shall be a minimum of three accounts on a given WBEMServer before the system can be made ready for use. They may, at the discretion of the customer, be owned by one person. I.e. in smaller configurations, the Administrator may create an Operator account with management access and control over all pools in the system and proceed to manage the system as has usually been done in SMI-S, with a single user having root-like privileges. This practice is discouraged in all but the smallest shops, however. In large configurations with many volumes and shares, assigned to multiple tenants, guidance is to create Operator accounts for each tenant and assign ownership of pools accordingly.

User documentation shall caution users to comply with the security practices and rules of their organization. The guidance regarding storage security given in ISO/IEC 27040 is recommended but not required by this specification.

3.5. Discovery

SSM implementations shall support SLPv2, as specified in DMTF DSP0205, WBEM Discovery Using the Service Location Protocol (SLP). SLPv2 is defined in IETF RFC 2608.

3.6. Messages

SSM supports two types of messages: Errors and Indications. Errors are represented as instances of the WBEM_Error class. Indications may be of several types, and are represented as instances of the WBEM_Indication class.

3.6.1. Errors

Errors shall be WBEM_Errors with a MessageID that contains a string from an entry in the message registry, which is found in the SSM distribution by the name `SSM_messages.xml`. Errors shall be encoded appropriately (i.e. XML for CIM-XML, JSON for CIM-RS) and included in the header of return messages. It is permitted and appropriate to return more than one error when multiple errors or combinations of errors are encountered.

> *Note: this version of the SSM spec contains the standard messages in table format in Clause 6. They are not yet available in XML format.*

A key component of errors is the RecommendedActions property. This property does not necessarily have standardized Values. In this version of this specification, it is designed to provide text that a client may display to a human administrator, recommending one or more courses of action designed to either fix the current error or prevent future occurrences of it. Implementers should make best effort to provide useful text in this property. Short strings containing accurate but non-useful information should be avoided. When it is not possible to adequately ascertain what to recommend, vendors should leave this property as blank or NULL.

At this date, all that is known about any kind of standardized RecommendedActions is that clients want them. Implementors are encouraged to develop an internal table of recommended actions and index into it to fill in the RecommendedActions property. At a future time, the SSM development team will attempt to survey implementors to see what, if any, common elements from these tables can be standardized.

3.6.2. Indications

Indications report changes in system or system element state. Lifecycle indications for element instance creation, modification and deletion may be required, depending on the class of the element. Support for alert indication is required for any classes that model things that can break. These requirements are specified in the RequiredIndications qualifier for each class (if there is no such qualifier on a class, Indications support is not required on that class)[3].

A conforming implementation of the WBEM and SSM models may legally provide only indications conforming to the types listed in RequiredIndications, and return errors on all other subscription attempts. However, note that support for other SMI-S profiles may require support for other CQL-based subscriptions and filters.

SSM does not use CQL or FQL for indication subscription. Instead, clients use the WBEMServer's SubscribeToIndication( ) method to subscribe to indications.

[3] In general, only concrete classes that have InstanceIDs require Indications.

All indications shall use one of the predefined messages in the Standard Message Registry, which accompanies the distribution, or in DMTF DSP8016, WBEM Operations Message Registry. A VENDOR_MESSAGE message is defined for situations that are not covered by the Standard Messages Registry. Vendors are encouraged, however, to bring such messages to the SSM development team for possible standardization.

3.6.2.1. Bellwether indications

Bellwethers are higher level events, such as the failure of a server. When a server fails, it is not useful to anyone to get a flood of messages regarding a change in state of its disk drives, network interfaces, power supplies, fans and so on. Bellwethers are therefore designed to signal events from which many other downstream events can be inferred.

Some guidelines are in order. In general, events that cause other events are candidates for bellwether. Bellwethers should, when possible, populate the AffectedElements array of the message with the InstanceIDs of the elements that would otherwise—without the bellwether—cause alerts. "When possible" means that the bellwether event being reported is not fatal to the point that further processing is not possible.

As noted above, this section is a candidate for deletion. Comments solicited.

3.6.2.2. Indication Subscription

As stated previously, clients use the WBEMServer_Server's SubscribeToIndication method to subscribe to a particular indication type. This method is documented in the WBEMServer_Server class notes. The SubscribeToIndication method allows subscription to any of the indication IDs from the Indication registry.

As mentioned before, the Indication registry is currently documented in clause 6.

3.7. CIM considerations

The following CIM features are required to be supported by implementations of the WBEM and SSM models. Implementers should specifically note the special requirements called out in sections 3.7.5, 3.7.7 and 3.7.8.

3.7.1. The Conditional qualifier

The Conditional qualifier is used liberally in the SSM MOF. This is a normative qualifier that uses a superset of FQL to express requirements for the existence of or support for a property or method. However, normative text imposes requirements on human readers, not directly on parsers. There is no requirement in v1.0 of this spec that an implementation be able to parse and make use of Conditional text.

3.7.2. FQL

FQL is specified in DMTF DSP0212, Filter Query Language. It is required for support of Open EnumerateInstances, Open Associators and OpenReferences. Implementations shall support the PropertyList parameter of these methods.

Other support requirements are limited to any automated processing of Conditional qualifiers that an implementation or infrastructure may wish to do.

3.7.3. EmbeddedInstances

EmbeddedInstances are heavily used in the WBEM and SSM models. Most uses are to support the language construct of *structures*, which has been present in CIM v2 since v2.36. An embedded instance is a string that encodes an instance of a named class. For example, in Figure 1, the Print method of class Foo will expect to find a string with 64-bit quantities encoded in it for the baz and bol properties.

```
   [Description(
       "A base class")]
class ACME_Bar {
      [Description("A class property")]
   uint64 baz;
};

   [Description(
       "A child class with a method that takes an "
       "embedded instance parameter.")]
class ACME_Foo : ACME_Bar {
      [Description("Another class property")]
   uint64 bol;

      [Description("A method taking an embedded instance parameter
")]
   uint16 Print(

         [Description("An instance of foo for printing"),
          EmbeddedInstance("foo")]
      string Item;
   );
};
```

Figure 1 **Embedded Instance example**

In CIM-XML, this string will look like

```
<class Name="Foo"> <property Name="baz"> 0xD329AC0043 </property> <property Name="bol"> 0x101 </property> </class>
```

3.7.4. ObjectPaths

ObjectPaths are also heavily used in SSM. These function like REFs but unlike REFs, can be stored in arrays of strings. The existing CIM qualifier "Reference" shall be used throughout the SSM MOF to indicate that a string is an ObjectPath.

To summarize, EmbeddedInstances are generally used where mainstream languages would use call by value, and where static data is acceptable. ObjectPaths are used where mainstream languages would use call by reference, and for situations where ongoing live update to and from an object are needed.

3.7.5. Valuemaps

The Values qualifier is not used in the WBEM and SSM models. Implementations transmit string values directly in the Valuemap for the qualified element. Standard strings are specified in the Valuemap qualifier of the element.

Vendor extensions to the Valuemap enumeration given in the MOF are not permitted in SSM, due to CIM v2 rules. The mechanism for vendor extensions is as follows:

- When vendor extensions are deemed appropriate or other values than those given in the array are deemed possible and appropriate, authors shall include "Other" in the Valuemap qualifier for the given element.
- Vendors shall use this in subclasses with a property that shall be named "Other". There is no restriction on the type of this extended property, therefore, arbitrary trees and arrays of data are possible. The value of the qualified element shall be set to "Other" to indicate that the Other property should be examined for the actual value.

3.7.5.1. Example

The following example illustrates the use of Other in a vendor class based on a hypothetical class Foo in the SSM schema.

```
class SSM_Foo {
      [Valuemap{ "A", "B", "C", "Other"}]
   string Bar;
};

class MY_Foo : SSM_Foo {
      [Valuemap { "A", "B", "C", "Other"}]
   string Bar;

      [Description(
          "Structure containing info about Acme Corp's "
          "special FooBar sauce."),
       EmbeddedInstance("MY_BarInfo")]
   string Other;
};
```

This discussion may be reduced to a reference to the corresponding text in the Simple WBEM specification, currently under development.

3.7.6. Associations

Arrays of EmbeddedInstances and ObjectPaths are used in many—but not all—cases to replace associations. The rule of thumb is that if the number of elements is expected to be small, an array is more appropriate than requiring a client to chase an association. For large numbers of elements, the existing association machinery or extrinsic methods appear to be more appropriate.

Another way to think about this is that the model only uses associations to link otherwise unrelated model elements, e.g. a storage server and its pools and volumes. Elements that are part of other elements, e.g. the supported features of an array, are included as properties in the enclosing element.

3.7.7. Intrinsics and Extrinsics

In CIM, "Extrinsics" are simply class methods that perform operations that are not provided by CIM itself via "Intrinsics." `GetInstance()` is an example of a CIM Intrinsic method. It constructs or returns an instance of a class given an instance of its key. This works for all keyed classes. Extrinsics, on the other hand, are specific to the class containing them.

The WBEM and SSM models supports a very limited set of Intrinsics. All SSM providers are required to support:

- GetInstance (on classes with keys)
- Open EnumerateInstances (on classes with keys)
- Open Associators
- Open References
- Pull Instances
- InvokeMethod (on classes with methods)
- CloseEnumeration

FQL support (DMTF DSP0212) is required for Open EnumerateInstances, Open Associators and OpenReferences.

Other functionality provided by the complete set of CIM Intrinsics is either not supported (e.g. AssociatorNames), or is provided by Extrinsic methods such as `GetCannedSettings()`.

3.7.8. Qualifiers

The WBEM and SSM models introduce several new qualifiers and drops many of the qualifiers in the CIM schema. A key requirement in the WBEM and SSM models is that *all class properties are mandatory* unless otherwise qualified by one of the new qualifiers Optional and Conditional.

3.7.8.1. New qualifiers

The new qualifiers are shown in Table 2 .

Table 2 New Qualifiers

Qualifier	Description

Qualifier	Description
Optional : boolean = false	If this qualifier is in place, the property, method, parameter or class is nominally optional to provide. Normative text in the description may indicate conditions under which the element is not optional, i.e. mandatory. (This is allowed for times when providing a conditional qualifier is too messy or complex).
Conditional : string <condition>	The property, method, parameter or class is mandatory to provide if the condition is true. Otherwise it is optional. The conditions used in the SSM MOF are simple expressions based on FQL and designed to be automatically parseable and used in CTP tests. There is no requirement for automatic parsing of Conditional text, however, in v1.0 of this specification.
RequiredIndications : string [ ]	An array of strings stating which indications shall be supported for a given class. This qualifier is never used on Info classes (structures). When an embedded structure is created, modified or deleted, this shall be treated as a modification to the containing class object.
Comment : string	The Comment qualifier may be used by authors to discuss issues. It is ephemeral and non-normative, and may be changed or removed by authors at any time. It should usually be stripped out by utilities that transform MOF into other formats aimed at non-author use.
Errors : string [ ]	The SSM_Object class defines a set of canonical error messages. Additionally DSP_8016 defines errors at the CIM protocol level. This qualifier lists any errors above and beyond those others that a given method may return.

Qualifier	Description
Tickets : string [ ]	An open source ticketing system called Trac is used internally at SNIA to track issues during SSM development. This qualifier may containTrac ticket numbers that are related to how an element came into its present state in the spec.

The complete list of qualifiers used in SSM is found in the files `SSM_Qualifiers.mof` and `WBEM_Qualifiers.mof` that comes with the distribution.

3.7.9. Infrastructure considerations

3.7.9.1. Schema

The schema for all SSM classes shall be "SSM". At present, all SSM classes are concrete. The abstract classes have been moved into a new "WBEM" schema.

3.7.9.2. Namespace

For any implementation, there shall be an "interop" namespace containing an instance of "SSM_Namespace" for each namespace supported by the CIM server.

3.7.9.3. Access Protocols

CIM-RS v2 is mandatory for implementations to support. CIM-XML support is recommended. WS-MAN may be supported on an optional basis.

3.7.9.4. Indications

Indications are sent using instances of a class called WBEM_Indication in the WBEM and SSM models.

Standard messages are documented in Clause 6. This table is automatically generated from the Standard Message Registry that accompanies the SSM distribution, and is normative.

3.7.9.5. Jobs

Jobs are simplified versions of CIM_Jobs. Details of their operation are given in Clause 5, in the documentation for SSM_Job.

Jobs will be moved to the WBEM schema in the near-term future.

3.7.10.Diagram rules

In the diagrams in this specification, plain black lines represent associations. Green lines indicate that an object has a property (whose name is listed) that contains one or more EmbeddedInstances or ObjectPaths to other objects of the indicated type. This is so that model traversing can be shown clearly. See Figure 2 for an example. Red lines indicate method calls.

4. System Design

4.1. Overview

The SSM classes represent, as much as possible, major systems objects like controllers, disks, disk groups, network adapters and so on. A high level view of the system is shown in Figure 2.

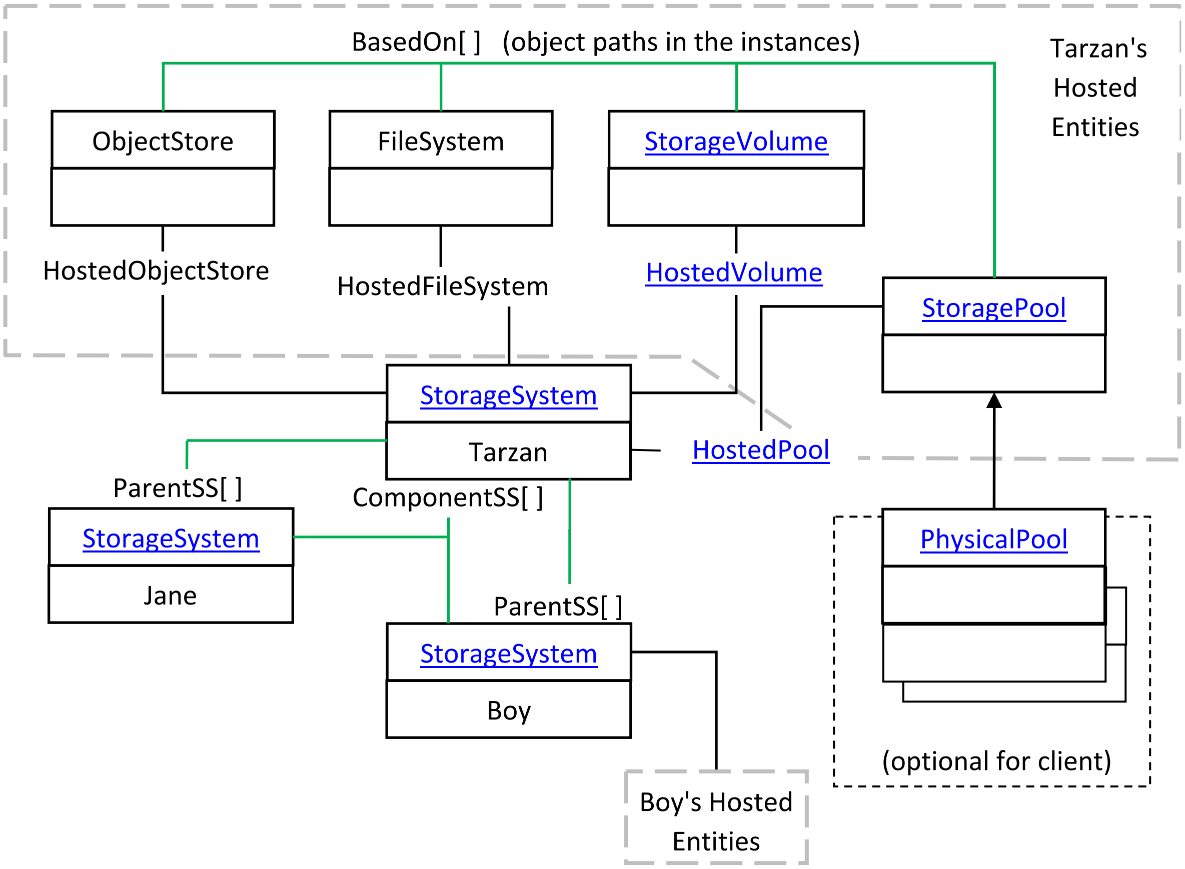

Figure 2 **Basic high-level system model**

Note that the associations to hosted objects follow a slightly different rule from the larger schema. In the larger schema of SMI-S, under the rules of the MultipleComputerSystem profile, what SSM calls HostedPools are associated via CIM_SystemDevice to the top-level system, when their availability matches that of the top-level system. In the WBEM and SSM models, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. In Figure 2, Jane is the name of the top-level system found during discovery. The actual resources hang off of Tarzan and Boy, which might be Virtual Machines (VMs) or an HA pair (HA = High Availability). In the latter case, the diagram remains the same during failover, because Tarzan is impersonating Boy or vice versa.

4.2. Abstract Classes

There are four abstract classes in the WBEM model: WBEM_Object, WBEM_KeyedObject, WBEM_Association, and WBEM_Device. They are summarized briefly here. More detail is available in the MOF and in the upcoming WBEM Schema spec.

4.2.1. class WBEM_Object

This is the top level class. It can be used to represent arrays of SSM objects of arbitrary types. As it has no properties, it is much used as the parent type for structures (which are embedded instances in this version of the spec). It is also handy as a parent for any classes that are used for simple lists such as log line listings etc.

4.2.2. class WBEM_KeyedObject : WBEM_Object

This class contains an InstanceID, defined as a 64-bit number conforming to RFC 4122. This is the only key allowed in the WBEM and SSM models, and all keyed classes inherit from it.

4.2.3. class WBEM_Device : WBEM_Object

This is the abstract base class for all primary system objects. These are persistent objects keyed on InstanceID. Device elements are described in Table 3 .

Table 3 Device properties

Device Properties	Description and Requirements
key string **InstanceID**	An identifier guaranteed to be unique within the managed system's provider. InstanceIDs shall be a 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string SystemID	The InstanceID of the scoping (enclosing) system. Top-level devices that *are* the scoping system shall populate this property with NULL.

Device Properties	Description and Requirements
string DurableName	A persistent vendor-defined name for the object. The rules regarding durable names given in clause 7 of the SMI-S Architecture book shall be followed with the following further restrictions: • SCSI Logical Unit names shall be in VPD page 83 type 3 (NAA) format • Fibre Channel ports shall use Port World Wide Names (i.e., FC Name_Identifier) • iSCSI targets shall use a dotted IP Address concatenated with a colon and a TCP port number, e.g. "10.1.219.5:9988". Clause 7 mentions "other types of devices" with specific guidance. This book interprets this to mean "other types of ManagedElements". As they are not ManagedElements, such sections of clause 7 are not applicable to Devices.
string DurableNameFormat	Valuemap: "Text", "VPD83type3" A description of what the DurableName signifies. Subclasses may override this to indicate other formats. The default shall be Text.
string GivenName	A name given by the user. Vendors should initialize it with the DurableName or an empty string. After being set by the client, the GivenName shall have the same persistence properties as the DurableName, except that a client may change it at any time.
string AssetTag	A persistent asset tag, typically contributed by an asset or inventory management system, and set by the client. This shall be an alphanumeric string that is empty until set by the client.
string Tag	When a FRU is taken out and replaced, some manufacturers "tag" the new FRU with an identifier that links it to the original FRU for warranty or other purposes. This shall be a persistent alphanumeric string that is empty until set by the client.

4.2.4. class WBEM_Association

The Association class is the abstract parent of all SSM association classes. The "Source" and "Target" properties have the same meaning as "Antecedent" and "Dependent" in the CIM schema.

4.3. Class Types

The SSM model consists of six kinds of classes.

1. Abstract classes from the WBEM schema, as discussed previously.

2. Device classes, inherited from WBEM_Device. These represent major system objects.

3. Info classes, inherited from WBEM_Object. These are structures. They may be nested, just as in other modern programming languages, but an effort should be made to constrain the nesting level. A single nesting level has been the aim of the designers of the initial version of SSM. Device classes routinely use Info classes to structure and manage the properties they contain.

   Due to the rules of CIM v2, Info classes must be declared using the Indication qualifier to indicate to the CIM machinery that they don't need InstanceIDs. They shall all also use the Structure qualifier.

   This specification uses the terms "Info class" and "structure" interchangeably. All true structures in SSM, that is, concrete classes with no InstanceIDs, are named with an "Info" suffix. E.g. SSM_ReplicaInfo and SSM_LocationInfo.

4. Settings classes, inherited from WBEM_KeyedObject. These are much like structures, but have InstanceIDs. The SSM model does not support "Goals". StorageSystems shall maintain a list of at least one canned setting applicable to each supported container type. Canned settings are predefined by the system; they shall be pre-configured out of band and discovered by the provider in a vendor-specific way.

5. Association classes. These mostly associate storage systems with all the pools, volumes, filesystems and so on hosted on them. All are inherited from WBEM_Association.

6. Utility classes. The CIMElement class provides a way of linking to elements in the SMI-S CIM schema within the context of the SSM model. Using this, providers can, for example, provide a ManagedElement or an Object in a single reference, depending on where the given object lives. The WBEM_Job, WBEM_Error and WBEM_Indication classes are similar to their counterparts in the CIM schema, but contain pointers to Objects instead of ManagedElements and have been simplified. All utility classes are mandatory.

4.4. Required classes for the base system

4.4.1. Required abstract classes

The following abstract classes, listed in Table 4 , are mandatory in the SSM model. All class properties are also required.

Table 4 Required abstract classes

WBEM_Object WBEM_Device	WBEM_Association WBEM_KeyedObject

4.4.2. Required concrete classes

The concrete classes listed in Table 5   are also mandatory in the SSM model. Note that some class elements, particularly methods, may be optional depending on the type of storage and operations that are supported by the managed storage. However, all contained properties and methods are mandatory unless otherwise indicated via the Optional or Conditional qualifiers.

The SSM model departs from the CIM schema in requiring Indications and Jobs. This is necessary to help clients avoid numerous and expensive scans of large storage arrays with thousands of volumes and other managed objects.

Table 5 Required classes for the base system

WBEM and WBEMServer classes	
WBEM_Error	An error, usually as a result of a method call
WBEM_Indication	An indication of an event
WBEM_Job	A job to accomplish a long-running task
WBEM_MethodParameterInfo	Info about input and output job method call parameters
WBEM_InitiativeRegistration	Models supported by the WBEMServer_Server
WBEMServer_Server	The infrastructure server
Device classes	
SSM_StorageSystem	A storage system
SSM_StoragePool	A pool of storage
Structures (Info classes)	
SSM_RedundancyInfo	Info about the redundancy level of a storage object
SSM_OperationalStatusInfo	System status
SSM_MethodParameterInfo	Info about a method call being served by a Job
SSM_AccountInfo	User account data
SSM_BasedOnInfo	Info about pool and volume basis
SSM_ProvenanceInfo	Destination info
SSM_KVInfo	A key-value pair
SSM_SpaceInfo	Info about space used, etc.
Settings	
SSM_StorageSetting	Parameters for various storage objects
SSM_WBEMServerSetting	Parameters for a WBEMServer
Associations	
SSM_HostedPool	StoragePools hosted on a StorageSystem
SSM_StorageSystemJobs	Jobs incarnated on a system
SSM_StorageSettings	Settings available on a system
SSM_ManagedSystems	Systems discovered and managed by WBEM Server
SSM_InitiativeSupported	Required subclass of WBEM_InitiativeSupported
Utility classes	
SSM_CIMElement	Reference to an object in the CIM schema

4.4.3. Base system element diagram

The instance diagram for the base system elements is shown in Figure 3. Also see Figure 2.

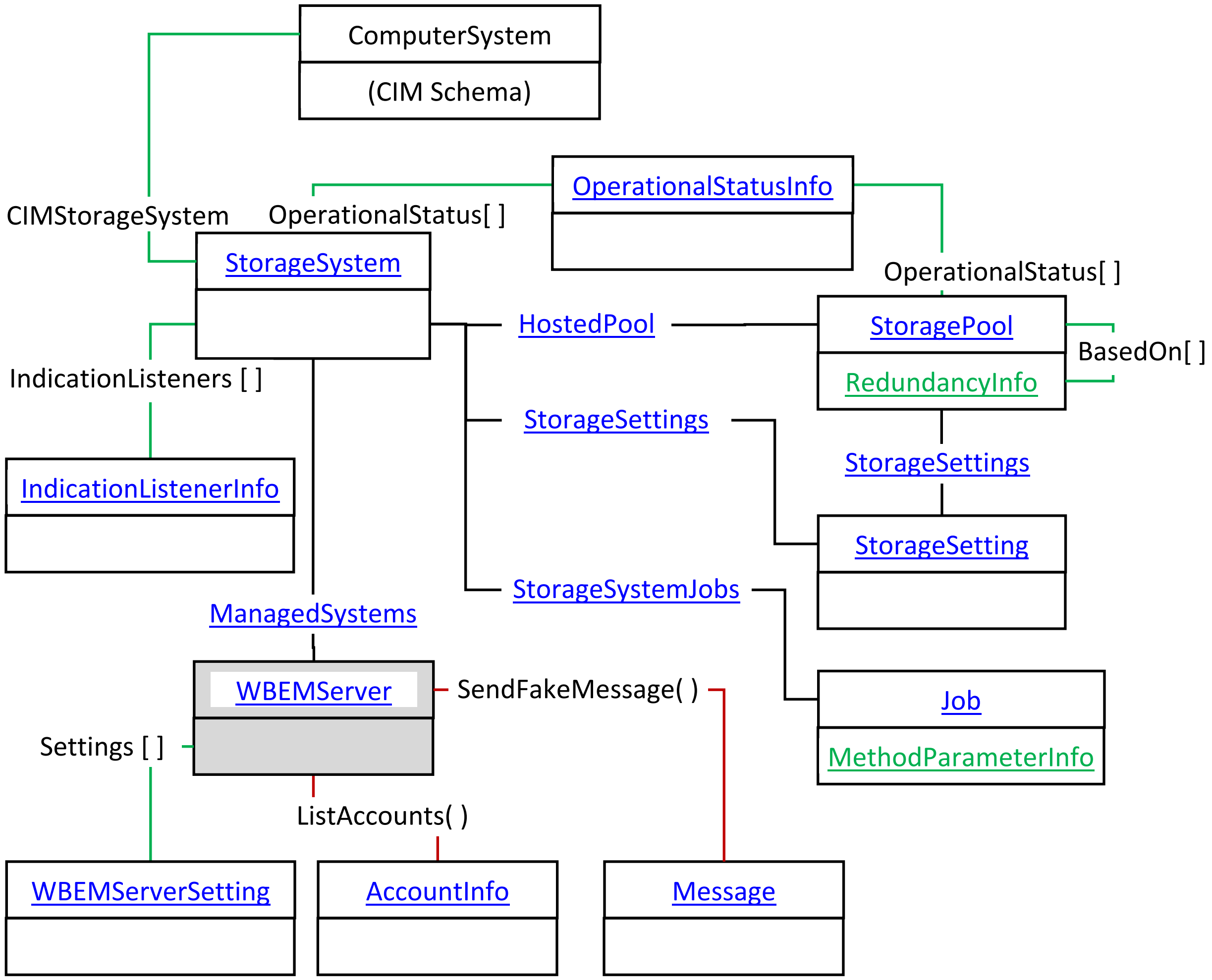

Figure 3 **Base system instance diagram**

Info classes are mostly not shown in Figure 3 because they are structures passed around as embedded instances, not real class objects with InstanceIDs.

4.4.4. Multiple Storage System requirements

Storage Systems in HA and clustered environments are related by their ParentSS[ ] and ComponentSS[ ] properties. Pools are associated to the physical systems that manage them. The instance diagram in Figure 4 shows an example configuration.

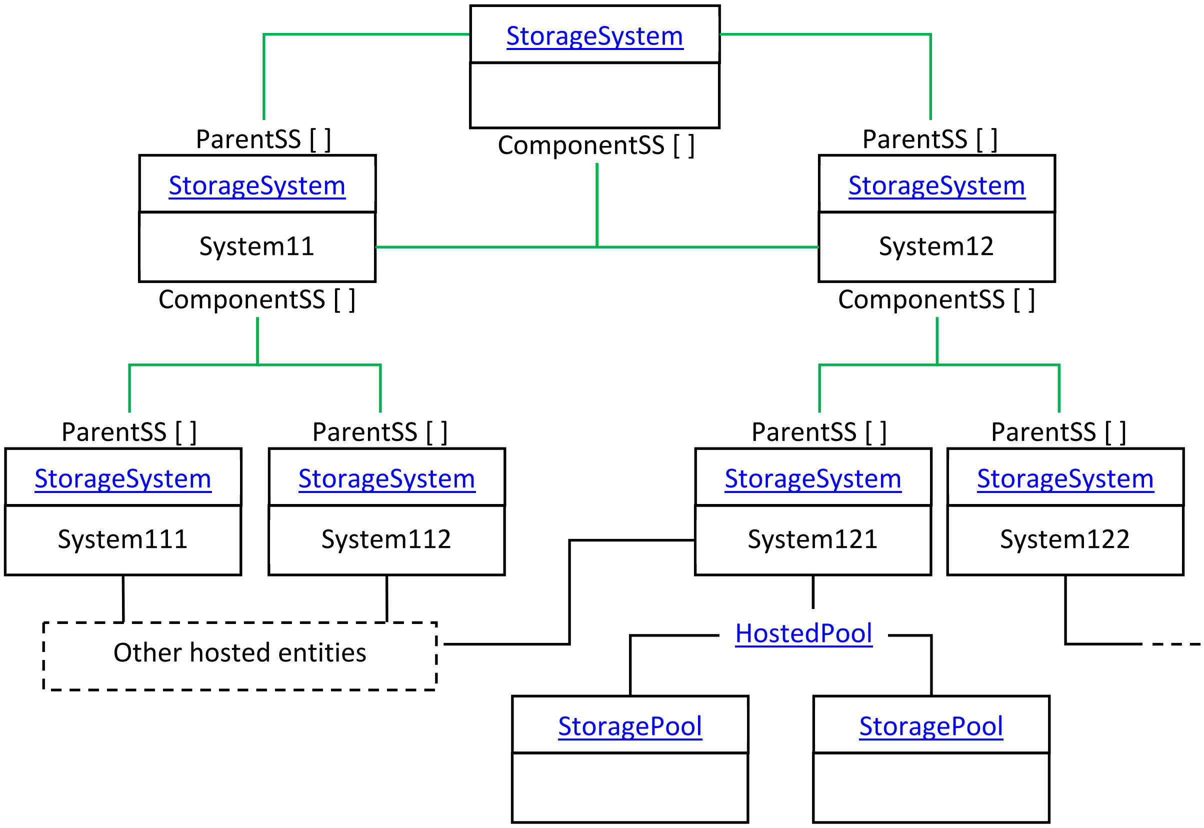

Figure 4 **Example tiered storage system**

It would be possible to establish either service methods or associations to get all the pools, volumes, etc. for a top-level system. Is this something clients want? There would be security issues that implementers would have to sort out and get right for when multi-tenancy is in the picture.

The picture gets more complex as volumes, filesystems and object stores are added. Virtual systems introduce additional complexity.

The diagram in Figure 5 illustrates a scenario in which there are virtual components and data containers at various layers.

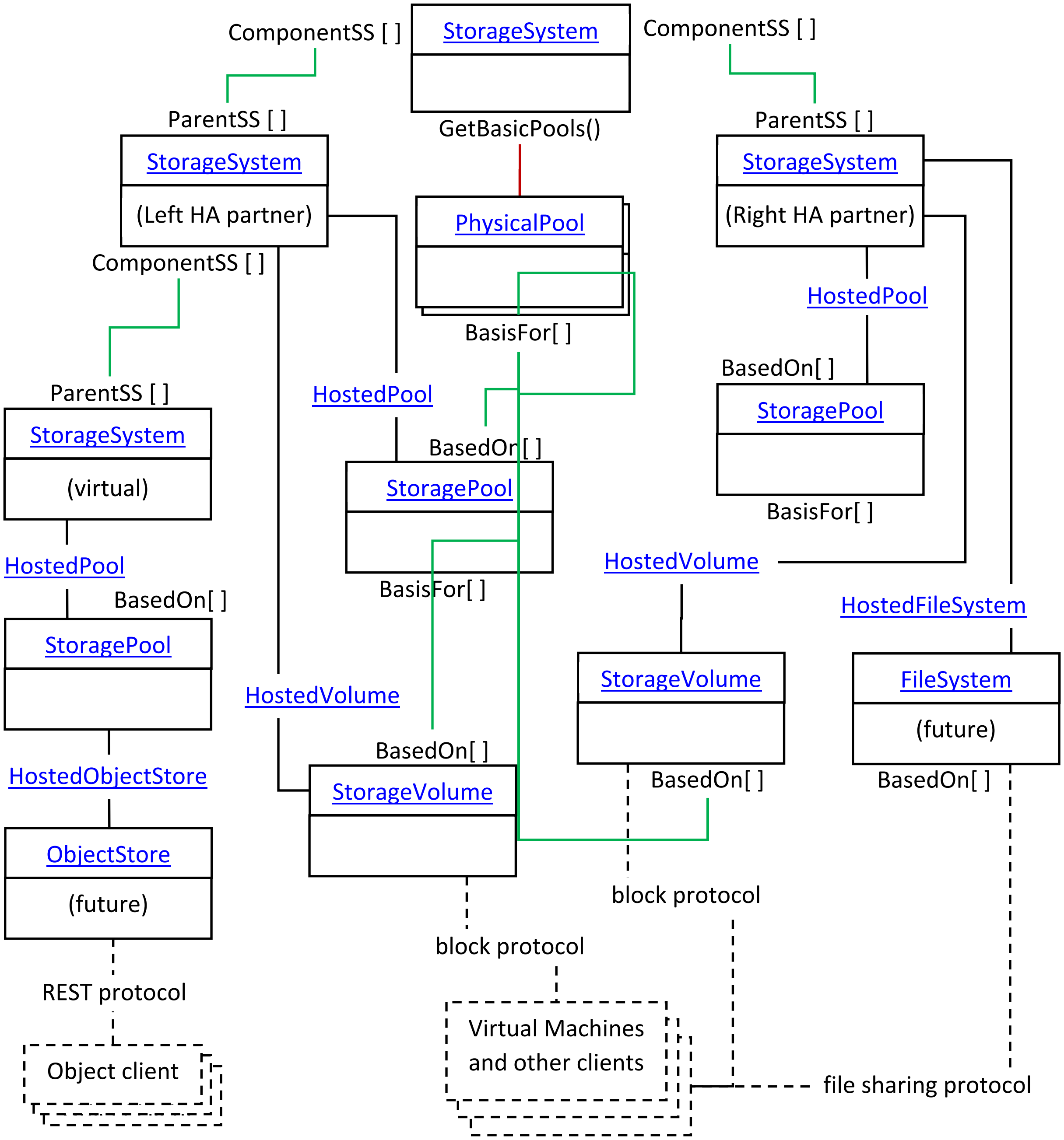

Figure 5 **Example tiered system with virtual components**

4.5. Required classes for physical hardware support

The classes listed in Table 6   are mandatory to implement if physical hardware is present in the system[4]. StorageSystem. SupportedElements[ ] shall contain "Physical System". All properties and methods are mandatory unless otherwise indicated via the Optional or Conditional qualifiers.

Table 6 Required classes for physical storage devices

SSM_PhysicalPool	A pool comprising a physical storage device
SSM_LocationInfo	User-supplied geographical location information
SSM_Fan	Fans
SSM_CPU	CPUs and other controllers such as GPUs
SSM_Memory	Memory
SSM_PowerSupply	Power supplies
SSM_Board	Circuit boards, motherboards, blades and cards
SSM_Enclosure	Enclosures such as controller cases, racks, disk shelves

4.5.1. Instance Diagram for Physical Hardware Support

Figure 6 shows the instance diagram for physical hardware support. The GetAttachedDevices( ) and GetContainedDevices( ) method calls of the Enclosure class return Devices of any of the five indicated types, filtering according to the DeviceTypes parameter. Either Embedded Instances or ObjectPaths can be returned, based on client request.

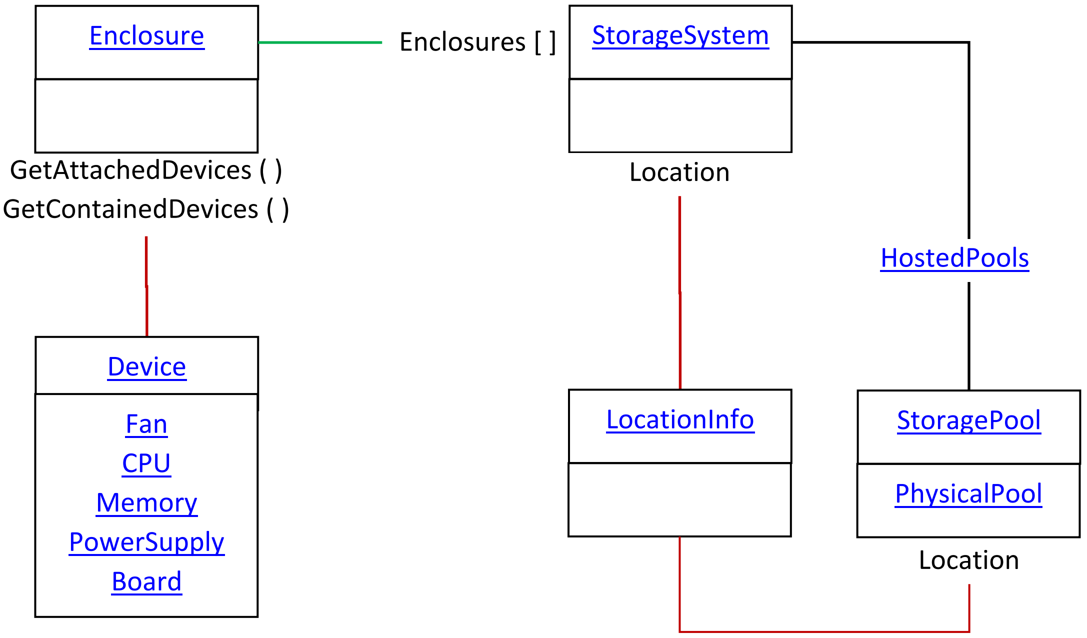

Figure 6 **Physical Hardware Instance Diagram**

[4] Physical hardware support is not mandatory in SSM. This is to accommodate virtual systems.

4.6. Required classes for block services support

The classes listed in Table 7   are mandatory if block services are supported[5]. All properties and methods are mandatory unless otherwise indicated via the Optional or Conditional qualifiers. Additionally, the StorageServer.SupportedServices array shall contain the string "Block Services".

Table 7 Required classes for block services

SSM_StorageViewInfo	Mapping and masking info
SSM_EndpointInfo	Endpoints in a View
SSM_HostedVolume	Association from a server to its volumes
SSM_StorageVolume	A storage volume addressable by LBA

4.6.1. Block Services Instance Diagram

Figure 7 contains an Instance Diagram for Block Services support. Masking and mapping is done simply, via the MapStorageVolume( ) and UnmapStorageVolume( ) methods. SSM does not attempt to manage zoning—this is at present the domain of the Fabric Book in SMI-S. See the StorageSystem class for more information on the class methods.

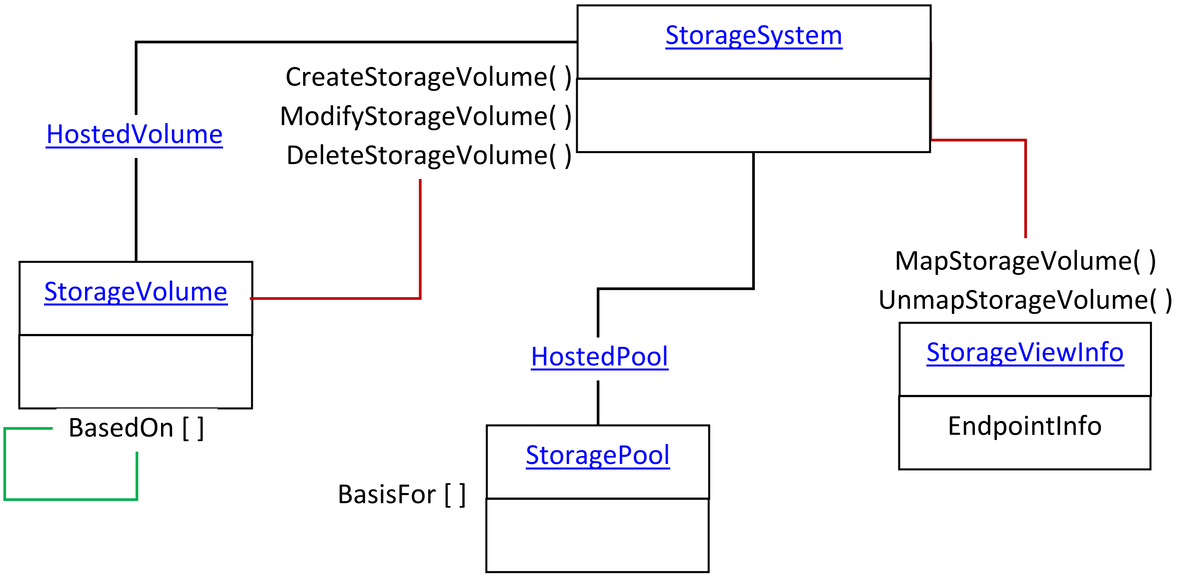

Figure 7 **Block Services Instance Diagram**

[5] Block support is not mandatory, but it is the only data access protocol supported in SSM v1.0.

4.7. Features planned for future releases

As previously stated, SSM 1.0 is purposely being kept small, with block support only, to present as small a barrier to entry as possible. However, a number of additional feature sets have preliminary designs completed, and will be introduced in future releases. All of these except role management will be optional to implement, according to the services implementors desire to offer.

4.7.1. Role management

A simplified form of role management is described in sections 3.4.3 and 3.4.4. It is optional in this release, but is planned to become mandatory in a future release of this specification. This will involve support for ACL-based authorization on systems and pools at a minimum. It may also possibly include support for ACL-based authorization on volumes, filesystems, object stores and shares, depending on development and performance impacts. Another desired feature is to place ACLs on shares for use by host-side filesystems.

4.7.2. File Services

Creation and management of file servers, filesystems and shares comprise the File Services feature set. If Performance Management is supported, there will likely be performance-related requirements on file servers.

4.7.3. Object Stores

Creation and management of object stores will be quite simple (before snorting, please recall that hope springs eternal). The goal is to create an object store such as CDMI or Hadoop, attach it to some storage, and get out of the way. This feature set is not yet fully designed. If Performance Management is supported, there will likely be performance-related requirements on object stores.

4.7.4. Replication Services

Replication is a complex but essential part of any real storage solution. The current design calls for a simplified form of SMI-S Replication, with copies, moves and snapshots broken out as separate operations. A ReplicationManager class—a peer of WBEMServer_Server in a sense, as it manages multiple systems—forms the core of the functionality. It contains many service methods to achieve setting up, pausing, fracturing, restoring and so on, for various types of replication.

4.7.5. Performance Management

Performance management is a complex discipline, requiring floods of data to manage properly. This feature set enables reporting of a comprehensive set of data in a CSV format for easy import into various established performance analysis applications. Clients can select which data they wish to monitor, to minimize performance impact on the base system.

5. Detailed class information (normative)

As part of the simplification effort, the authors have attempted to reduce the number and size of source documents, especially the number of source documents that anyone might consider to be normative. To that end, there are three document formats used in the source documents for this specification:

- Microsoft Word
- Machine-readable class declaration files in DMTF MOF format
- XML for the Standard Message Registry

The class information in this section is provided in HTML format, and is automatically generated from the SSM MOF files. All MOF files shall be considered normative for the purposes of this specification. In cases when the MOF and other portions of this specification diverge or conflict, the MOF shall be considered authoritative.

5.1. SSM Class Listing

This class listing also appears in a file named "SSM_Index.html", bundled with the html file set that accompanies the distribution. For convenience, the class listing is duplicated in Table 8 .

Table 8 SSM Class listing

Class	Description
SSM_Association	The parent class for all SSM model associations
SSM_BasedOnInfo	A structure used in BasedOn[] listings to describe the elements a higher-level element is based on, and the amount of space allocated from each of the underlying elements
SSM_Board	A circuit board, motherboard or bus-attached card on a physical system
SSM_CIMElement	A class allowing SSM objects to refer to CIM_ schema objects
SSM_CPU	A CPU on a physical system
SSM_Enclosure	An enclosure on a physical system
SSM_EndpointInfo	Information about an endpoint in a network
SSM_Fan	A fan on a physical system

SSM_HostedPool	An associator for the top-level pools on a system
SSM_HostedVolume	An associator for the volumes on a system
WBEM_InitiativeRegistration	The WBEM Management Initiative name and version supported
SSM_InitiativeSupported	SSMInitiative - InitiativeSupported sub class
SSM_KVInfo	A Key/Value pair
SSM_LocationInfo	A structure containing information about the location of a storage element
SSM_Memory	A physical memory device on a physical system
SSM_OperationalStatusInfo	A structure containing an OperationalStatus string and its human-readable description
SSM_PhysicalPool	A disk, SSD, tape cartridge, NVDIMM, etc
SSM_PowerSupply	A power supply on a physical system
SSM_ProvenanceInfo	A structure containing information about one "stage" in the provenance of a device
SSM_RedundancyInfo	A structure containing information about the redundancy inherent in a set of devices
SSM_SpaceInfo	A structure used for reporting space usage on a container
SSM_StoragePool	A StoragePool combines elements from CIM_StoragePool and CIM_StorageExtent
SSM_StorageSetting	A StorageSetting instance describes a canned (predefined) setting
SSM_StorageSettings	An associator for the canned (predefined) settings supported by a StorageSystem
SSM_StorageSystem	The top-level system object that is discovered via SLPv2, or any of its component systems in the case of an HA or clustered

	system configuration
SSM_StorageSystemJobs	An association for jobs running in the system
SSM_StorageViewInfo	In SCSI, views are collections of masking and mapping paths
SSM_StorageVolume	A StorageVolume is a byte-addressable container, usually called a 'LUN' in the storage industry
SSM_ViewItemInfo	In SCSI, views are collections of masking and mapping endpoint pairs
WBEM_Device	The top-level class for all WBEM Device types
WBEM_Error	WBEM_Error contains information about an error
WBEM_Indication	WBEM Indication representation
WBEM_InitiativeRegistration	The WBEM Management Initiative name and version supported
WBEM_InitiativeSupported	An association from the InitiativeRegistration to the central class of the WBEM Management Initiative
WBEM_Job	A job, normally created when a method call may take longer than an HTTP timeout
WBEM_KeyedObject	Abstract base class for all Keyed Objects
WBEM_Object	Abstract base class for all Objects in A WBEM Management Initiative
WBEM_MethodParameterInfo	Information about the method call that caused a Job to be created
WBEMServer_Account	An account represents a local account for a WBEM Server used for authentication
WBEMServer_AccountSettings	Account settings
WBEMServer_IndicationDestination	IndicationDestination provides the information needed to deliver indications to a WBEM Listener

WBEMServer_IndicationSettings	WBEM Server Indication Capabilities and Settings
WBEMServer_IndicationSubscription	IndicationSubscription provides the subscription information
WBEMServer_InitiativeSupported	WBEM Server Initiative - InitiativeSupported sub class
WBEMServer_ProtocolConfiguration	WBEM_ProtocolConfiguration is used to configure a protocol entry point for the WBEM server
WBEMServer_ProtocolSettings	WBEM Server protocol settings/capabilities for a specified protocol
WBEMServer_Server	Server represents the WBEM Server infrastructure

The remainder of this clause contains one subsection for each class. These listings have been transformed from MOF text to be more readable. With the exception of HTML markup embedded in the MOF text, the listings contain the MOF text word for word. Please report any discrepancies between the MOF text and this document via the SNIA Feedback Portal at

http://www.snia.org/feedback/

5.2. SSM_AccountInfo

An administrative account on the system. SSM defines three levels of user:

1. Security Officer. This user is only empowered to create and manage security for other users.
2. Administrator ("admin"). An administrator sets up pools and assigns them to Operators using the Pools list.
3. Operator. An operator manages pools and allocates volumes, filesystems and object stores from them, and manages those via mapping, shares etc.

Similarly to settings, Operator differs from Info objects in having an InstanceID. Only Security Officers shall be permitted to create, delete and modify the Type, Name and Email in Operator accounts. Admins shall be permitted to modify the Pools property and no other.

class **SSM_AccountInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **AccountType**	Values { "SecurityOfficer", "Administrator", "Operator" } The account type of this user.
string **AccountName**	The name the account is known by on this system. This is the account name the user uses to log into the CIMOM, and shall not be changeable.
string **AccountEmail**	Optional Email address, as entered when the account was created or changed.
string **OwnedEntities** [ ]	Reference (SSM_Device) Pools, volumes, filesystems and object stores this account has Operator privilege on. This property shall be NULL for Security Officer accounts and shall be limited to pools for Administrator accounts.

5.3. SSM_ACEInfo

An ACE object conformant to the ACL structure used in Windows, NFSv4 and CDMI. ACLs are attached to file shares and object store containers.

class **SSM_ACEInfo**	Experimental Indication Structure Version (2.xx.0)
string **ACEType**	Values { "ALLOW", "DENY" } Each ACE ALLOWs or DENYs access via one or more Operations to one user or group.
string **Who**	The user or group that this ACE pertains to. Note that user and group names are opaque to SSM. It is the responsibility of the client to ensure that the user and group names in the ACE are meaningful in the context of the domain from which the share or container is being accessed.
string **WhoType**	Values { "User", "Group" } Whether the entity named in Who is a user or a group.
string **Operations** [ ]	Values { "READ_OBJECT", "LIST_CONTAINER", "WRITE_OBJECT", "ADD_OBJECT", "APPEND_DATA", "ADD_SUBCONTAINER", "READ_METADATA", "WRITE_METADATA", "EXECUTE", "DELETE_OBJECT", "DELETE_SUBCONTAINER", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES", "WRITE_RETENTION", "WRITE_RETENTION_HOLD", "DELETE", "READ_ACL", "WRITE_ACL", "WRITE_OWNER", "SYNCHRONIZE", "ALL" } The operation(s) to allow or deny. As with Who, these values are opaque to the server and are enforced by the host accessing the object the ACL is attached to. Multiple operations may be specified at once.

5.4. SSM_ACLInfo

An ACL object conformant to the ACL structure used in Windows, NFSv4 and CDMI. ACLs are attached to file shares and object store containers. RFC 3530 or the SNIA CDMI Specification may be consulted for further reference.

class **SSM_ACLInfo**	Experimental Indication Structure Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion" }
string **ACEs** [ ]	Reference (SSM_ACEInfo) An array of ACEInfo entries ("ACEs"). Each entry sets ALLOW or DENY access to one user or group. Entries are evaluated in order until permission to use the attached container is explicitly granted or denied. If at the end of evaluation no permission has been explicitly granted, permission is denied. While SSM provides a way to place and view ACLs, there is no implied mechanism for enforcement. Enforcement is strictly the responsibility of the host-side access protocol (SMB, NFS etc.)

5.5. SSM_Association

The parent class for all SSM model associations.

class **SSM_Association**	Association Abstract Version (2.xx.0)
SSM_Object REF **Source**	Key The source member of the association. Some CIM association classes call this the Antecedent.
SSM_Object REF **Target**	Key The target member of the association. Some CIM association classes call this the Dependent.

5.6. SSM_Board

A circuit board, motherboard or bus-attached card on a physical system.

Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing belwether events from Enclosures or StorageSystems.

class **SSM_Board** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
sint32 **Power**	Current board or card power in milliwatts. If the device does not support reporting this, enter -1.
string **BoardType** [ ]	Values { "Unknown", "Other", "Multi-purpose", "HBA", "NIC", "Video", "Sound", "Memory", "Serial", "Compute", "Special-purpose" } Textual representation of the board or card type, for display by the client to an admin. Multi-purpose boards shall list Multi-purpose as the first entry in the array, with other entries following.
string **AttachType**	Values { "Unknown", "Other", "PCIe", "USB", "Infiniband", "Wireless", "BlueTooth" } Attach type of the board.

string **AttachVersion**	Version of the AttachType. E.g. for USB either 1.0, 2.0 or 3.0 should be entered. For Wireless, use 802.11g etc. If a version is unknown or inapplicable, this string shall be empty.
string **BoardDescription**	Optional Freeform text containing additional descriptive material, for display by the client to an admin.
sint8 **HotPluggable**	Values { "-1", "0", "1" } Whether the board is hot-pluggable. If the device does not support reporting this, enter -1. Otherwise, zero shall mean False and one shall mean True.
string **Memory**	Conditional ('Memory' in BoardType) Reference (SSM_Memory) In the event that this board is a memory card, the object path to the SSM representation of the memory. When the condition is not true, this property shall be NULL.

5.7. SSM_CIMElement

A class allowing SSM objects to refer to CIM_ schema objects. This is meant to allow clients to navigate to the larger CIM model for more detailed management options than the SSM schema offers.

class **SSM_CIMElement** : SSM_Object	Experimental Version (2.xx.0)
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **ManagedElement**	Reference (CIM_ManagedElement) A reference to an object in the CIM schema. It shall contain a valid ObjectPath in the provider's namespace.

5.8. SSM_CPU

A CPU on a physical system.

Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing belwether events from Boards, Enclosures or StorageSystems.

class **SSM_CPU** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
sint32 **Power**	Current CPU power usage in milliwatts. If the device does not support reporting this, enter -1.
string **ProcessorSpeed**	Free-form representation of CPU speed, for display by a client. Providers should format this as a number followed by GHz or MHz, e.g. "3.42 GHz". If the device does not support reporting this, enter -1.
sint8 **Cores**	The number of cores. If the device does not support reporting this, enter -1.
sint8 **HardwareThreads**	Total number of hardware threads. If the device does not support reporting this, enter -1.

5.9. SSM_Device

The top-level class for all SSM Device types. A valid InstanceID is mandatory for all objects of such types.

class **SSM_Device** : SSM_Object	Abstract Experimental Version (2.xx.0)
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema. Top-level devices that *are* the scoping system shall populate this property with NULL.
string **DurableName**	A persistent vendor-defined name for the object. This property replaces the welter of DeviceID, Tag, Name and other properties in the CIM schema. The rules regarding durable names given in clause 7 of the SMI-S Architecture book shall be followed with the following further restrictions: • SCSI Logical Unit names shall be in VPD page 83 type 3 (NAA) format • Fibre Channel ports shall use Port World Wide Names (i.e., FC Name_Identifier) • iSCSI targets shall use a dotted IP Address concatenated with a colon and a TCP port number, e.g. "10.1.219.5:9988". Clause 7 mentions "other types of devices" with specific guidance. This book interprets this to mean "other types of ManagedElements". As SSM objects are not ManagedElements, such sections of clause 7 are deemed not applicable to Devices.
string **DurableNameFormat**	Values { "Text", "VPD83type3" } A description of what the DurableName signifies. Subclasses may override this to indicate other formats. The default shall be Text.
string **GivenName**	A name given by the user. Vendors should populate this initially with the DurableName or with an empty string. After being set by the client, the GivenName shall have the same persistence properties as the DurableName, except that a client may change it at any time.

5.10. SSM_Enclosure

An enclosure on a physical system.

NOTE: for all extrinsic methods of this class, a return value of zero shall mean successful completion or creation of a job to complete the method call. In methods that furnish a Job parameter, the Job shall be NULL if no job was created, and populated with a REF to the job if one was created. All other outcomes shall cause a return value of non-zero and inclusion of an error message from the message registry in the header of the return message.

Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing bellwether events from StorageSystems.

class **SSM_Enclosure** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
string **Location**	EmbeddedInstance (SSM_LocationInfo) Description of the part's physical location.
sint32 **Power**	Enclosure power in milliwatts. This shall include the power of all devices contained in the enclosure. If the enclosure does not support reporting this, enter -1.
boolean	Get the devices physically in the enclosure. Disks shall be represented as

GetContainedDevices ( )	PhysicalPools. Memory, CPUs, fans, boards and cards all have Device classes representing them. The method shall return True on successful completion or creation of a job, and False with an SSM_Message error otherwise.
boolean **GetEmbeddedInfo**	Optional IN Whether to get device embedded instance info. The default is to only obtain object paths.
string **DeviceTypes** [ ]	Optional IN Values { "Other", "Enclosure", "Fan", "CPU", "Board", "Power Supply", "Memory", "PhysicalPool", "All" } The type of device(s) to get. The default shall be All.
string **ContainedDevices** [ ]	OUT Conditional (GetEmbeddedInfo = True) EmbeddedInstance (SSM_Device) The returned array of device embedded instances, if requested. If GetEmbeddedInfo is False, this array shall not be returned.
string **ContainedDevicePaths** [ ]	OUT Conditional (GetEmbeddedInfo = False) Reference (SSM_Device) The returned array of device object paths, if requested. If GetEmbeddedInfo is True, this array shall not be returned.
SSM_Job **Job**	OUT A job created to do the work. If no job is needed, this parameter shall be NULL.
boolean **GetAttachedDevices ( )**	Get the devices that are attached to the system on the "back end". These can be disk shelves, i.e. other enclosures, fans, or any other physical element type indicated by the ElementType input paramter. The method shall return True on successful completion or creation of a job, and False with an SSM_Error otherwise.
boolean **GetEmbeddedInfo**	Optional IN Whether to get device embedded instance info. The default is to only obtain object paths.
string	Optional

DeviceTypes [ ]	IN Values { "Other", "Enclosure", "Fan", "CPU", "Board", "Power Supply", "Memory", "PhysicalPool", "All" } The type of device(s) to get. The default shall be All.
string **AttachedDevices** [ ]	OUT Conditional (GetEmbeddedInfo = True) EmbeddedInstance (SSM_Device) The returned array of device embedded instances, if requested. If GetEmbeddedInfo is False, this array shall not be returned.
string **AttachedDevicePaths** [ ]	OUT Conditional (GetEmbeddedInfo = False) Reference (SSM_Device) The returned array of device object paths, if requested. If GetEmbeddedInfo is True, this array shall not be returned.
SSM_Job **Job**	OUT A job created to do the work. If no job is needed, this parameter shall be NULL.

5.11. SSM_EndpointInfo

Information about a single endpoint in a network

class **SSM_EndpointInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **EndpointType**	Values { "iSCSI", "HTTP", "HTTPS", "FCoE", "FC", "Other", "Unknown" } The type of endpoint. Vendor specific endpoint types shall be prefixed with a vendor symbol such as a stock ticker symbol.
string **HostID**	Conditional ( ) The InstanceID of the host system.
string **IPAddress**	Conditional (WWNodeName = NULL) The system endpoint IP address, if any, in dotted decimal format. E.g. 10.1.242.19 .
uint32 **Port**	Conditional (IPAddress != NULL) The port number
string **MACAddress**	Optional The MAC address, if any, in colon-separated format. E.g. AC:81:75:4F:01:00 .
string **NodeWWN**	Conditional (IPAddress = NULL) The WWN node name, if any, in VPD83Type3 (NAA) format.
string **PortWWN**	Conditional (IPAddress = NULL) The port WWN of the endpoint, if any, in VPD83Type3 (NAA) format.
string **iSCSIid**	Conditional (EndpointType = 'iSCSI') The iSCSI identifier of the endpoint. This may be in either IQN or EUI format. See RFC 3721 and RFC 3722 for specifications of this format. Briefly, all IQN identifiers begin with "iqn" and all EUI identifiers begin with "eui".

5.12. SSM_Fan

A fan on a physical system.

Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing belwether events from Enclosures or StorageSystems.

class **SSM_Fan** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
string **Enclosure**	Reference (SSM_Enclosure) Enclosure the fan is in.
uint8 **Number**	Number of the fan in the enclosure, beginning with 1.
sint32 **CurrentRPM**	Current fan speed. If the device does not support reporting this, enter -1.
sint32 **MaxRPM**	Maximum fan speed. If the device does not support reporting this, enter -1.
sint32 **Power**	Current power draw in milli-amps. If the device does not support reporting this, enter -1.

sint8 **HotPluggable**	Values { "-1", "0", "1" } Whether the fan is hot-pluggable. If the device does not support reporting this, enter -1. Otherwise, zero shall mean False and one shall mean True.

5.13. SSM_FileShare

A FileSystem represents a filesystem mounted on underlying storage, able to export shares to clients. A HostedFileShare association shall exist between every FileShare and the hosting StorageSystem. On clustered systems this will generally not be the top level scoping system, but a component system.

class **SSM_FileShare** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **Path**	Path name of the exported ("shared") location on the server. This shall include the server name as required for mounting. E.g. for CIFS/SMB: "\\Server\Path", for NFS: "Server:/Path"
string **Access** [ ]	Values { "None", "Unknown", "Read", "Write", "Write Once", "Append", "Streaming" } Uses of the share will be restricted to these access modes. The share ACL may add further restrictions. Values may be ORed together, by placing both strings in the array, as follows: • { Read, Write } • { Read, Append } • { Read, Write Once } • { Streaming, Read } • { Streaming, Write } • { Streaming, Append } • { Streaming, Read, Write } • { Streaming, Read, Append } Append shall mean that writes may only be appended to an open file. Other combinations should not be used (the Streaming variants may only work with LTFS).

	Servers are not required to support every possible mode, and should return a NOT_SUPPORTED error if a client attempts to create a share with an unsupported Access type.
string **ACL**	EmbeddedInstance (SSM_ACLInfo) The ACL placed on the share, if any. A NULL value shall indicate that no ACL has been placed on the share.
string **BasedOn** [ ]	Reference (SSM_FileSystem) Underlying filesystem[s] that the share is based on.
sint32 **AddACE ( )**	Construct an ACE and add it to the share's ACL, creating the ACL if necessary. In consonance with SMB standard practice, ALLOW ACEs shall always be kept by this routine at the front of the list and DENY ACEs in the rear. There are some limited use cases for other arrangements ("allow everyone from group x except user y"), but clients that wish to achieve this must manipulate the ACL and set it directly, without the help of this convenience routine.
string **ACEType**	Values { "ALLOW", "DENY" } Whether to allow or deny the listed Operations.
string **Who**	The user or group in question.
string **WhoType**	Values { "User", "Group" } Whether Who is a user or a group.
string **Operations** [ ]	Values { "READ_OBJECT", "LIST_CONTAINER", "WRITE_OBJECT", "ADD_OBJECT", "APPEND_DATA", "ADD_SUBCONTAINER", "READ_METADATA", "WRITE_METADATA", "EXECUTE", "DELETE_OBJECT", "DELETE_SUBCONTAINER", "READ_ATTRIBUTES", "WRITE_ATTRIBUTES", "WRITE_RETENTION", "WRITE_RETENTION_HOLD", "DELETE", "READ_ACL", "WRITE_ACL", "WRITE_OWNER", "SYNCHRONIZE", "ALL" } Operations to be allowed or denied. These are essentially opaque to the implementation, and are used by the file access protocol engine when accessing the share.
sint32 **RemoveACE ( )**	Remove an ACE from an ACL. As it illegal to remove ACEs that don't exist, this method assumes that the ACL has been accessed and that the ACEs are available as EmbeddedInstances. To remove an ACE, find the one you want to remove and call this method with the ACE's EmbeddedInstance.

string **ACE**	EmbeddedInstance (SSM_ACEInfo) The ACE to remove.

5.14. SSM_FileSystem

A FileSystem represents a filesystem mounted on underlying storage, able to export shares to clients. A HostedFilesystem association shall exist between every FileSystem and the hosting StorageSystem. On clustered systems this will generally not be the top level scoping system, but a component system.

There are no extrinsic methods on this class; all FileSystem management is done by the StorageSystem object that owns the filesystem.

class **SSM_FileSystem** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **CIMFileSystem**	Reference (CIM_FileSystem) The filesystem as represented in the CIM_ schema. If CIM objects are unavailable or not implemented, this property shall contain NULL
string **Root**	Path name defining the root of the filesystem on the server. This shall include the server name as required for mounting. Concatenating the Root and the Name properties in the way specified by a given access protocol shall yield a string that may be used by a client to mount the filesystem using that access protocol. E.g. CIFS/SMB: "\Root\Name" NFS: "/Root/Name"
string **Name**	The name of the filesystem as used for mounting. See the Root property for semantics.
string **ObjectInfo** [ ]	EmbeddedInstance (SSM_StorageObjectInfo) An array of object type info. Each entry contains the name of the type, such as Inode or

	File, the maximum supported size of an object of that type, the maximum supported number of objects of that type, and the current number of objects of that type. ObjectInfo shall not be empty. Servers shall report at least Bytes. See FSObjectInfo for more detail.
string **Access** [ ]	Values { "None", "Unknown", "Read", "Write", "Write Once", "Append", "Streaming" } Mounts of the filesystem will be restricted to these access modes. The mount itself may add further restrictions. Values may be ORed together, by placing both strings in the array, as follows: • { Read, Write } • { Read, Append } • { Read, Write Once } • { Streaming, Read } • { Streaming, Write } • { Streaming, Append } • { Streaming, Read, Write } • { Streaming, Read, Append } Append shall mean that writes may only be appended to an open file. Other combinations should not be used (the Streaming variants may only work with LTFS). Servers are not required to support every possible mode, and should return a NOT_SUPPORTED error if a client attempts to create a share with an unsupported Access type.
uint64 **BlockSize**	FileSystems read and write data in blocks which are defined independently of the underlying storage. This property captures the FileSystem's block size for data storage and retrieval, in bytes.
string **SpaceInfo**	EmbeddedInstance (SSM_SpaceInfo) Allocation and usage data.
string **Usage** [ ]	Values { "Other", "Available", "Reserved", "In Use", "Reserved by Replication Services", "Reserved by Migration Services", "Remote Replica Target", "Remote Replica Source", "Local Replica Target", "Local Replica Source", "Delta Replica Target", "Delta Replica Source" } Indicates the intended usage or any restrictions that may have been imposed on the usage of this component. Multiple values may be supplied. Vendors should include 'Reserved' for any filesystem that may not be mounted except by the reserving entity. Vendors should include 'In Use' for filesystems that are exported. Otherwise, vendors should include 'Available' in the list.
string **BasedOn** [ ]	Reference (SSM_Device)

	Underlying system objects that the filesystem is based on. Normally one or more pools.
uint8 **SnapshotUsed**	The percentage of total space in the filesystem that is used for snapshots. This should be the actual space used, not the amount of space set aside for a snapshot reserve. If snapshot data is kept elsewhere on the system, say in a separate container or on system storage, this number shall be zero (0).

5.15. CIM_FSObjectInfo

A structure containing information about various filesystem objects.

class **CIM_FSObjectInfo** : CIM_Object	Experimental Version (2.xx.0)
string **ObjectType**	Read Values { "Inodes", "Files", "Directories", "Links", "Devices", "Files/directories", "Blocks", "K-V objects", "UID objects" } An object type supported by the FileSystem. Some systems do not report files and directories separately; these systems should use 'Files/directories'.
uint64 **NumberOfObjectsMax**	The maximum number of objects of type ObjectType that this FileSystem can contain. A value of 0 shall mean that there is no specific maximum associated with the object type.
uint64 **ObjectSizeMax**	The maximum size of objects of type ObjectType that this filesystem can contain. A value of 0 shall mean that there is no specific maximum associated with the object type.
uint64 **NumberOfObjects**	The number of objects of type ObjectType that this FileSystem contains. Given the rapid change in a loaded server, this number should be considered a best effort calculation.

5.16. SSM_HostedFileShare

An associator for the exported filesystem shares on a system. Note that the associations to hosted objects follow a different rule from the CIM_ schema. In the CIM_ schema, HostedFileShares are associated to the top-level system, under the rules of the MultipleComputerSystem profile. In the SSM model, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. To list all the file shares hosted on a top-level storage system, traverse to the "bottom" of the StorageSystem hierarchy using the ComponentCS property, and aggregate the hosted file shares from each of the component systems.

Security considerations: An empty set of associations shall be returned to a Security Officer. Only associations to file shares managed by the Operator shall be returned to an Operator.

class **SSM_HostedFileShare** : SSM_Association	Experimental Association Version (2.xx.0) Conditional ("File Services" in SSM_StorageSystem.SupportedServices)
SSM_StorageSystem REF **Source**	Key Override (Source) The storage system hosting the FileShare.
SSM_FileShare REF **Target**	Key Override (Target) The FileShare being hosted.

5.17. SSM_HostedFileSystem

An associator for the top-level filesystems on a system. Note that the associations to hosted objects follow a different rule from the CIM_ schema. In the CIM_ schema, HostedFileSystems are associated to the top-level system, under the rules of the MultipleComputerSystem profile. In the SSM model, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. To list all the filesystems hosted on a top-level storage system, traverse to the "bottom" of the StorageSystem hierarchy using the ComponentCS property, and aggregate the hosted filesystems from each of the component systems.

Security considerations: An empty set of associations shall be returned to a Security Officer. Only associations to filesystems managed by the Operator shall be returned to an Operator.

class **SSM_HostedFileSystem** : SSM_Association	Experimental Association Version (2.xx.0) Conditional ("File Services" in SSM_StorageSystem.SupportedServices)
SSM_StorageSystem REF **Source**	Key Override (Source) The storage system hosting the FileSystem.
SSM_FileSystem REF **Target**	Key Override (Target) The FileSystem being hosted.

5.18. SSM_HostedObjectStore

An associator for the object stores on a system. Note that the associations to hosted objects follow a different rule from the CIM_ schema. In the CIM_ schema, HostedObjectStores (if there were such a thing) would be associated to the top-level system, under the rules of the MultipleComputerSystem profile. In the SSM model, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. To list all the stores hosted on a top-level storage system, traverse to the "bottom" of the StorageSystem hierarchy using the ComponentCS property, and aggregate the hosted stores from each of the component systems.

Security considerations: An empty set of associations shall be returned to a Security Officer. Only associations to object stores managed by the Operator shall be returned to an Operator.

class **SSM_HostedObjectStore** : SSM_Association	Experimental Association Version (2.xx.0) Conditional ("Object Store" in SSM_StorageSystem.SupportedServices)
SSM_StorageSystem REF **Source**	Key Override (Source) The storage system hosting the store.
SSM_ObjectStore REF **Target**	Key Override (Target) The object store being hosted.

5.19. SSM_HostedPool

An associator for the top-level pools on a system. Pools may be aggregated to any depth. This association shall only return the highest level pools, and no intermediate pools, nor Basic pools that have not yet been aggregated. Use GetBasicPools on the server to find Basic pools. The server may or may not allow creation of LUNs or other containers on Basic or intermediate pools, but any such practice is discouraged.

Note that the associations to hosted objects follow a different rule from the CIM_ schema. In the CIM_ schema, HostedPools are associated to the top-level system, under the rules of the MultipleComputerSystem profile. In the SSM model, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. To list all the pools hosted on a top-level storage system, traverse to the "bottom" of the StorageSystem hierarchy using the ComponentCS property, and aggregate the hosted pools from each of the component systems.

Security considerations: An empty set of associations shall be returned to a Security Officer. Only associations to pools managed by the Operator shall be returned to an Operator.

class **SSM_HostedPool** : SSM_Association	Experimental Association Version (2.xx.0)
SSM_StorageSystem REF **Source**	Key Override (Source) The storage system hosting the pool.
SSM_StoragePool REF **Target**	Key Override (Target) The pool being hosted.

5.20. SSM_HostedVolume

An associator for the volumes on a system. Note that the associations to hosted objects follow a different rule from the CIM_ schema. In the CIM_ schema, HostedVolumes are associated to the top-level system, under the rules of the MultipleComputerSystem profile. In the SSM model, hosted entities are associated to the StorageSystems that allocate or manage them. This change is necessary for security reasons in highly virtualized environments. To list all the volumes hosted on a top-level storage system, traverse to the "bottom" of the StorageSystem hierarchy using the ComponentCS property, and aggregate the hosted volumes from each of the component systems.

Security considerations: An empty set of associations shall be returned to a Security Officer. Only associations to volumes managed by the Operator shall be returned to an Operator.

class **SSM_HostedVolume** : SSM_Association	Experimental Association Version (2.xx.0) Conditional ("Block Services" in SSM_StorageSystem.SupportedServices)
SSM_StorageSystem REF **Source**	Key Override (Source) The system containing the volumes.
SSM_StorageVolume REF **Target**	Key Override (Target) A volume.

5.21. SSM_Job

A job, normally created when a method call may take longer than an HTTP timeout. A JOB_COMPLETED indication (an SSM_Message) is sent on completion. If completion is successful, clients may then examine the job to verify and consume the output parameters of the method. Providers may delete jobs ten minutes after successful completion. Providers shall not delete jobs that complete with errors for 36 hours, except as directed by the StorageSystem method DeleteJob().

When a job is created, an association instance of StorageSystemJobs, pointing to the job, shall be returned upon appropriate invocation of the Associators() intrinsic.

Implementations shall support the InstDeletion lifecycle indication and SSMM_JOB_COMPLETED alert. Support for other indications and alerts is optional. InstDeletion indications shall not be sent in the presence of encompassing bellwether events from the StorageSystem.

class **SSM_Job** : SSM_Object	Experimental Version (2.xx.0) RequiredIndications { "InstDeletion", "Alert" }
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
uint64 **JobID**	An integer ID for the job. Implementations shall monotonically increase the ID number for each successive job. This number shall be used to form the InstanceID.
string **Status**	Values { "Completed", "New", "In process", "Error", "Terminated", "Suspended" } Job status.
string **PercentComplete**	Percentage completed in the C format "2271928%".
string **Error**	EmbeddedInstance (SSM_Message) A CIM error in the case that an error has occurred during processing of the job. If no error has occurred, this property shall be NULL.
string **MethodName**	The name of the method whose invocation caused the job to be created.
string **MethodInputParameters** [ ]	EmbeddedInstance (SSM_MethodParameterInfo)

	The input arguments to the method parameters.
string **MethodOutputParameters** [ ]	EmbeddedInstance (SSM_MethodParameterInfo) The output values for the method parameters. These may not be available until the job is completed.
datetime **TimeOfExit**	The time of completion, error, or termination of the job, in CIM datetime format.

5.22. SSM_KVInfo

A Key/Value pair. Arrays of "KV pairs" are useful in many situations. Implementations may sort them for any purpose, but clients shall not assume that this has been done unless specifically advised to that effect.

class **SSM_KVInfo** : SSM_Object	Experimental Version (2.xx.0)
string **Key**	The key for the pair. This is NOT a CIM key. It is a key specific to the array the pair is contained in. No structure for the key is implied by SSM.
string **Value**	The value component of the pair. No structure is implied.

5.23. SSM_ListenerDestinationInfo

Info about Indication listeners. Each of the arrays in this structure comprises a list of listeners to one of the indication types defined in the SSM model. Indications are subscribed to using the SubscribeToIndication method of the relevant StorageServer.

"Listeners" in these arrays shall be valid URIs to which indications shall be sent. It is not specified in SSM whether UDP or TCP should be used to send the messages. It is permissible to use multicast on all or portions of these lists.

class **SSM_ListenerDestinationInfo** : SSM_Object	Experimental Indication Structure Version (2.xx.0)
string **InstCreationListeners** [ ]	Listener destination URIs for InstCreation indications.
string **InstDeletionListeners** [ ]	Listener destination URIs for InstDeletion indications.
string **InstModificationListeners** [ ]	Listener destination URIs for InstModification indications.
string **AlertListeners** [ ]	Listener destination URIs for Alerts.

5.24. SSM_LocationInfo

A structure containing information about the location of a storage element. All properties of this structure are mandatory to implement, but may legally be NULL.

class **SSM_LocationInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **Country**	The ISO 3166-1 alpha-2 ASCII country code or ISO 3166-1 numeric country code of the country in which the part is installed
string **Territory**	Name of the territory in the country. Not all countries use this in addresses. India and China do.
string **State**	Name of the state in the country or territory.
string **City**	Name of the town or city in which the part is installed.
string **Address1**	Freeform address info for the location.
string **Address2**	Additional freeform address info for the location.
string **Address3**	Additional freeform address info for the location.
string **PostalCode**	Postal code (or "zip" code)
string **Building**	Name of the building in which the part is installed.
string **Room**	Name or number of the room in which the part is installed.
string **Row**	Row name or number in which the part is installed.
string **Rack**	Rack name or number. If Row is specified, this should be the rack number within the row, otherwise it may have a more global meaning.

string **Shelf**	Shelf or unit name or number. If Rack is specified, this should be the rack number within the rack, otherwise it may have a more global meaning.
string **Item**	Item position. If Shelf is specified, this should be the rack number within the shelf, otherwise it may have a more global meaning.
string **GPSCoords**	The GPS coordinates of the part. If furnished, this shall be expressed in the format '[-][nn]n.nnnnnn, [-][nn]n.nnnnn', i.e. two numbers, either positive or negative, with six decimal places of precision, comma-separated.
string **OtherLocationInfo**	Other freeform text describing the item's location.

5.25. SSM_ManagedSystems

An association from a WBEM Server to its managed systems.

class **SSM_ManagedSystems** : SSM_Association	Experimental Association Abstract Version (2.xx.0)
SSM_WBEMServer REF **Source**	Override (Source) The WBEM server
SSM_StorageSystem REF **Target**	Override (Target) The target member of the association. Some CIM association classes call this the Dependent.

5.26. SSM_Memory

A physical memory device on a physical system. An instance of this class should be reported for each DIMM or memory card.

Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing belwether events from Enclosures or StorageSystems.

class **SSM_Memory** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
sint32 **Power**	Memory power in milli-amps. If the device does not support reporting this, enter -1.
string **MemorySpeed**	Free-form representation of Memory speed and type, for display by a client. Providers should format this like "%d MHz %s" (or GHz), e.g. "667 MHz DDR". If the device does not support reporting this, enter "Unknown".
sint64 **MemorySize**	Memory unit size in bytes. If the device does not support reporting this, enter -1.
boolean **Persistent**	Whether the memory is transient (False) or persistent, like NAND flash (True).
sint8 **HotPluggable**	Values { "-1", "0", "1" }

	Whether the memory unit is hot-pluggable. If the device does not support reporting this, enter -1. Otherwise, zero shall mean False and one shall mean True.

5.27. SSM_Message

A simplified class for all forms of messaging, including errors, alerts and indications.

class **SSM_Message**	Indication Exception Experimental Version (2.xx.0)
string **Owner**	The organization that owns the registry that contains the message in question. I.e. DMTF, SNIA etc. When issuing a message from the SSM message registry, this property shall be set to "SSM".
string **MessageID**	Unique message ID within the context of the message owner. The SSM standard message registry defines a textual string identifier for each error and message type. See the section on Standard Messages in the SSM Book.
datetime **TimeStamp**	Time of creation of the message.
string **Originator**	Optional Reference (SSM_Device) Originator of the message. If the originator is the same object as the Sender, this property may be omitted or NULL.
string **Sender**	Reference (SSM_Device) Sender of the message. This shall be either a StorageSystem or a WBEMServer.
string **What**	Values { "Error", "Alert", "Indication" } Whether the message is an error, alert or indication. One of the three shall be supplied.
string **Severity**	Values { "N/A", "Unknown", "Debug", "Informative", "Warning", "Minor", "Major", "Critical", "Fatal" } Severity level of an error or alert that the message contains. If the message is not in regard to an error or alert condition, the provider shall populate this property with N/A. Unknown. The Perceived Severity of the condition is unknown or indeterminate. Debug. This severity level should only be used for debugging messages. These are not expected to be meaningful to end users.

	Informative. An informative message that requires no action. Clients may choose to filter out messages that are this level and below, under advisement that job completion messages are at this level. Warning. Should be used when it's appropriate to let the user decide if action is needed. Minor. Should be used to indicate action is needed, but the situation is not serious at this time. Major. Should be used to indicate action is needed NOW. Critical. Should be used to indicate action is needed NOW and the scope is broad (perhaps an imminent outage to a critical resource). Fatal. Should be used to indicate a non-recoverable error occurred; it's too late to take remedial action, but time to notify all applicable personnel.
boolean **Bellwether**	Whether this alert or indication is a bellwether. Bellwethers indicate that an event on a given system element has occurred, and may have caused similar events on contained or attached elements, but that these subsidiary events are being suppressed by the system in order to prevent alert storms. For example, a system power failure may cause hundreds or thousands of logical system elements to go offline; this fact can be inferred without processing an alert for every one of said logical elements.
string **OriginatingObject**	Reference (SSM_Object) The ObjectPath of the originating element for the message.
string **Message**	The main body string for the message. This functions similarly to the format string in C's printf() routine, but uses named placeholders. These placeholders are listed in the standard message registry (see the Standard Messages section in the SSM book), for standard messages. Vendor-defined messages shall use placeholders of the format '', i.e. a placeholder name inside of an < and >pair. Each said placeholder shall have a corresponding entry in the Args array.
string **Args** [ ]	Reference (SSM_KVPairInfo) Arguments to the message. These may be substituted for the placeholders in the Message string. This property may be NULL if the Message string does not contain any placeholders. Otherwise it shall contain a key/value pair with a key matching a placeholder, for each placeholder in the Message.
string **RecommendedActions**	Optional Recommended actions to take, if the message indicates a problem. Vendors should populate this string with free-form text for display to users, to help users figure out what to do to fix an issue.

5.28. SSM_MethodParameterInfo

When a method call is expected to take longer than a HTTP timeout, a Job is created to track the progress of the method. The input and output parameters of the method are tracked using objects of this class.

class **SSM_MethodParameterInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **ParameterName**	The name of the method parameter.
string **ParameterType**	Values { "sint8", "uint8", "sint16", "uint16", "sint32", "uint32", "sint64", "uint64", "boolean", "string", "real32", "real64", "char", "datetime", "REF", "Reference", "EmbeddedInstance" } The CIM type of the parameter. See DSP0004.
string **SchemaType**	Conditional (ParameterType in {"Reference", "EmbeddedInstance"}) The type of the ParameterValue string if it is not a native CIM type. If the ParameterType is Reference, this shall indicate that ParameterValue contains an ObjectPath to an object of the CIM or SSM class type given here. If the parameter type is EmbeddedInstance, this shall indicate that ParameterValue contains an embedded instance of the class given here.
string **ParameterValue**	The parameter value encoded as a string.

5.29. SSM_Object

A top-level class for all SSM types. This allows arbitrary collections of objects. It also enables lightweight listing of things, like lines from a log, that don't require InstanceIDs. Unfortunately, because of CIM v2 rules, it cannot be the parent class of SSM_Association, which is therefore the other top level class in the model.

class **SSM_Object**	Experimental Abstract Version (2.xx.0)

5.30. SSM_ObjectStore

Placeholder for an object storage element. Some properties referred to from elsewhere are listed.

class **SSM_ObjectStore** : SSM_Device	Experimental Abstract Version (2.xx.0)
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
uint8 **SnapshotUsed**	The percentage of total space in the device that is used for snapshots. This should be the actual space used, not the amount of space set aside for a snapshot reserve. If snapshot data is kept elsewhere on the system, say in a separate container or on system storage, this number shall be zero (0).
string **SpaceInfo**	EmbeddedInstance (SSM_SpaceInfo) Allocation and usage data.

5.31. SSM_OperationalStatusInfo

A structure containing an OperationalStatus string and its human-readable description.

class **SSM_OperationalStatusInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **Status**	Values { "Unknown", "Other", "OK", "Degraded", "Stressed", "Predictive Failure", "Error", "Non-Recoverable Error", "Starting", "Stopping", "Stopped", "Suspended", "In Service", "No Contact", "Lost Communication", "Aborted", "Dormant", "Supporting Entity in Error", "Completed", "Power Mode", "Relocating" } The OperationalStatus string. Classes that use this structure should indicate which of the values listed are acceptable.
string **Description**	The human-readable description for the string. This string can and should vary depending on the type and possible status conditions of the element in question.

5.32. SSM_AccountInfo

SSM defines three levels of user.

1. Security Officer. This user is only empowered to create and manage security for other users.
2. Administrator ("admin"). An administrator sets up pools and assigns them to Operators.
3. Operator. An operator manages pools and allocates volumes, filesystems and object stores from them, and manages those via mapping, shares etc.

See the security section in the SSM Book for additional normative detail on this and other security-related classes.

Implementations shall support the following alerts:

1. SSMM_PASSWORD_CHANGED
2. SSMM_LOGIN
3. SSMM_LOGOUT
4. SSMM_ACCOUNT_CREATED
5. SSMM_ACCOUNT_DELETED

Account creation and deletion shall also trigger InstCreation and InstDeletion lifecycle indications.

class **SSM_AccountInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **AccountType**	Values { "SecurityOfficer", "Administrator", "Operator" } The account type of this user.
string **AccountName**	Account name on this system. This name shall not be changeable.
string **AccountEmail**	Optional Email address, as entered when the account was created or changed.
string **OwnedEnties** [ ]	OUT Reference (SSM_Device) Pools, volumes, filesystems and object stores this account has Operator privilege on. This property shall be NULL for Security Officer accounts, and shall contain only pools for Administrator accounts.

5.33. SSM_PerfDataInfo

Performance statistics for various element types. Statistics are cumulative over time; clients that want periodic values must save the prior values and do subtraction to obtain them.

Performance statistics are problematic for some systems. For all `uint64` and `real32` properties below, if the element being reported on does not support a given property, it shall be reported as zero (0). Vendors are encouraged to develop the capability to report all of these properties, however, as this list has been vetted by storage performance specialists.

AGY notes: need a "manifest" filter (use a set method in StorageSystem?) Need to return an error for unsupported manifest items. Need a get method in StorageSystem.See table 128 Summary of Statistics Support by Element in BSP.

class **SSM_PerfDataInfo**	Experimental Indication Structure Version (2.xx.0)
string **ElementType**	Values { "StorageSystem", "Network Endpoint", "Volume", "FileSystem", "ObjectStore", "StoragePool", "PhysicalPool", "File", "Object", "Directory", "Container" } The type of element this statistic is for.
string **ElementID**	The InstanceID of the element this statistic is for.
datetime **StatisticTime**	The time at which the data for the element this statistic is for was last updated, in CIM v2.2 format. Implementations that do not do detailed time logging should report the time at which the statistic is generated for return.
datetime **StatisticBeginTime**	Optional For counters that report averages, the time at which data collection for this statistic began. This property is mandatory for any such counter.
uint64 **TotalIOs**	Optional The total cumulative IOs done by the element. If it is a block device, these are traditional block-level IOs. If it is a FileSystem, directory or file, these shall be filesystem requests fulfilled. If it an ObjectStore, container or object, this quantity shall be the count of all HTTP operations (or equivalent) to the store. If it is a network endpoint, these shall be the number of packets transferred. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for TotalIOs in the ElementType.

uint64 **KiBytesTransferred**	Optional The total number of Kibibytes (1024 bytes) transferred by the element. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for KiBytesTransferred in the ElementType.
uint64 **IOTimeCounter**	Optional The total amount of processing time used by the element to actively perform I/O. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for IOTimeCounter in the ElementType.
uint64 **ReadIOs**	Optional The total cumulative number of Read IOs done by the element. As with TotalIOs, these are either block-level reads, filesystem read requests, or object read requests. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for ReadIOs in the ElementType.
uint64 **ReadHitIOs**	Optional The total cumulative number of Read IOs that were satisfied out of cache. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for ReadHitIOs in the ElementType.
uint64 **ReadIOTimeCounter**	Optional The amount of processing time used to service the ReadIOs reported. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for ReadIOTimeCounter in the ElementType.
uint64 **ReadHitIOTimeCounter**	Optional The amount of processing time used to service the ReadIO requests that were satisfied out of cache. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for ReadHitIOTimeCounter in the ElementType.
uint64 **KiBytesRead**	Optional

	The number of Kibibytes (1024 bytes) transferred to service read requests. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for KiBytesRead in the ElementType.
uint64 **WriteIOs**	Optional The total cumulative number of write IOs. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for WriteIOs in the ElementType.
uint64 **WriteHitIOs**	Optional The total cumulative number of Write IOs that were already in cache. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for WriteHitIOs in the ElementType.
uint64 **WriteIOTimeCounter**	Optional The amount of processing time used to provide the Write IOs reported. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for WriteIOTimeCounter in the ElementType.
uint64 **WriteHitIOTimeCounter**	Optional The amount of processing time used to service the Write IO Hits reported. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for WriteHitIOTimeCounter in the ElementType.
uint64 **KiBytesWritten**	Optional The number of Kibibytes (1024 bytes) transferred to service write requests. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for KiBytesWritten in the ElementType.
uint64 **IdleTimeCounter**	Optional The amount of processing time, in seconds, during which the element has been "idle", i.e. not servicing any host I/O requests. Background processing shall count as idle time for the purposes of this statistic. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for

	IdleTimeCounter in the ElementType.
uint64 **MaintOp**	Optional The total cumulative number of maintenance operations (vendor defined) done by the element. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaintOp in the ElementType.
uint64 **MaintTimeCounter**	Optional The total amount of time used by the element to perform maintenance operations (as defined by the vendor). This should ordinarily be less than the IdleTimeCounter. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaintTimeCounter in the ElementType.
real32 **MaxWatts**	Optional The maximum watts used by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaxWatts in the ElementType.
real32 **MinWatts**	Optional The minimum watts used by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MinWatts in the ElementType.
real32 **InstWatts**	Optional The instantaneous wattage reading for the element at the time the statistic was prepared. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for InstWatts in the ElementType.
real32 **AvgWatts**	Optional The average watts used by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for AvgWatts in the ElementType.

real32 **MaxPF**	Optional The maximum power factor encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaxPF in the ElementType.
real32 **MinPF**	Optional The minimum power facter encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MinPF in the ElementType.
real32 **InstPF**	Optional The intantaneous power factor reading at the time the statistic was prepared. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for InstPF in the ElementType.
real32 **AvgPF**	Optional The average power factor encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for AvgPF in the ElementType.
real32 **MaxVolts**	Optional The maximum volts measured by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaxVolts in the ElementType.
real32 **MinVolts**	Optional The minimum volts measured by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MinVolts in the ElementType.
real32 **InstVolts**	Optional

	The intantaneous voltage reading at the time the statistic was prepared. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for InstVolts in the ElementType.
real32 **AvgVolts**	Optional The average volts measured by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for AvgVolts in the ElementType.
real32 **MaxAmps**	Optional The maximum amps drawn by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaxAmps in the ElementType.
real32 **MinAmps**	Optional The minimum amps drawn by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MinAmps in the ElementType.
real32 **InstAmps**	Optional The intantaneous current draw in amps at the time the statistic was prepared. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for InstAmps in the ElementType.
real32 **AvgAmps**	Optional The average amps drawn by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for AvgAmps in the ElementType.
real32 **MaxTHD**	Optional The maximum total harmonic distortion encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MaxTHD in the ElementType.

real32 **MinTHD**	Optional The minimum total harmonic distortion encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for MinTHD in the ElementType.
real32 **InstTHD**	Optional The intantaneous total harmonic distortion reading at the time the statistic was prepared. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for InstTHD in the ElementType.
real32 **AvgTHD**	Optional The average total harmonic distortion encountered by the element during the reported PowerTime interval. This item is mandatory if StorageSystem.SupportedPerfCounters contains an entry for AvgTHD in the ElementType.

5.34. SSM_PhysicalPool

A disk, SSD, tape cartridge, NVDIMM, etc. In the SSM Model, a pool that represents a single physical entity shall be of type PhysicalPool. Clients can determine which pools are physical by inspecting the type of the instance. Note that physical pools may contain pools that are not physical, as in the case of partitions, which are not physical according to this model, on a disk drive, which is.

Implementations shall support the ELEMENT_OFFLINE, ELEMENT_FAILED and ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing bellwether events from Enclosures or StorageSystems.

class **SSM_PhysicalPool** : SSM_StoragePool	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **CIMStoragePool**	The pool as represented in the CIM_ schema.
string **OperationalStatus** [ ]	An array of structures describing the current state of the pool.
string **Setting**	The canned setting used to create this pool.
string **PoolType**	Pools are either Basic or aggregated to form larger pools.
string **Access**	Whether the pool is readable, writeable, etc.
uint64 **BlockSize**	Block size used for I/O, in bytes.

string **SpaceInfo**	Allocation and usage data for the pool
string **BasedOn** [ ]	Pools this pool is based on.
string **BasisFor** [ ]	Pools, volumes, filesystems and object stores that are BasedOn this pool.
string **CIMPhysicalPool**	Reference (CIM_ManagedElement) The pool device as represented in the CIM_ schema. If CIM objects are unavailable or not implemented, this property shall contain NULL. In the common use case that the PhysicalPool is a disk drive of some sort, implementations should pass an ObjectPath to a CIM_DiskDrive when possible.
string **ProvenanceInfo**	Conditional (MediaType in {"HDD", "SSD", "IO Bus Attach", "Hybrid"}) EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, Model, etc.
string **Location**	EmbeddedInstance (SSM_LocationInfo) Description of the part's physical location.
string **SecureEraseMethods** [ ]	Values { "Set Password", "Zero Fill", "Multi-pass" } Secure erasure methods supported by the device. Set Password is how self-encrypting drives (SEDs) do cryptographic erasure. Zero Fill refers to simple "clearing" (in ISO 27040 parlance). Multi-pass refers to any method of writing multiple passes of 0's, 1's and random data. This property may be NULL or empty if the device does not support secure erase.
string **GetSupportedSizes ( )**	Values { "Completed", "Error" } Return a set of supported discrete pool sizes. StoragePools that support a set of discrete volume sizes should support this method to return that set.
string **ElementType**	IN Values { "Storage Pool", "Storage Volume", "Logical Disk", "Thin Provisioned Volume", "Thin Provisioned Logical Disk" } The type of element for which sizes are requested. If thin provisioned types are requested but not supported, the provider shall return an INVALID_ELEMENT_TYPE message in the return header. StoragePools that support a range instead of discrete sizes should return a USE_GET_SUPPORTED_SIZE_RANGE message in the return

	header. Note that due to other client-side activity, the actual sizes may change between the return of this method and invocation of a method to create or modify an element based on this information.
uint64 **Sizes** [ ]	OUT List of supported sizes, in bytes, for element creation or modification.
string **GetSupportedSizeRange ( )**	Values { "Completed", "Error" } Get the supported range of pool sizes. StoragePools that support a range of volume sizes should support this method to return the limits for that range. StoragePools that only support discrete sizes should return a USE_GET_SUPPORTED_SIZES message in the return header. Note that due to other client- or host-side activity, the actual sizes may change between the return of this method and invocation of a method to create or modify an element based on this information.
string **ElementType**	IN Values { "Storage Pool", "Storage Volume", "Logical Disk", "Thin Provisioned Volume", "Thin Provisioned Logical Disk" } The type of element for which sizes are requested. If thin provisioned types are requested but not supported, the provider shall include a INVALID_ELEMENT_TYPE message in the return header.
uint64 **MinimumVolumeSize**	OUT The minimum size for a volume/pool in bytes.
uint64 **MaximumVolumeSize**	OUT The maximum size for a volume/pool in bytes.
uint64 **VolumeSizeQuantum**	OUT A volume/pool size, in bytes, must be a multiple of this value. Use 1 when any size between the min and max is acceptable.
string **GetReplicationInfo ( )**	Values { "Completed", "Job Created", "Error" } Get information about replication activity. StoragePools that support Replication services should support this method to return the replicas associated to them.
string	OUT

ReplicaInfo [ ]	EmbeddedInstance (SSM_ReplicaInfo) Information about each replica target or source for this storage element.
string **SetLEDState ( )**	Values { "Success", "Bad Parameter", "Vendor Reserved", "DMTF Reserved" } Set the LED state of the device. No query for supported states is furnished; clients may cycle through the available states noting which ones result in a Bad Parameter error. Providers shall populate the parameters of this routine with the current state of the LED upon completion of the routine. 'on' and 'off' should be used when the LED does not support more than two states.
string **Color**	IN OUT Values { "red", "green", "blue", "violet", "yellow", "white", "black", "on", "off" } Set the color of the LED. On and off should only be used in exclusion to the other colors, and shall appear in a color of the vendor's choice.
string **FlashState**	IN OUT Values { "on", "off", "very fast", "fast", "slow", "very slow" } Set the flashing speed of the LED.
string **SecureErase ( )**	Values { "Success", "Unsupported Method", "Unsupported Partition", "Other Error" } Cause secure erasure of the disk or partition. Use this method with extreme care, as it causes irreversible data loss. A return value of Success indicates successful deletion of data. Unsupported Method and Unsupported Partition return values shall not cause alterations to data. An Other Error makes no promises; data may or may not have been deleted, in full or in part.
string **MethodName**	IN The secure erase method to use. This shall be one of the strings found in SecureEraseMethods gotten via the instance.
sint32 **Partition**	IN The index of the partition to erase. If this parameter is NULL or is negative, the entire physical element shall be erased.

5.35. SSM_PowerSupply

A power supply on a physical system. Implementations shall support the SSMM_ELEMENT_OFFLINE, SSMM_ELEMENT_FAILED and SSMM_ELEMENT_ONLINE alerts. These shall not be sent in the presence of encompassing belwether events from Enclosures or StorageSystems.

class **SSM_PowerSupply** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
sint32 **Power**	The amount of power being drawn at the moment of query. If the device does not support reporting this, enter -1.
sint32 **RatedPower**	The rated power of the power supply. If the device does not support reporting this, enter -1.
sint8 **HotPluggable**	Values { "-1", "0", "1" } Whether the power supply is hot-pluggable. If the device does not support reporting this, enter -1. Otherwise, zero means False and one means True.

5.36. SSM_ProvenanceInfo

A structure containing information about the provenance of a device.

class **SSM_ProvenanceInfo** : SSM_Object	Experimental Version (2.xx.0)
string **Manufacturer** [ ]	The manufacturer's full name. For rebranded OEM parts, this may be the OEM's name or the branding entity's name, at the discretion of the vendor. If desired, multiple names may be disclosed. In such cases, the primary name, to be displayed by clients, shall be in the first entry in the array. On media where this information is not available, providers shall enter "Unknown".
string **PartNumber** [ ]	The part number(s) assigned by the manufacturer. For rebranded OEM parts, this may be the OEM's PN or the branding entity's PN, at the discretion of the vendor. If desired, multiple PNs may be disclosed. This property shall be indexed with the Manufacturer property value. On media where this information is not available, providers shall enter "Unknown".
string **SerialNumber** [ ]	Serial number(s) for the device. For rebranded OEM parts, this may be the OEM's SN or the branding entity's SN, at the discretion of the vendor. If desired, multiple names may be disclosed. This property shall be indexed with the Manufacturer property value. On media where this information is not available, providers shall enter "Unknown".
string **Model** [ ]	The model number of the part. For rebranded OEM parts, this may be the OEM's model no. or the branding entity's, at the discretion of the vendor. If desired, both model numbers may be disclosed. This property shall be indexed with the Manufacturer property value. On media where this information is not available, providers shall enter "Unknown".
string **FirmwareVersionString** [ ]	The version of the part's firmware. For rebranded OEM parts, this may be the OEM's version string or the branding entity's, at the discretion of the vendor. If desired, both version strings may be disclosed. This property value shall be indexed with the Manufacturer property value. On media where this information is not available, providers shall enter "Unknown".
string **MediaType**	Values { "Unknown", "N/A", "On Board", "HDD", "Mem Bus Attach", "IO Bus Attach", "Floppy Disk", "Optical Disk", "SSD", "Hybrid", "Tape" } The physical media type of the device, mandatory when it is a storage media device. Mem Bus Attach shall include NVDIMMs and like elements. IO Bus Attach shall include NVMe, DRAM cards and like elements. Non-media devices such as

	fans and CPUs shall report N/A.
boolean **IsFRU**	Whether the part is field-replaceable.

5.37. SSM_RedundancyInfo

A structure containing information about the redundancy inherent in a set of devices. PackageRedundancy refers to the number of physical elements that can fail without data loss. DataRedundancy refers to the number of complete copies of the stored data that are kept. These are defined in conformance with the definitions of PackageRedundancy and DataRedundancy elsewhere in the SMI-S specification.

class **SSM_RedundancyInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
uint8 **PackageRedundancy**	The number of physical elements that can fail without data loss. This includes one hot spare if there are any. So RAID 6 with one or more hot spares should report a PackageRedundancy of 3, as two drives in the RAID set and a hot spare can fail without data loss.
uint8 **DataRedundancy**	The number of complete copies of data that are kept in a set of storage elements. RAID 6 should report 1. RAID 1 should report 2. Systems such as Hadoop that use replication will generally report 3 or more. Admins should note that high values for DataRedundancy imply higher power usage.

5.38. SSM_ReplicaInfo

A structure containing information about replicas of a given storage element. This is the same information that is captured by the StorageSynchronized association in the CIM schema. In the SSM model, it is gotten via the GetReplicationInfo() method.

Note that CopyType and other relative relationships refer to the calling element as the 'source'.

class **SSM_ReplicaInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **SourceObject**	Reference (SSM_Device) ObjectPath to the source container. This property may be missing or NULL in the cases of a detached connection or when the target is a candidate container for replication of another entity.
string **SourceObjectType**	Values { "StoragePool", "StorageVolume", "FileSystem", "ObjectStore", "File", "Volume", "Object" } The object type the replica pertains to.
string **TargetObject**	Reference (SSM_Device) The ObjectPath to the target container.
string **TargetObjectType**	Values { "Unknown", "Pool", "Volume", "Filesystem", "File", "ObjectStore", "Object" } The container type of the replication target. This shall only be File, Volume or Object for targets that have already been created. I.e. when querying a system for candidates to host replication targets, only pools, filesystems and object stores shall be returned.
string **CopyType**	Values { "AsyncDeltaPIT", "AsnycDeltaCont", "AsyncFullPIT", "AsyncFullCont", "SyncDeltaPIT", "SnycDeltaCont", "SyncFullPIT", "SyncFullCont" } The type of replication. Values are taken from the 3-D matrix that represents the copy type: S x T x M, where • S = { asynchronous, synchronous }, • T = { delta, full copy }, • M = { PIT, continuous }

string **ConnectionState**	Values { "Waiting", "Error", "Updating", "Fracturing", "Fractured", "Resynching", "Resynched", "Suspending", "Suspended", "Resuming", "Resumed", "Detaching", "Detached", "Restoring", "Restored", "Restarting", "Restarted" } The state of the connection between source and target. • Fractured shall mean that the connection has been 'split', and that the target is available for use independently of the source. • Fracturing shall indicate that the connection is in transition to a Fractured state. • Resynched shall mean that a fractured connection has been restored, and the delta changes have been integrated. As this state is momentary and transitional, display of this state is at the discretion of the vendor. • Resynching shall indicate the process of progressing from the Fractured to the Resynched state. • Suspended shall have the same meaning as '\quiesced'\, i.e. the replication arrangement has been (presumably temporarily) suspended, and no writes to the target are allowed. • Suspending shall indicate the process of progressing into the Suspended state. Normally this means that new updates from the source are blocked, but that the target is continuing to ingest updates sent prior to command invocation. • Resumed shall mean that a Suspended connection has been re-opened for traffic. • Resuming shall indicate the process of progressing into the Resumed state. As this state is momentary and transitional, display of this state is at the discretion of the vendor. • Detached shall mean that the arrangement has been permanently terminated. • Detaching shall mean that the arrangement is in the process of being permanently terminated. New updates shall not be allowed, but the target may, depending on the system design, be still ingesting writes made before the command to terminate the relationship was given. • Restored shall mean that the source has been restored from the target. • Restoring shall mean that the source is in process of being restored from the target. • Restarted shall mean that a Restored relationship has been restarted in its original configuration and returned to a normal operational status. As this state is momentary and transitional, display of it is at the discretion of the vendor. • Restarting shall mean that the connection is being configured to its original state and returned to a normal operational status.
string **CopyPriority**	Optional Values { "Not Managed", "Low", "Same", "High", "Urgent" } Copy engine priority relative to other tasks. CopyPriority allows the priority of background copy engine I/O to be managed relative to host I/O operations during a background copy operation. Values are: • Low: copy engine I/O has lower priority than host I/O. • Same: copy engine I/O has the same priority as host I/O. • High: copy engine I/O has higher priority than host I/O.

	• Urgent: copy operation to be performed as soon as possible, regardless of the host I/O requests.

5.39. SSM_SliceInfo

A slice of a volume or file. Unsure that slices of pools, filesystems or object stores make sense.

class **SSM_SliceInfo** : SSM_Object	Experimental Version (2.xx.0)
string **ElementType**	Values { "Unknown", "Other", "File", "Volume" } The type of element
string **Element**	EmbeddedInstance (SSM_Device) The element
uint64 **FirstByte**	Beginning logical byte address (LbA) of the slice (zero-based).
uint64 **LastByte**	Ending logical byte address (LbA). For beginning LbA of M and ending LbA of N, N - M shall be the size of the slice, and attempts to access LbAs M - 1 or N + 1 shall be an error.

5.40. SSM_SpaceInfo

A structure used for reporting space usage on a container. A "container" is, for these purposes, any storage element--from a file up to a StorageSystem itself--that presents an amount of space that may be written. The base quantities are

- AvailableSpace
- UsedSpace
- PotentialSpace
- ProvisionedSpace
- SpaceLimit
- SnapshotReserve
- SnapshotUsed

The rules around space reporting are complex and confusing, especially in the presence of thin provisioning. Other complications include external snapshots (as opposed to in-container snapshots) and whether containers may be dynamically expanded. For convenience, these options are abbreviated here as TP, XX, SS, and DE, respectively. Some rules:

For systems without TP, XX, SS or DE

- ProvisionedSpace = LogicalLimit = SpaceLimit = PotentialSpace
- AvailableSpace = SpaceLimit - UsedSpace

For systems with TP only, and
For systems with TP and DE

- ProvisionedSpace = SpaceLimit
- AvailableSpace = MIN(available physical space, ProvisionedSpace)
- PotentialSpace = system determined

For systems with SS only

- ProvisionedSpace = SpaceLimit + SnapshotReserve
- PotentialSpace = AvailableSpace

For systems with XX only

- ProvisionedSpace = SpaceLimit
- PotentialSpace = AvailableSpace
- SnapshotReserve = space allocated external to the container

For systems with DE only, and
For systems with XX and DE

- ProvisionedSpace = SpaceLimit
- PotentialSpace = system determined

- AvailableSpace = SpaceLimit - UsedSpace

For systems with SS and TP

- ProvisionedSpace = SpaceLimit + SnapshotReserve
- PotentialSpace = system determined
- AvailableSpace = SpaceLimit - UsedSpace

For systems with XX and TP

- ProvisionedSpace = SpaceLimit
- PotentialSpace = system determined
- AvailableSpace = SpaceLimit - UsedSpace

For systems with SS and DE

- ProvisionedSpace = SpaceLimit + SnapshotReserve
- PotentialSpace = system determined
- AvailableSpace = SpaceLimit - UsedSpace

For systems with SS, DE and TP

- ProvisionedSpace = SpaceLimit + SnapshotReserve
- PotentialSpace = system determined
- AvailableSpace = MIN(available physical space, ProvisionedSpace)

For systems with XX, DE and TP

- ProvisionedSpace = SpaceLimit
- PotentialSpace = system determined
- AvailableSpace = MIN(available physical space, ProvisionedSpace)

The above rules do not apply to unbounded containers such as files or object stores. These use the following rules, regardless of the capabilities of the underlying storage

- ProvisionedSpace = UsedSpace
- SpaceLimit = PotentialSpace = AvailableSpace = system determined
- SnapshotReserve = 0, or system determined
- SnapshotUsed = 0, or system determined

class **SSM_SpaceInfo** : SSM_Object	Experimental Indication Structure Version (2.xx.0)

uint64 **AvailableSpace**	The total amount of space, in bytes, in the container, that has not already been written or otherwise reserved. That is, the amount of space that is available for writing new data or metadata. For containers with capacity limits set at the time of provisioning--typically volumes, some filesystems, pools on most systems, etc.--this quantity shall be the lesser of two other quantities: 1. The amount of unused physical storage available, and 2. The ProvisionedSpace. Other containers shall report the lesser of three different quantities: 1. The amount of unused physical storage available, and 2. The predetermined system limit for the container type, and 3. The PotentialSpace On some thin provisioning systems and for some container types, this value may change, even without any write activity, as additional storage is added to the system.
uint64 **PotentialSpace**	The potentially consumable space in the container, in bytes. That is, space nominally "available" for allocation of filesystems, volumes and object stores, or for writing of data to subsidiary containers. On thin provisioning systems, this can be much larger than either AvailableSpace or ProvisionedSpace, and should be set to the predetermined system limit for a container of the type this structure is attached to.
uint64 **ProvisionedSpace**	The total amount of space that has been provisioned, in bytes. This shall include any space set aside for reserves of any type IN the container. Externally allocated reserves shall not be included in this number; they should be reported in SnapshotReserve. On thin provisioning systems, this may total up to greater than the AvailableSpace, as it is the sum of the nominal sizes of the containers that have been allocated plus the space set aside for any reserves. On systems that do not do thin provisioning, this shall be less than or equal to AvailableSpace. Containers that are not space-provisioned per se, such as most object stores, shall report zero (0).
uint64 **SpaceLimit**	The nominal size of the container, that is, the amount of space specified at provisioning time and putatively available--assuming enough physical capacity--for writing before writes begin to fail.

uint64 **MetadataSpace**	Optional On systems that report it, the amount of space consumed by metadata, in bytes. E.g. inodes, dirents, dedup catalogs, RAID parity etc. This may be either physical or virtual space.
uint8 **AvailableSpaceRatio**	The ratio of AvailableSpace to SpaceLimit, expressed as a percentage.
uint8 **LowSpaceWarningThreshold** [ ]	The values of AvailableSpaceRatio below which action should be taken to add storage. This array may be set empty with StorageSystem.SetLowSpaceWarningThreshold. though that is strongly recommended against for StorageSystems and StoragePools when thin provisioning is present and active. In all cases, the last value in the array shall trigger a critical alert, and the penultimate value shall trigger an urgent alert (see SSM_Message). All other preceding values shall trigger informational alerts.
uint64 **SnapshotReserve**	The amount of space reserved IN the container for replica data. When the space used for replicas exceeds the reserve reported by the container's setting, this value shall be equal to SnapshotUsed, and the container setting's value shall remain unchanged. This number shall be zero if the system does not store snapshot data in the container this structure is attached to. Duh.
uint8 **SnapshotUsed**	The amount of total space in the container that is used for snapshots. This should be the actual space used, not the amount of space set aside for a snapshot reserve. If the system does not take snapshots at this container's level, or if snapshot data is kept elsewhere on the system, say in a separate container or on system storage, this number shall be zero (0).

5.41. SSM_StorageManager

A service for managing interactions between storage servers. We are not calling it a "Service" because CIM veterans expect a service to be something that can be started and stopped.

class **SSM_StorageManager** : SSM_Device	Experimental Version (2.xx.0)
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **SupportedMethods** [ ]	Values { "RelocateElement", "CopyElement", "CreateReplica", "ModifyReplica", "AttachReplica", "DetachReplica", "FractureReplica", "ResyncReplica", "SuspendReplica", "ResumeReplica", "RestoreReplica", "RestartReplica" } An array of methods on this system that are supported. The SSM Model does not explicitly distinguish between synchronous and asynchronous actions. Clients may learn whether an action is being performed synchronously or asynchronously by whether the Job parameter in the method call is populated upon return. This array shall not be empty. CompareFWVersionStrings is mandatory to implement. Note that methods related to operations on single systems are contained in the StorageSystem class. Check for SupportedMethods related to those operations on the StorageSystem.
boolean **RelocateStorageVolume ( )**	Conditional ("RelocateElement" in SupportedMethods) Attempt to relocate a container element from one pool to another. Successful relocation or creation of a job to do so shall return True. The Status field in this case shall contain either Completed or Job Started strings, respectively. If the volume is in use and cannot be relocated transparently (without detaching it), the return value shall be False with a Volume in Use status.All other results shall return False with the appropriate Status message, and shall mean that no job has been started, and no action taken.

	Successful completion of this routine or creation of a job to relocate the volume shall return True. All other results shall return False, shall mean that no job has been created and no action taken, and shall include one or more instances of SSM_Message in the return header describing the error.
string **TheVolume**	IN Reference (SSM_StorageVolume) The volume to relocate
string **NewPools** [ ]	IN Reference (SSM_StoragePool) The new pool to home the volume on.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **CopyElement ( )**	Conditional ("CopyElement" in SupportedMethods) Make a copy of an element to a new location. The type of source determines which destination types can accept it. • Pool: target must be a pool or a StorageSystem. If it is a pool, it must be large enough to contain the source. A new pool of the same size as the source shall be created on the target and the objects in the source copied onto the target. • Volume: target must be a pool or volume. In the case it is a volume, the target's contents shall be overwritten with the source's contents. The target volume must be large enough to completely contain the source data. If it is larger by N bytes, the last N bytes of the target's address space shall be zero-filled. In the case the target is a pool, there must be enough available space in the pool to completely contain a copy of the source volume, and a volume of the same size as the source shall be automatically created and filled. • Volume slice: target must be a slice. (see SSM_SliceInfo). It must be the same size as the source, and shall be overwritten with the source content. • File: target must be a file or directory. If the target is a file, its contents shall be overwritten with the contents of the source. • Directory: target must be a filesystem or directory. In the case the target is a directory, its contents shall be overwritten with the contents of the source. • FileSystem: target must be a pool. The pool must contain enough room. A new filesystem shall be created and the source filesystem's directory tree copied into it. Recommended error returns are COPY_FAILED, in conjunction with OBJECT_IN_USE, LOST_COMMUNICATION_WITH_SOURCE, INVALID_TARGET_TYPE, LOST_COMMUNICATION_WITH_TARGET, INVALID_SOURCE_TYPE, UNSUPPORTED_SOURCE_TYPE, UNSUPPORTED_TARGET_TYPE.

string **Source**	Reference (SSM_Object) The source element to be copied.
string **Target**	Reference (SSM_Object) The target element to be copied.
boolean **MoveElement ( )**	Conditional ("MoveElement" in SupportedMethods) Move an object to another location. SSM specifies this as a copy plus a delete of the source. However, implementations that are able to do space-efficient copies and deletes are free to do so. The rules for source and target are exactly the same as for CopyElement. The only differences between the two operations are that • A copy may fail if the target is in use • A move may fail if the source is in use and cannot be deleted. Implementations shall verify that the source may be successfully deleted before commencing the Recommended error returns are MOVE_FAILED, in conjunction with OBJECT_IN_USE, LOST_COMMUNICATION_WITH_SOURCE, INVALID_TARGET_TYPE, LOST_COMMUNICATION_WITH_TARGET, INVALID_SOURCE_TYPE, UNSUPPORTED_SOURCE_TYPE, UNSUPPORTED_TARGET_TYPE.
string **Source**	Reference (SSM_Object) The source element to be copied.
string **Target**	Reference (SSM_Object) The target element to be copied.
boolean **CreateReplica ( )**	Conditional ("CreateReplica" in SupportedMethods) Create a replica relationship. This shall include creation of the Target element if the Target is NULL. Otherwise the types of the Source and Target elements must match, or a return value of False with an INCOMPATIBLE_ARGUMENTS error shall be returned. Parameters for the replica relationship are given in the ReplicaInfo argument. An error in these shall cause a return of False with Invalid Argument status. In the common case that replica creation will take some time, a ConcreteJob should be started, and a REF to it passed out in Job. If the target element is successfully created, but the replication cannot be established, the method shall return False with Could not Create status. In this case the client may access the target element to delete it via the Target OUT parameter or TheElement property in the Job. Some arrays may not support arbitrary nesting of pools. These should return False with a Pool Nesting too Deep status when

	the Target is a pool and TargetPool is specified. Successful completion of this routine or creation of a job to create a replica shall return True. All other results shall return false, shall mean that no job has been created and no action taken, and shall include one or more instances of SSM_Message in the return header describing the error.
string **GivenName**	IN The user-defined name for the replica, if any.
SSM_Device **Source**	IN The source element for the replication.
SSM_Device **Target**	IN OUT The target element for the replication. If this is NULL, the System shall create a target element on the pool given in TargetPool, and return a REF to it. Otherwise the OUT value for this parameter is undefined.
SSM_StoragePool **TargetPool**	IN The pool to use for creating the target. If the Target is itself a pool, this shall result in a replica that is a pool hosted on the TargetPool.
string **ReplicationInfo**	IN EmbeddedInstance (SSM_ReplicaInfo) The replication parameters (see SSM_ReplicaInfo for details).
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **ModifyReplica ( )**	Conditional ("ModifyReplica" in SupportedMethods) The SSM Model does not support modification of replica types. Only the GivenName and CopyPriority may be changed. Use AttachReplica and DetachReplica to change the state of the replication relationship. Return values are intended to be self explanatory. If necessary and supported, a job may be created and returned in the Job parameter. Successful completion of this routine or creation of a job to modify the replica shall return True. All other results shall return False, shall mean that no job has been created and no action taken, and shall include one or more instances of SSM_Message in the return header describing the error.
SSM_StorageReplica	IN

Replica	The replica to be modified
string **GivenName**	IN A new GivenName for the replica.
string **CopyPriority**	IN Values { "Not Managed", "Low", "Same", "High", "Urgent" } A new CopyPriority for the replica.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **AttachReplica ( )**	Conditional ("AttachReplica" in SupportedMethods) Attach the source to the target replica and begin or resume synchronization, or start a job to do so. Successful attach or creation of a job to do so shall return True. The Status field in these cases shall contain either Completed or Job Started strings, respectively. Other outcomes shall return False with appropriate status strings. In particular, an attempt to attach a fractured replica on a system that does not support that should return an 'Operation not Supported' status. Successful completion of this routine or creation of a job to XXX shall return True. All other results shall return False, shall mean that no job has been created and no action taken, and shall include one or more instances of SSM_Message in the return header describing the error.
string **TheReplica**	IN EmbeddedInstance (SSM_StorageReplica) The replication endpoints to attach.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **DetachReplica ( )**	Conditional ("DetachReplica" in SupportedMethods) Detach a replica relationship in the manner specified. Successful detach or creation of a job to do so shall return True. The Status field in these cases shall contain either Completed or Job Started strings, respectively. Other outcomes shall return False with appropriate status strings. Successful completion of this routine or creation of a job to detach the replica shall return True. All other results shall return False, shall mean that no job has been created and no action taken, and shall include one or more instances of SSM_Message

	in the return header describing the error.
SSM_StorageReplica **TheReplica**	IN The replication relationship to detach.
string **DetachType**	IN Values { "Default", "Fracture", "Quiesce" } The type of detach operation to attempt. 'Default' shall leave this to the discretion of the system. 'Fracture' shall mean to break the connection and halt any queuing operations on the source. 'Quiesce' shall mean to temporarily stop copying operations, but to maintain state in preparation for subsequent reattach and resuming of the copy operations.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **SuspendReplica ( )**	Conditional ("SuspendReplica" in SupportedMethods) placeholder
boolean **ResumeReplica ( )**	Conditional ("ResumeReplica" in SupportedMethods) placeholder
boolean **FractureReplica ( )**	Conditional ("FractureReplica" in SupportedMethods) placeholder
boolean **ResyncReplica ( )**	Conditional ("ResyncReplica" in SupportedMethods) placeholder
boolean **RestoreReplica ( )**	Conditional ("RestoreReplica" in SupportedMethods) placeholder
boolean **RestartReplica ( )**	Conditional ("RestartReplica" in SupportedMethods) placeholder

5.42. SSM_StorageObjectInfo

A structure containing information about various filesystem objects. I.e. files, directories, inodes, etc.

class **SSM_StorageObjectInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **ObjectType**	Values { "Inodes", "Files", "Directories", "Links", "Soft Links", "Devices", "Files/Directories", "Blocks", "K-V objects", "Shortcuts", "OID objects", "Named objects", "Volumes", "FileSystems", "ObjectStores", "Bytes" } An object type supported by the system. As this structure is used in settings for all kinds of containers including pools, the container types are also represented. Some systems do not report files and directories separately; these systems should use Files/Directories. All container types (Volumes, FileSystems and ObjectStores) shall report Bytes.
uint64 **NumberOfObjectsMax**	The maximum number of objects of type ObjectType that the enclosing container can contain. A value of 0 shall mean that there is no specific maximum associated with the object type.
uint64 **ObjectSizeMax**	The maximum size of objects of type ObjectType that the enclosing container can contain. A value of 0 shall mean that there is no specific maximum associated with the object type.
uint64 **NumberOfObjects**	The number of objects of type ObjectType that the enclosing container contains. Given the rapid change in the number of some objects on a loaded server, this number, when provided, may be an ephemeral figure.

5.43. SSM_StoragePool

A StoragePool combines elements from CIM_StoragePool and CIM_StorageExtent. There are no extents in the SSM model. Volumes and Filesystems are allocated directly from pools. There may be a number of pool layers, such as disk partitions, disks, RAID groups, mirrors; however, only one is mandatory. This base layer of pools--usually drives or NVM structures, are termed Basic pools, and may be all that a JBOD exposes. Upper layers are called Aggregated pools. If a provider does not support the CreateStoragePool method, then only Basic pools are available. The CreateStoragePool method is used to "stack" Basic pools into larger groups. The layer describing the physical basis for the pool, e.g. disks, shall use the PhysicalPool subclass of this class.

class **SSM_StoragePool** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **CIMStoragePool**	Reference (CIM_StoragePool) The pool as represented in the CIM_ schema. If CIM objects are unavailable or not implemented, this property shall contain NULL
string **OperationalStatus** [ ]	EmbeddedInstance (SSM_OperationalStatusInfo) An array of structures describing the current state of the pool.
string **Setting**	Reference (SSM_StorageSetting) The canned setting used to create this pool.
string **PoolType**	Values { "Basic", "Aggregated" } Pools are either Basic or aggregated to form larger pools. For example, disk drives

	are often aggregated into RAID groups. A basic pool is the smallest unit of aggregation. Usually it is a disk, but it could even be a disk partition in some systems.
string **Access**	Values { "None", "Unknown", "Readable", "Writeable", "Read/Write", "Write Once", "Read/Write Once", "Append Only", "Read/Append Only", "Streaming" } Whether the pool is readable, writeable, etc.
uint64 **BlockSize**	Block size used for I/O, in bytes. If it is variable, return the largest size allowed. If it is unknown or the pool is byte-addressable, return 1.
string **SpaceInfo**	EmbeddedInstance (SSM_SpaceInfo) Allocation and usage data for the pool
string **BasedOn** [ ]	Reference (SSM_StoragePool) Pools this pool is based on. When a pool is layered ("stacked") on top of one or more other pools, this property shall contain the ObjectPaths of those pools. If NULL, this pool resides at the bottom of the pool hierarchy and PoolType shall be Basic.
string **BasisFor** [ ]	Reference (SSM_Device) Pools, volumes, filesystems and object stores that are BasedOn this pool. It is possible in some systems for higher-level containers to be based on multiple pools, so clients shall not assume that a container in this array is exclusively BasedOn this pool.
string **GetSupportedSizes ( )**	Values { "Completed", "Error" } Return a set of supported discrete pool sizes. StoragePools that support a set of discrete volume sizes should support this method to return that set.
string **ElementType**	IN Values { "Storage Pool", "Storage Volume", "Logical Disk", "Thin Provisioned Volume", "Thin Provisioned Logical Disk" } The type of element for which sizes are requested. If thin provisioned types are requested but not supported, the provider shall return an INVALID_ELEMENT_TYPE message in the return header. StoragePools that support a range instead of discrete sizes should return a USE_GET_SUPPORTED_SIZE_RANGE message in the return header. Note that due to other client-side activity, the actual sizes may change between the return of this method and invocation of a method to create or modify an element

	based on this information.
uint64 **Sizes** [ ]	OUT List of supported sizes, in bytes, for element creation or modification.

string **GetSupportedSizeRange ( )**	Values { "Completed", "Error" } Get the supported range of pool sizes. StoragePools that support a range of volume sizes should support this method to return the limits for that range. StoragePools that only support discrete sizes should return a USE_GET_SUPPORTED_SIZES message in the return header. Note that due to other client- or host-side activity, the actual sizes may change between the return of this method and invocation of a method to create or modify an element based on this information.
string **ElementType**	IN Values { "Storage Pool", "Storage Volume", "Logical Disk", "Thin Provisioned Volume", "Thin Provisioned Logical Disk" } The type of element for which sizes are requested. If thin provisioned types are requested but not supported, the provider shall include a INVALID_ELEMENT_TYPE message in the return header.
uint64 **MinimumVolumeSize**	OUT The minimum size for a volume/pool in bytes.
uint64 **MaximumVolumeSize**	OUT The maximum size for a volume/pool in bytes.
uint64 **VolumeSizeQuantum**	OUT A volume/pool size, in bytes, must be a multiple of this value. Use 1 when any size between the min and max is acceptable.

string **GetReplicationInfo ( )**	Values { "Completed", "Job Created", "Error" } Get information about replication activity. StoragePools that support Replication services should support this method to return the replicas associated to them.
string **ReplicaInfo** [ ]	OUT EmbeddedInstance (SSM_ReplicaInfo) Information about each replica target or source for this storage element.

5.44. SSM_StorageReplica

A replication relationship between two storage entities. The details are in the ReplicaInfo structure.

class **SSM_StorageReplica** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **Info**	EmbeddedInstance (SSM_ReplicaInfo) Replica-specific information.
string **Source**	Reference (SSM_StorageSystem) The source system in the replication relationship.
string **Target**	Reference (SSM_StorageSystem) The target system in the replication relationship.

5.45. SSM_StorageSetting

A StorageSetting instance describes a canned (predefined) setting. StorageSettings differ from Info structures in having a key.

StorageSettings shall be constructed in a vendor-specific way (perhaps via a config file) and made available by implementations for use by a StorageServer. At least one canned (pre-defined) setting shall be available at all times for each supported element from the set { StoragePool, StorageVolume, FileSystem, ObjectStore }.

Dynamic construction of settings is not supported in SSM, and client-constructed settings shall not be used as settings on storage objects. However, client-constructed settings may be used as templates for matching against when calling GetCannedSettings(), etc.

The GivenName property shall be modifiable with the ModifyInstance operation; all other attempted changes shall return an ILLEGAL_OPERATION error.

class **SSM_StorageSetting** : SSM_Object	Experimental Version (2.xx.0)
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **DurableName**	A vendor-supplied name for the canned setting. Recommended values are • Beryllium, Beryllium-DR • Platinum, Platinum-DR • Gold, Gold-DR • Silver, Silver-DR • Bronze Vendors may use any naming scheme they like, however. Clients shall not make QOS assumptions based on this DurableName. Other parameters must be examined to determine QOS levels offered by any one Setting.
string **GivenName**	A client-supplied name for the canned (pre-defined) setting. Implementations should populate this property with the DurableName until a client changes it using the intrinsic ModifyInstance.
string **ElementType**	Values { "StoragePool", "StorageVolume", "FileSystem", "ObjectStore", "StorageSystem" } The type of storage element supported by this setting. Clients shall not assume that all settings apply to element types other than the one given here.

string **SupportedAccessProtocols** [ ]	Values { "SMB", "CIFS", "SMB 2", "SMB 3", "NFSv3", "NFSv4", "NFSv4.1", "NVSv4.2", "AFP", "AFS", "SCSI", "SAS", "FC", "FCoE", "iSCSI", "CKD", "FICON", "ESCON" } Access protocols that an element supports. When names have no versions, it is assumed that auto-negotiation between the client and server will determine the actual protocol version to use. When a version is given, the same assumption holds, but the latest supported version is given in the protocol name.
uint8 **DeltaReservation**	The percentage, from 0 to 100, of space that is set aside in an element for replication purposes. For full copies, this is normally 100. For systems that use a snapshot reserve in the same element, this is the initial size of the snapshot reserve. If snapshot usage overflows the reserve, the actual percentage of space used by the snapshot (or other replication mechanism) shall be reported by the storage pool in .SnapshotUsed. Clients may change the DeltaReservation percentage using ChangeDeltaReservation(), but this should be done only in accordance with guidance from the StorageSystem vendor.
boolean **NoSinglePointOfFailure**	Also called NoSPOF. For this property to be true, only static elements such as backplanes may be non-redundant in the system(s) supporting the container. All other elements must be at least duplicated so that failure of any active element leaves a surviving active element to take the load.
string **RedundancyInfo** [ ]	EmbeddedInstance (SSM_RedundancyInfo) The number of physical elements, including at most one spare, that can fail without data loss. Also the number of complete copies of data kept on the container conforming to this setting. This data, known as PackageRedundancy and DataRedundancy, is provided in a structure as there may be multiple physical element storage types, such as disks, back end cloud stores, NVDIMMs and so on. Each should be reported separately. Example: RAID 6 with one hot spare has a package redundancy of three: two for the parity drives and one for the spare. RAID 5 and RAID 6 both should report 1 for data redundancy, and RAID 1 should report 2. Systems that use replication instead of RAID will generally report a data redundancy of 3 or more, bless their power-hungry little hearts.
string **ProvisioningType**	Values { "Full", "Thin" } The type of thin provisioning used by the pool. The default, when the property is NULL, shall be Full.
string **ProvisioningLimitType**	Values { "Unspecified", "Quota", "System determined", "Limitless" } The type of allocation limits supported by the pool. WORK NEEDED: Some explanation of what these things mean would be useful, Mr. Hadavi, so in the absence of that, the default, when the property is NULL, is Unspecified.

string **Encryption**	Values { "Unencrypted", "Encrypted", "SED" } Whether encryption is applied by the container. The default shall be Unencrypted. SED refers to the Trusted Computing Group (TCG) Self Encrypting Drive technology, which supports cryptographic erasure and does not expose the key outside of the drive.
string **EncryptionCipher**	The encryption type(s) used in the pool. Each string shall be of the form <cipher>-<key length>, or the special value N/A. Example strings are "AES-256" and "N/A".
string **CompressionType**	The compression method used in the pool, if any. This is a free form string for display by clients. The special value "None" shall be the default, and shall mean that no compression is being done. All other values shall indicate that compression is or will be operative.
string **ChangeDeltaReservation ( )**	Change the DeltaReservation percentage on a Setting. As implementations of snapshots and snapshot reserves vary dramatically from system to system, this should be done only in accordance with guidance from the specific system's vendor.
uint8 **NewValue**	Desired new value for the snapshot reserve. This shall be a value from 0 to 100. Note that depending on the implementation, this value may be strictly an advisory default.

5.46. SSM_StorageSettings

An associator for the canned settings supported by an OStorageSystem. Clients may use either this association or GetCannedSettings() to discover these settings. However, GetCannedSettings() may offer better filtering.

class **SSM_StorageSettings** : SSM_Association	Experimental Association Version (2.xx.0)
SSM_StorageSystem REF **Source**	Key Override (Source) The system containing the canned setting.
SSM_StorageSetting REF **Target**	Key Override (Target) The canned setting.

5.47. SSM_StorageSystem

The top-level system object that is discovered via SLPv2, or any of its component systems in the case of an HA or clustered system configuration.

By SSM Model rules, all vendor-specific values used in any class property or method parameters shall be prefixed with a stock ticker symbol or other unique vendor ID.

NOTA BENE: for all extrinsic methods of this class, a return value of zero shall mean successful completion or creation of a job to complete the method call. In methods that furnish a Job parameter, the Job shall be NULL if no job was created, and populated with a REF to the job if one was created. All other outcomes shall cause a return value of non-zero and inclusion of one or more error messages from the SSM message registry in the header of the return message (see the message registry section in the SSM book.

class **SSM_StorageSystem** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **SystemType**	Values { "Physical", "Virtual" } Whether the system is physical or virtual.
string **CIMStorageSystem**	Reference (CIM_ComputerSystem) The corresponding system in the CIM_ schema. If CIM objects are unavailable or not implemented, this property shall contain NULL
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, serial number, etc.

string **Enclosures** [ ]	Conditional (SystemType = "Physical") Reference (SSM_Enclosure) The enclosure for a physical system. There can be more than one of these, for instance a controller case in a system rack. The Enclosure class has methods for querying the other composed and attached devices on the system, such as fans, disk shelves, etc.
string **OperationalStatus** [ ]	Values { "OK", "Unknown", "Stressed", "Predictive Failure", "Servicing", "No Contact", "Lost Communication", "Starting", "Stopped", "Aborted", "Dormant", "Supporting Entity in Error" } The operational status of the system. Stressed indicates that the element is functioning, but needs attention. Examples of Stressed states are overload, overheating, and so on. Predictive Failure indicates that an element is functioning nominally but is predicting a failure in the near future. Servicing describes an element being configured, maintained, cleaned, or otherwise administered. No Contact indicates that the monitoring system has knowledge of this element, but has never been able to establish communications with it. Lost Communication indicates that the element has been contacted successfully in the past, but is currently unreachable. Starting means the machine is starting up and not yet available. Stopped and Aborted mean a clean and orderly stop and an abrupt stop that may indicate issues, respectively. Dormant indicates that the system is inactive or quiesced. Supporting Entity in Error indicates that this element might be OK but that another element, on which it is dependent, is in error. When this property is provided and there are multiple values, it is mandatory that the primary status of the system (usually OK) be the first array value.
string **ElementCausingError**	Conditional ("Supporting Entity in Error" in OperationalStatus) EmbeddedInstance (SSM_Object) When OperationalStatus contains Supporting Entity in Error, this property shall contain either the special string Unknown, or an embedded instance structure pointing to the element in error. This element may be either an Object or a ManagedElement wrapped in a CIMObject.
string **ParentSystems** [ ]	Reference (SSM_StorageSystem) In clustered, HA and other hierarchically arranged environments, this property indicates the 'parents' of the system represented by this object. For example, an HA pair is represented by three objects A, B and P, where P represents the overall system (the pair) and A and B represent the individual systems in the pair. A and B both populate this property with P. Top-level systems (such as P) shall populate this property with NULL. While this model permits multiple parents, all systems are required to have a single ParentSystem at the very top of the StorageSystem hierarchy, and that ParentSystem shall have the same InstanceID as is reported in SystemID. Parenthetical: use case for multiple parents. Imagine a large cluster, with top-

	level system P, physical cluster members C1..Cn, and a large virtual machine V serviced by some or all of C1..Cn. P declares C1..Cn as children (components), C1..Cn declare P as parent and V as child, and V declares C1..Cn as parents.
string **ComponentSystems** [ ]	Optional Reference (SSM_StorageSystem) In clustered, HA and other hierarchically arranged environments, this property indicates the 'children' of the system represented by this object. For example, an HA pair is represented by three objects A, B and P, where P represents the overall system (the pair) and A and B represent the individual systems in the pair. P shall populate this property with A and B, while they should omit it as they have no children. More complex configurations may result in servers with both parents and children.
string **ProvenanceInfo**	EmbeddedInstance (SSM_ProvenanceInfo) Manufacturer, model, serial number(s), etc.
string **SpaceInfo**	EmbeddedInstance (SSM_SpaceInfo) Space allocation and summary usage data.
string **Location**	Conditional (SystemType = "Physical") EmbeddedInstance (SSM_LocationInfo) Information about the physical location of the object.
string **SupportedReplicationTypes** [ ]	Conditional ("Replication" in SupportedElements) Values { "AsyncDeltaPIT", "AsnycDeltaCont", "AsyncFullPIT", "AsyncFullCont", "SyncDeltaPIT", "SnycDeltaCont", "SyncFullPIT", "SyncFullCont" } SupportedReplicationTypes describes the replication capabilities supported by this service. Values are taken from the 3-D matrix that represents the copy type: S x T x M, where • S = { asynchronous, synchronous }, • T = { delta, full copy }, • M = { PIT, continuous }
string **SupportedServices** [ ]	Values { "Block Services", "File Services", "Object Store", "Replication Services", "Performance Reporting" } Enumeration of the type of services provided by the system. This list is used for discovery; it is a high-level indication of system type. SupportedElements and SupportedMethods must be examined to determine whether the system

	supports the particular protocols or features that are wanted within a given service category. An empty array indicates a base system with no capabilities but discovery and pool management.
string **SupportedElements** [ ]	Values { "Physical System", "StoragePool", "StorageVolume", "FileSystem", "ObjectStore", "ThinlyProvisionedFileSystem", "ThinlyProvisionedStorageVolume", "ThinlyProvisionedPool", "ThinlyProvisionedObjectStore", "iSCSI", "FC", "FCoE", "SMB", "NFS", "AFS", "AFP", "CDMI", "S3" } Enumeration of the type of storage elements that are supported by the server. StoragePool, StorageVolume, FileSystem and ObjectStore shall indicate support for fully ("thickly") provisioned elements. Use the ThinlyProvisioned variants when thin provisioning of any type will be used on the element. To indicate support for block services, use one or more volume types and one or more of iSCSI, FC and FCoE. To indicate support for file services, use one or more filesystem types and one or more of AFS, SMB, NFS, AFP etc. To indicate support for object storage services, use one or more objectstore types, and one or more of CDMI and S3.
string **SupportedMethods** [ ]	Values { "SubscribeToIndication", "CompareFWVersionStrings", "SetLowSpaceWarningThresholds", "GetCannedSettings", "GetBasicPools", "GetMaxPoolSize", "CreateStoragePool", "ModifyStoragePool", "DissolveStoragePool", "EraseStoragePool", "CreateStorageVolume", "ModifyStorageVolume", "DeleteStorageVolume", "MapStorageVolume", "UnmapStorageVolume", "GetVolumeViews", "GetSupportedPerfCounters", "GetPerfData", "CreateSnapshot", "DeleteSnapshot", "CreateFileSystem", "DeleteFileSystem", "ModifyFileSystem" } An array of methods on this system that are supported. The SSM Model does not explicitly distinguish between synchronous and asynchronous actions. Clients may learn whether an action is being performed synchronously or asynchronously by whether the Job parameter in the method call is populated upon return. This array shall not be empty. CompareFWVersionStrings is mandatory to implement. Note that methods related to operations that may affect multiple systems, such as Copy, Move, and replication, are contained in the separate StorageManager class. Check for SupportedMethods related to those operations on the StorageManager.
string **SupportedPerfCounters** [ ]	EmbeddedInstance (SSM_SupportedPerfCounterInfo) An array of SupportedPerfCounter objects that indicate what type of performance data may be collected. See the SupportedPerfCounterInfo class for descriptions of and requirements for these objects.
string	EmbeddedInstance (SSM_ListenerDestinationInfo)

IndicationListeners [ ]	Indication listeners that have successfully subscribed to one or more indication types. The ListenerDestinationInfo structure contains an array of listeners for each of the defined indication types.
string **ManagementIPAddrs** [ ]	Management IP address(es) for the system, in dotted decimal form. This property shall contain only IP addresses that may be used to manage the system. In other words, data-only IP addresses shall not be used to populate this property. If management is not done via IP on this system, the special strings "Other" or "Serial only" shall be used.
string **DataIPAddrs** [ ]	Data IP address(es) for the system. There is no requirement that these be distinct from the ManagementIPAddrs. If the system does not serve data via IP, this property shall be NULL.
boolean **CompareFWVersionStrings ( )**	Compare two firmware version strings and provide an indication of which one succeeds the other and whether both are compatible with the device. A>B shall mean that version A is a later version than B.
string **A**	IN
string **B**	IN
string **Status**	OUT Values { "Unknown error", "A>B", "A==B", "A
boolean **SetLowSpaceWarningThresholds ( )**	Method to set the LowSpaceWarningThresholds on a system or pool. Two or more values should normally be given, in ascending order of severity (this should be the opposite of the percentage number order). For example, they might be given as { 20, 15, 10 }. In this case, the implementation should issue an informational alert when the AvailableSpaceRatio drops to 20, an urgent alert when it drops to 15, and a critical alert when it drops to 10 As SpaceInfo is present in many container objects, it is appropriate to set the warning thresholds array to NULL on elements to which one cannot physically add space, especially virtual elements.
string **Element**	Reference (SSM_Device) The element on which to set thresholds.

uint8 **Levels** [ ]	The levels to set, or NULL
boolean **GetCannedSettings ( )**	Conditional ("GetCannedSettings" in SupportedMethods) Return the list of canned settings supported by the device. The client shall choose and use one of the returned settings in operations on pools, volumes and filesystems. A value of True shall be returned upon successful completion, False otherwise. Clients may construct a settings template and pass it in in SettingTemplate.
string **SettingTemplate**	IN EmbeddedInstance (SSM_StorageSetting) A partially filled out setting to match against. A NULL or missing SettingTemplate matches everything. A NULL or missing property in the template also matches the corresponding property in every canned setting. The matching rules are a subset of the BRE matching rules in POSIX. See the section of the SSM spec entitled Pattern matching in GetCannedSettings.
string **Settings** [ ]	OUT EmbeddedInstance (SSM_Setting) The list of settings
boolean **GetBasicPools ( )**	Conditional ("GetBasicPools" in SupportedMethods) Return the list of available Basic pools. These are usually but not necessarily physical pools. The system may be reserving some Basic pools for parity, spares, replicas and so on, so this list may not comprise all the Basic pools in the system. It may be used to construct one or more higher-level pools. If furnished, the RequiredSetting indicates the QOS level that all returned pools shall be able to meet. The resulting list may therefore be a subset of the actual available Basic pools (c.f. the case of tiered storage). An invalid setting shall cause a return of False with an INVALID_ARGUMENT status.
string **SettingTemplate**	IN EmbeddedInstance (SSM_StorageSetting) A setting template that available pools' settings must meet or exceed.
string **PoolsToGet**	IN Optional Values { "GetAllPools", "GetAvailableOnly" } Whether to return all pools or only those that are uncommitted, i.e. available to be aggregated into higher-level pools. The default shall be GetAvailableOnly.
string **Pools** [ ]	OUT

	EmbeddedInstance (SSM_StoragePool) The list of Basic pools that can support the RequiredSetting. If NULL, there are no available Basic pools that meet the requirement.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **GetMaxPoolSize ( )**	Conditional ("GetMaxPoolSize" in SupportedMethods) Get the maximum size of pool that can be constructed using available pools that match a template.
string **SettingTemplate**	IN EmbeddedInstance (SSM_StorageSetting) The setting that all pools included in the space count shall be able to support. This shall be the InstanceID of one of the settings returned by GetCannedSettings. If NULL or not provided, the provider shall use the default setting. The client should not assume that this means that all Basic pools will be included.
uint64 **Size**	OUT The computed size, in bytes. This shall be a runtime figure based on available resources, not a system limit.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **CreateStoragePool ( )**	Conditional ("CreateStoragePool" in SupportedMethods) Values { "Creation Completed", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Argument", "StoragePool is too small to satisfy the request", "Default StoragePool cannot be created", "Illegal Setting", "Job Started" } Create a storage pool from other storage pools. This is normally done upon device discovery to create a larger pool out of the drives or pools discovered on the system. The Setting parameter shall be taken from the list of canned StorageSettings associated to the system via the StorageSettings association.
string **GivenName**	IN An end user relevant name for the StoragePool being created. If NULL, a system-supplied default name can be used. The value will be stored in the GivenName property for the created element.
string	IN

CannedSetting	EmbeddedInstance (SSM_StorageSetting) Settings for the StoragePool to be created. This is an element of class SSM_StorageSetting, encoded as a string-valued embedded instance parameter. The SSM model requires that this Setting be one of the settings returned by GetCannedSettings(). If NULL or the empty string, the service shall use the default canned setting.
string **InPools** [ ]	IN Reference (SSM_StoragePool) The StoragePools on which the created StorageVolume should reside. If this is NULL, the implementation should select from its set of available Basic pools that conform to the canned setting.
SSM_StoragePool **TheStoragePool**	OUT The newly created StoragePool.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **ModifyStoragePool ( )**	Conditional ("ModifyStoragePool" in SupportedMethods) Values { "Modification Completed", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Argument", "StoragePool In Use, Failed", "Incompatible Settings", "Cannot satisfy new Size", "Cannot set GivenName", "Job Started" } Modify a previously created StoragePool. Successful modification or creation of a Job to do so shall return True. The Status field in these cases shall contain either Completed or Job Started strings, respectively. All other results shall return False with the appropriate Status message, and shall indicate that no job has been started, and no action taken. The SSM Model does not support changing the Setting used to create the StoragePool. Only the Size and GivenName may be changed. Changing the size may be done by adding pools or by indicating a new size with or without specifying pools (in which case the system shall add or remove underlying pools at its discretion.)
string **GivenName**	IN A end user relevant name for the StoragePool being modified. If NULL, then the name will not be changed. If not NULL, this parameter supplies a new name for the StoragePool element. If the name cannot be set, a value of False with a status of Cannot set GivenName shall be returned.
uint64 **Size**	IN OUT The new minimum desired size for the StoragePool, in bytes. Systems may quantize this number upward based on the size of available Basic pools. The actual size used shall be

	returned in this property.
string **InPools** [ ]	IN Reference (SSM_StoragePool) A list of pools (not necessarily Basic) to add to the the pool to be modified. If this parameter is present and non-empty, the Size parameter shall be treated as follows: if 0, all the pools from the list should be added. If non-zero, the system should use pools from the list at its discretion to meet the desired size. If Size indicates an overall size reduction, this parameter shall be ignored. If the Setting on any of the InPools is incompatible with the setting on TheStoragePool, a value of False with an Incompatible Settings status shall be returned, and no action taken.
SSM_StoragePool **TheStoragePool**	IN The StoragePool to modify.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **DissolveStoragePool ( )**	Conditional ("DissolveStoragePool" in SupportedMethods) Dissolve a StoragePool into its constituent parts.
SSM_StoragePool **TheStoragePool**	IN The StoragePool to be deleted.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **EraseStoragePool ( )**	Conditional ("EraseStoragePool" in SupportedMethods) Erase a StoragePool. There are several kinds of media erasure defined in ISO/IEC 27040, but only two of them are pertinent in a software-only context. (Physical destruction is the most secure.) These are overwriting all the data using zeros, alternating patterns of 1s and 0s, or random data, up to and including DOD-style shredding. The other is cryptographic erasure. An encrypted pool may be erased by destroying all copies of the encryption key. Except for SEDs, which maintain a single undisclosed instance of the key, additional key management tasks may be required to cryptographically erase the pool. Clients should note well that erasure times vary enormously among storage types. SED technology is very fast on spinning media and solid state alike. Unencrypted solid state can be erased relatively quickly but may require a job. Unencrypted spinning media takes time up to days to completely scrub. The MediaEncryption property may be examined to see what kind of media is at hand. Clients shall make no assumptions about the quality of the erasure, but it is recommended that servers implement NIST-style multiple-pass erasure or use TCG SED technology.

boolean **Recursive**	IN Whether or not to erase recursively. If False and the pool is not a Basic pool, the method shall fail with a NON-BASIC_STORAGEPOOL error. It is recommended that clients try False first, particularly in SED environments. The default is False.
SSM_StoragePool **TheStoragePool**	IN The StoragePool to be erased.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **CreateStorageVolume ( )**	Conditional ("CreateStorageVolume" in SupportedMethods) Create a StorageVolume from one or more storage pools. The SSM Model does not support manual allocations from multiple pools; the storage system shall determine ratios statically or dynamically. If no input pool is specified, the system shall use or create a pool of its choosing. Clients may attempt to specify a provisioning scheme in the ProvisioningMethod parameter.
string **GivenName**	IN An end user relevant name for the StorageVolume being created. If NULL, a system-supplied default name may be used. The value shall be stored in the GivenName property for the created element.
uint64 **Size**	IN The nominal size of the volume. A value of zero shall mean that the volume may expand to consume all available resources. Otherwise it shall indicate the byte offset from the beginning of the volume at which writes may fail.
string **ProvisioningType**	IN Values { "Full", "Nominal", "System" } The type of provisioning desired. Full provisioning pre-allocates all storage resources needed to guarantee that an application will not run out of space in the container. Full provisioning combined with a size of 0 shall allocate all resources to the volume, including any added later. Nominal provisioning shall indicate that resources are allocated as writes occur, up to the volume Size. System provisioning, or NULL, shall indicate that the system should use its discretion. Other provisioning types are vendor specific.
SSM_StoragePool **InPools** [ ]	IN The StoragePool(s) on which the created StorageVolume should reside. If NULL or empty, the implementation shall create or use existing pools at its discretion.

SSM_StorageVolume **TheStorageVolume**	OUT The newly created StorageVolume.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **ModifyStorageVolume ( )**	Conditional ("ModifyStorageVolume" in SupportedMethods) Modify a previously created StorageVolume. The SSM Model supports changing the Size and GivenName of the volume. Changing the size may be done by adding pools or by indicating a new size with or without specifying pools (in which case the system shall add or remove underlying pools at its discretion). The system may or may not require the volume to be unmapped before performing this operation.
string **GivenName**	IN A end user relevant name for the StorageVolume being modified. If NULL, then the name will not be changed. If not NULL, this parameter supplies a new name for the StorageVolume element. If the name cannot be set, a value of Cannot set GivenName shall be returned.
uint64 **Size**	IN The new desired size for the StorageVolume, in bytes. Depending on the provisioning method, this may be the nominal or the fully allocated size.
SSM_StorageVolume **TheStorageVolume**	IN The StorageVolume to modify.
string **InPools** [ ]	IN Reference (SSM_StorageVolume) A list of pools to use. If provided, the system should use pools from the list at its discretion to meet the desired size. If Size indicates an overall size reduction, this parameter shall be ignored.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **DeleteStorageVolume ( )**	Conditional ("DeleteStorageVolume" in SupportedMethods) Delete a StorageVolume.

SSM_StorageVolume **TheStorageVolume**	IN the StorageVolume to be deleted.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **MapStorageVolume ( )**	Conditional ("MapStorageVolume" in SupportedMethods) Map one or more StorageVolume to one or more hosts (initiators) through one or more target ports on the system. The NetworkEndpoint class provides the network parameters necessary to perform this task. If a View is provided, the new paths shall be added to it; otherwise a new View shall be created. The resulting View shall be returned in the OutView parameter. If any of the Initiator or Port parameters are found to be incompatible with each other, the operation shall be terminated with no changes, and a value of False returned with an INCOMPATIBLE_ARGUMENTS error. Clients are advised to retry by adding paths one at a time to the View. The SSM Model does not support 'partial views'.
string **Initiators** [ ]	IN EmbeddedInstance (SSM_EndpointInfo) An array of initiators. If this list is empty or NULL, an INVALID_ARGUMENT error shall be returned.
string **Targets** [ ]	IN EmbeddedInstance (SSM_EndpointInfo) An array of targets. If this list is empty or NULL, an INVALID_ARGUMENT error shall be returned.
string **StorageVolumes** [ ]	IN Reference (SSM_StorageVolume) An array of StorageVolumes. If this list is empty or NULL, an INVALID_ARGUMENT error shall be returned.
string **InView**	IN EmbeddedInstance (SSM_StorageViewInfo) An existing view to add paths to. If this parameter is NULL and other parameters check out okay, a new view shall be created for the new paths and returned in the OutView parameter. An error referencing the InView shall cause an INVALID_VIEW error.
string **OutView**	OUT EmbeddedInstance (SSM_StorageViewInfo) The view resulting from the Attach operation upon successful completion.

SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **UnmapStorageVolume ( )**	Conditional ("UnmapStorageVolume" in SupportedMethods) Unmap a StorageVolume. The paths implied by the specified initiator and target ports shall be removed from the specified view. If the Disconnect parameter is present and true, the system should also disconnect any existing connections through these paths, but this capability is system dependent and should not be relied upon (the target ports may need to be shut down). The resulting View is returned in the OutView parameter. If any of the Initiator or Port parameters are not in the InView, they shall be ignored, and a value True with a COMPLETED_WITH_ERRORS status shall be returned.
string **Initiators** [ ]	IN Reference (SSM_EndpointInfo) An array of initiator ports. If this list is empty or NULL, no initiators shall be deleted from the View.
string **Targets** [ ]	IN Reference (SSM_EndpointInfo) An array of target ports. If this list is empty or NULL, no target ports shall be deleted from the View.
string **StorageVolumes** [ ]	IN Reference (SSM_StorageVolume) An array of StorageVolumes. If this list is empty or NULL, Invalid Argument shall be returned.
SSM_StorageViewInfo **InView**	IN An existing view to remove paths from. If this parameter is NULL it shall cause a return of Invalid View
string **OutView**	OUT EmbeddedInstance (SSM_StorageViewInfo) The view resulting from the Detach operation upon successful completion.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **GetVolumeViews ( )**	Conditional ("GetVolumeViews" in SupportedMethods) Get the list of Views pertaining to a given Volume.
string **Volumes** [ ]	IN Reference (SSM_StorageVolume) The InstanceIDs of the volumes to retrieve Views for. If the Which parameter is set to Only, then only return views that include all the specified volumes. If it is set to All, then return every view that includes a path for any of the specified volumes. The default is Only
string **Which**	IN Values { "All", "Only" } Which views to return, as related above (Default: Only)
string **OutViews** [ ]	OUT EmbeddedInstance (SSM_StorageViewInfo) The list of views whose paths contain the specified StorageVolume(s).
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **GetSupportedPerfCounters ( )**	A convenience routine to find out which performance counters are supported by the system and which are the more expensive ones.
string **ElementType**	IN Values { "StorageSystem", "Network Endpoint", "Volume", "FileSystem", "FileShare", "ObjectStore", "StoragePool", "PhysicalPool", "File", "Object", "Directory", "Container" } The type of element to get performance counter support for.
sint32 **ElementCount**	IN The number of elements to use as a basis for estimating the cost of the counter. For example, any fully loaded server will probably need a job to enumerate the performance data for all its volumes. By contrast, it may never need a job to report on a single volume. Clients that only need to get data on one or a few elements at a time may take advantage of this by providing a low number in this parameter. Provide -1 to indicate that the client intends to make calls enumerating all the elements of the given ElementType. Servers shall provide a best-effort answer given this parameter. Clients are cautioned that numbers like "1000" are problematic, and that servers may not be able to predict with complete accuracy where the cutoffs between Low,

	Medium and High occur for given element and counter types.
string **CounterCost**	IN Values { "Low", "Medium", "High", "All" } The type of counters to report. If this paramenter is Low, servers shall report only counters that never require jobs for the given element type and number of elements. Clients shall not be surprised if implementations don't report any jobless counters; however, vendor implementors who choose this shall not be surprised if clients proceed to act sub-optimally, as they have no choice but to treat all counters as equal. Therefore, vendors should attempt to report low-cost (jobless) counters as such. If this parameter is High, servers should report only high cost counters. Clients are cautioned that use of these counters will impact overall system performance and should be done sparingly. If this parameter is Medium, servers should report only counters that may require jobs or impact performance to a moderate extent. Clients are cautioned not to overuse these counters.
string **MatchingCounters** [ ]	OUT Counters supported by the implementation that match the criteria in ElementType, ElementCount and CounterCost.

boolean **GetPerfData ( )**	Conditional ("GetPerfData" in SupportedMethods) Get performance data for an attached or contained system element. Eligible element types are listed in the values enumeration for ElementType. The counters for which data is requested shall be listed in RequestedItems (this list is also called the "manifest"). If the server does not support one or more of the items in the manifest, an SSM_NOT_SUPPORTED error listing the unsupported filter items shall be returned in the return message header. The server should also return data for the requested items that are supported in the body of the return message. SSM does not support timed pushes of performance data. Clients may poll GetPerfData at timed intervals to achieve the same end.
string **Element**	IN Optional Reference (SSM_Device) A particular element to get performance data for.
string **ElementType**	IN Conditional (Element != NULL) Values { "StorageSystem", "Network Endpoint", "Volume", "FileSystem", "ObjectStore", "StoragePool", "PhysicalPool", "File", "Object", "Directory", "Container" } The type of element to get performance data for. When non-NULL and Element is NULL, supported data for all elements of this type shall be returned. When Element is non-NULL, this parameter shall be ignored.

string **RequestedItems** [ ]	IN Values { "TotalIOs", "KiBytesTransferred", "IOTimeCounter", "ReadIOs", "ReadHitIOs", "ReadIOTimeCounter", "ReadHitIOTimeCounter", "KiBytesRead", "WriteIOs", "WriteHitIOs", "WriteIOTimeCounter", "WriteHitIOTimeCounter", "KiBytesWritten", "IdleTimeCounter", "MaintOp", "MaintTimeCounter", "MaxWatts", "MinWatts", "InstWatts", "AvgWatts", "MaxPF", "MinPF", "InstPF", "AvgPF", "MaxVolts", "MinVolts", "InstVolts", "AvgVolts", "MaxAmps", "MinAmps", "InstAmps", "AvgVolts", "MaxTHD", "MinTHD", "InstTHD", "AvgTHD" } The individual performance parameters to return. Clients should query GetSupportedPerfCounters() to ascertain which manifest items may be expected to return without errors.
string **PerfData** [ ]	OUT EmbeddedInstance (SSM_PerfDataInfo) Returned performance data. Servers should return instances for all supported items requested and populate the return header with SSM_Message errors for each unsupported item that has been requested. Other errors shall be reported via appropriate SSM_Messages.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **CreateSnapshot ( )**	Conditional ("CreateSnapshot" in SupportedMethods) placeholder
boolean **DeleteSnapshot ( )**	Conditional ("DeleteSnapshot" in SupportedMethods) placeholder
boolean **CreateFileSystem ( )**	Conditional ("CreateFileSystem" in SupportedMethods) Create a FileSystem on a StoragePool. Successful creation of a filesystem shall return True. Successful creation of a job to create one shall return True and return the job in the OUT parameter Job. All other results shall return False and include an appropriate SSM_Message error in the return message header, and shall indicate that no job has been started, and no action taken. The OUT parameter TheFileSystem shall contain a Reference to the created FileSystem if this operation completed synchronously and successfully. In the event that a Job is created, the OUT parameter TheElement in the embedded object JobOutParameters

	property of the Job shall contain this Reference. The StoragePool to use is specified by the InPool parameter. If this is NULL, a default StoragePool should be created in a vendor-specific way and used. One way to create the default StoragePool is to use one of the canned settings supported by the service. The desired settings for the FileSystem are specified by the Setting parameter. Setting is a SSM_StorageSetting encoded as a string-valued embedded object parameter. Clients shall use settings that are returned by GetCannedSettings. Use of a non-pre-defined setting shall generate an error.
string **GivenName**	IN An end user relevant name for the FileSystem being created. If NULL, a system-supplied default name can be used. The value will be stored in the GivenName property for the created element.
SSM_StorageSetting **Setting**	IN Settings that the FileSystem will use and maintain. This is an element of class SSM_StorageSetting, encoded as a string-valued embedded instance parameter. The SSM model requires that this Setting be one of the settings returned by GetCannedSettings(). If NULL or the empty string, the service shall use the default canned setting.
SSM_StoragePool **InPool**	IN The StoragePool on which the created FileSystem should reside. If this is NULL, a default StoragePool should be created in a vendor-specific way and used.
uint64 **Size**	IN The requested size of the filesystem. The actual size that results may be quantized upward to meet various block boundary requirements.
string **ProvisioningType**	IN Values { "thin", "thick" } If the server does not support the type of provisioning requested (thin vs. thick), the operation shall fail with Thin type not supported status.
SSM_FileSystem **TheFileSystem**	OUT The newly created FileSystem. This shall either not be returned or be set to NULL if a Job is furnished.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

boolean **ModifyFileSystem ( )**	Conditional ("ModifyFileSystem" in SupportedMethods)

	Modify a previously created FileSystem. The SSM Model does not support changing the Setting used to create the FileSystem. Only the Size and GivenName may be changed. The Size parameter contains the new desired size for the FileSystem. A value of zero shall mean no change. If the new size can for some reason not be accommodated, a CANNOT_SATISFY_NEW_SIZE error shall be returned.
string **GivenName**	IN An end user relevant name for the FileSystem being modified. If NULL, then the name will not be changed. If not NULL, this parameter supplies a new name for the FileSystem element. If the name cannot be set, a value of Cannot set GivenName shall be returned.
uint64 **Size**	IN The new desired size for the FileSystem, in bytes.
SSM_FileSystem **TheFileSystem**	IN The FileSystem to modify.
string **InUseOptions**	IN Values { "Do Not Modify", "Wait, then Modify", "Quiesce, then Modify" } The action to take if the FileSystem is still in use when this request is made. This option is only relevant (and may be NULL) if the FileSystem must be made unavailable while the request is being executed.The default is Quiesce, then Modify.
uint32 **WaitTime**	IN An integer that indicates the time (in seconds) that the provider must wait before performing the request on this FileSystem. If InUseOptions = Quiesce, then Modify and WaitTime = 0 (the default), then the system shall wait (forever) until Quiescence, then modify the filesystem. A job shall be returned for all Wait and Quiesce requests with non-zero WaitTime.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL
boolean **DeleteFileSystem ( )**	Conditional ("DeleteFileSystem" in SupportedMethods) Delete a FileSystem.
SSM_FileSystem **TheFileSystem**	IN the FileSystem to be deleted.
string	IN

InUseOptions	Values { "Do Not Delete", "Wait, then Delete", "Quiesce, then Delete" } The action to take if the FileSystem is still in use when this request is made. This option is only relevant if the FileSystem must be made unavailable while the request is being executed. The options are (a) to fail the request, (b) wait for an amount of time specified in WaitTime and then delete the filesystem, and (c) attempt to quiesce the filesystem for the specified amount of time and then delete it.
uint32 **WaitTime**	IN An integer that indicates the time (in seconds) that the provider must wait before performing the request on this FileSystem. If WaitTime is not zero, the method should create a job, if supported by the provider, and return immediately. If the provider does not support asynchronous jobs, there is a possibility that the client could time out before the job is completed. If InUseOptions = Quiesce, then Delete and WaitTime = 0 (the default), then the system shall wait (forever) until Quiescence, then delete the filesystem, asynchronously and with a Job if possible.
SSM_Job **Job**	OUT The job, in the case that one has been created, otherwise NULL

5.48. SSM_StorageSystemJobs

An association for jobs running in the system.

class **SSM_StorageSystemJobs** : SSM_Association	Association Version (2.xx.0) Experimental
SSM_StorageSystem REF **Source**	Override (Source) The system that owns the job.
SSM_Job REF **Target**	Override (Target) The job.

5.49. SSM_StorageViewInfo

In SCSI, views are collections of masking and mapping paths. This class contains an array of these paths.

class **SSM_StorageViewInfo** : SSM_Object	Experimental Indication Version (2.xx.0)
string **Paths** [ ]	EmbeddedInstance (SSM_EndpointInfo) An array of endpoint pairs
string **Volume**	EmbeddedInstance (SSM_StorageVolume) The volume (LUN) in an I_T_L nexus.

5.50. SSM_StorageVolume

A StorageVolume is a byte-addressable container, usually called a 'LUN' in the storage industry. It is a simplified version of a CIM_StorageVolume, but includes necessary properties of what in the CIM schema comprises its hosting CIM_StorageExtent.

A HostedVolume association shall exist between every StorageVolume and the hosting StorageSystem. On clustered systems this will generally not be the top level scoping system, but a component system.

class **SSM_StorageVolume** : SSM_Device	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **SystemID**	The InstanceID of the scoping (enclosing) system in the SSM schema.
string **DurableName**	A persistent vendor-defined name for the object.
string **DurableNameFormat**	A description of what the DurableName signifies.
string **GivenName**	A name given by the user.
string **DurableNameFormat**	Override (DurableNameFormat) Values { "VPD83NAA6", "VPD83NAA5", "VPD83Type2", "VPD83Type1", "VPD83Type0", "SNVM", "NodeWWN", "NAA", "EUI64", "T10VID" }
string **CIMStorageVolume**	Reference (CIM_StorageVolume) The volume as represented in the CIM_ schema. If CIM objects are unavailable or not implemented, this property shall contain NULL
string **Access** [ ]	Values { "None", "Unknown", "Read", "Write", "Write Once", "Append", "Streaming" } Mounts of the volume will be restricted to these access modes. The mount itself may add further restrictions. Values may be ORed together, by placing both strings in the array, as follows: • { Read, Write } • { Read, Append }

	• { Read, Write Once } • { Streaming, Read } • { Streaming, Write } • { Streaming, Append } • { Streaming, Read, Write } • { Streaming, Read, Append } Append shall mean that writes may only be appended to an open file. Servers are not required to support every possible mode, and should return a NOT_SUPPORTED error if a client attempts to create a volume with an unsupported Access type.
uint64 **BlockSize**	Size in bytes of the blocks which form this StorageVolume. If the block size is variable, then the maximum block size in bytes should be specified. If the block size is unknown or if a block concept is not valid (for example, with Memory), enter a 1.
string **SpaceInfo**	EmbeddedInstance (SSM_SpaceInfo) Allocation and usage data.
string **Usage** [ ]	Values { "Other", "Unrestricted", "Reserved", "In Use", "Reserved by Replication Services", "Reserved by Migration Services", "Remote Replica Target", "Remote Replica Source", "Local Replica Target", "Local Replica Source", "Delta Replica Target", "Delta Replica Source", "Composite Volume Member", "Reserved for Sparing" } Indicates the intended usage or any restrictions that may have been imposed on the usage of this component. Multiple values may be supplied. Vendors should include Reserved for any volume that may not be used for other purposes. Vendors should include In Use for volumes that are in use but may be available for other purposes as well. Otherwise, vendors should include Unrestricted in the list. *Note: I'm sceptical that this is very useful. -- agy*
string **BasedOn** [ ]	Reference (SSM_Device) Underlying system objects that the StorageVolume is based on. Normally one or more pools.
uint8 **SnapshotUsed**	The percentage of total space in the volume that is used for snapshots. This should be the actual space used, not the amount of space set aside for a snapshot reserve. If snapshot data is kept elsewhere on the system, say in a separate container or on system storage, this number shall be zero (0).

5.51. SSM_SupportedPerfCounterInfo

A structure used to indicate support for a given performance counter, for various physical and virtual system elements.

class **SSM_SupportedPerfCounterInfo**	Experimental Indication Structure
string **ElementType**	Values { "StorageSystem", "Network Endpoint", "Volume", "FileSystem", "ObjectStore", "StoragePool", "PhysicalPool", "File", "Object", "Directory", "Container" } The type of element this item refers to.
string **PhysicalType**	Values { "Physical", "Virtual" } It is common for an implementation to support reporting of data for physical systems but not virtual ones. This field allows vendors to distinguish these cases. Implementations that support both for a given counter shall populate StorageSystem.SupportedPerfCounters with an instance of this structure for each one.
string **Counters** [ ]	Values { "TotalIOs", "KiBytesTransferred", "IOTimeCounter", "ReadIOs", "ReadHitIOs", "ReadIOTimeCounter", "ReadHitIOTimeCounter", "KiBytesRead", "WriteIOs", "WriteHitIOs", "WriteIOTimeCounter", "WriteHitIOTimeCounter", "KiBytesWritten", "IdleTimeCounter", "MaintOp", "MaintTimeCounter", "MaxPower", "MinPower", "InstPower", "AvgPower", "MaxVolts", "MinVolts", "InstVolts", "AvgVolts", "MaxAmps", "MinAmps", "InstAmps", "AvgVolts", "MaxTHD", "MinTHD", "InstTHD", "AvgTHD" } The performance counters that are supported for the listed element and physical types. See the PerfDataInfo class for descriptions pertaining to the named items. Any vendor-specific counters shall be, by SSM rules, prefixed with the same vendor ID used in InstanceIDs.

5.52. SSM_ViewItemInfo

In SCSI, views are collections of masking and mapping endpoint pairs. This structure resembles an I_T_L nexus in concept, and contains information about a single endpoint pair (termed a connection herein); a view is an array of these structures.

class **SSM_ViewItemInfo** : SSM_Object	Experimental Indication Version (2.xx.0) Structure
string **Protocol**	Values { "Other", "FC", "FCoE", "iSCSI", "SAS", "SATA", "Infiniband", "PCIe", "Not Connected" } The protocol in use for this connection
string **Initiator**	EmbeddedInstance (SSM_EndpointInfo) The initiator (host) endpoint
string **Target**	EmbeddedInstance (SSM_EndpointInfo) The target (server) endpoint
string **Volume**	EmbeddedInstance (SSM_StorageVolume) The volume that the path exposes.

5.53. SSM_WBEMServer

This class takes care of all WBEM infrastructure tasks, including indication support, CIM-XML communications and so on.

class **SSM_WBEMServer** : SSM_Object	Experimental Version (2.xx.0) RequiredIndications { "InstCreation", "InstModification", "InstDeletion", "Alert" }
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **Settings** [ ]	Reference (SSM_WBEMServerSetting) Settings for each of the protocols supported by the WBEM Server.
uint8 **DeliveryRetryAttempts**	The number of times message sends will be retried before giving up. Providers are not required to support changing of this number, but may do so. If the last attempt fails, the provider may remove the message, and if it is in response to an indication subscription, the provider may also remove the subscription.
uint32 **DeliveryRetryInterval**	The number of seconds between delivery attempts.
uint32 **SubscribeToIndication ( )**	A method a client may call to subscribe to or unsubscribe from one or more indications.
string **System** [ ]	IN Reference (SSM_StorageSystem) The system(s) to send the subscription request(s) to.
string **RequestType**	Values { "Subscribe", "Unsubscribe" } Whether this is a Subscribe or Unsubscribe request.
string **Filters** [ ]	IN Values { "InstCreation", "InstDeletion", "InstModification", "Alert" } The indication subscription(s). At least one subscription shall be specified. The values in the Values array shall be interpreted as follows • InstCreation - "SELECT * FROM InstCreation WHERE SourceInstance ISA

	SSM_Object • InstModification - SELECT * FROM InstModification WHERE SourceInstance ISA SSM_Object • InstDeletion - SELECT * FROM InstDeletion WHERE SourceInstance ISA SSM_Object • Alert - SELECT * FROM AlertIndication WHERE SourceInstance ISA SSM_Object There are no user-level subscriptions in SSM, and no query language. Implementations shall send indications for every event that satisfies the following criteria: 1. The class of the object the event pertains to contains the event type in its RequiredIndications qualifier. For example, a fan should not send InstModification indications, while a volume very well may. 2. There is no overriding belwether event.
string **Listeners** [ ]	IN The listener(s) to send the resulting indications to. At least one listener shall be specified. The format for IPv4 addresses shall be in dotted decimal format with a colon separator and port number, e.g. "10.1.10.20:4291". IPv6 addresses shall conform to the formats specified in RFC 5952.
uint32 **SendFakeMessage ( )**	Instruct one or more systems to send a fake error, alert or indication to a specified listener destination. This is done by sending an embedded instance in via this method. Any failure to return the same embedded instance back to the listener within a reasonable time shall indicate an error in the implementation.
string **Message**	IN EmbeddedInstance (SSM_Message)
string **Systems** [ ]	In EmbeddedInstance (SSM_StorageSystem)
string **ListenerDestinations** [ ]	IN
uint32 **GetWBEMServerNamespaces ( )**	Get the namespaces served by the WBEMServer. One of these shall be "/interop/"
string	OUT

namespaces [ ]	All the namespaces served by the WBEMServer. This array shall not be empty.
boolean **CreateAccount ( )**	Create a user. The rules for this are as follows 1. Security Officers may create Security Officers, Administrators and Operators. 2. Administrators may create Operators. All attempts to do anything else shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header.
string **AccountType**	IN Values { "Security Officer", "Administrator", "Operator" } The type ("role") of the account. The rules in the Description of this method shall be enforced.
string **AccountName**	IN The account name of the user to create. There are no requirements; SSM relies on the security practices of the vendor and customer to enforce proper account naming.
string **TemporaryPassword**	IN A temporary password for the account. The WBEMServer is required to force a password change upon first login to the new account. There are no other requirements; SSM relies on the security practices of the vendor and customer to enforce proper password formation.
string **EmailAddress**	IN An email address for the owner of the new account. The WBEMServer shall arrange for an email notification of the account creation to be sent to this address, containing information about how to access the account. The mechanism and content of the message are left to the discretion of the vendor.
boolean **SuspendAccount ( )**	Suspend (lock) or unsuspend (unlock) an account. The rules for this shall be: 1. Security Officers may suspend any account. 2. Administrators may suspend Operators. All attempts outside of these boundaries shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header.

string **AccountName**	IN The name of the account to suspend
string **OnOff**	IN Values { "Lock", "Unlock" } Whether to turn the account on or off.
boolean **ResetPassword ( )**	Reset the password on an account. The rules for this shall be: 1. Security Officers may reset the password on any account. 2. Administrators may reset the password of Operators. All attempts outside of these boundaries shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header. After reset, the same rules as are given in CreateUser() shall apply. That is, the account holder shall be required to change the account password upon next login. The mechanism for enforcing this is left to the discretion of the vendor.
string **AccountName**	IN The account name
string **TemporaryPassword**	IN A temporary password for the account. The WBEMServer is required to force a password change upon first login to the new account.
string **EmailAddress**	IN Optional An email address for the owner of the new account. If this property is non-NULL, the WBEMServer shall arrange for an email notification of the account creation to be sent to this address, containing information about how to access the account. The mechanism and content of the message are left to the discretion of the vendor.
boolean **DeleteAccount ( )**	Delete an account. The rules are as follows: 1. The Security Officer may delete any account except its own. 2. Administrators may delete Operator accounts. All attempts to do otherwise shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header.

string **AccountName**	IN The account to delete.
boolean **ListAccounts ( )**	List accounts. The rules for this are as follows: 1. Security Officers and Administrators may list all accounts. Attempts that fall outside this rule shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header.
string **Accounts** [ ]	OUT EmbeddedInstance (SSM_AccountInfo) Account records. Vendors should follow their adopted security practices in deciding whether to include the optional email address in these records.
boolean **ForcePasswordChange ( )**	Force a password change on the next login. The account holder shall be prompted for a new password, using the same vendor-defined mechanisms as elsewhere, the next time the account is successfully logged in to. The rules are as follows: 1. The Security Officer may force a password change on any account 2. Administrators may force change on Operator accounts. All attempts to do otherwise shall be rejected with an SSMM_ILLEGAL_OPERATION error message in the return header.
string **AccountName**	IN The account name

5.54. SSM_WBEMServerSetting

WBEM Server Settings.

class **SSM_WBEMServerSetting**	Experimental Version (2.xx.0)
string **InstanceID**	Key A 128-bit opaque UUID (GUID) generated in conformance with RFC 4122.
string **URI**	The URI at which clients should make requests based on the SupportedProtocol. This property shall be user configurable. The server shall provide a default value in the event that the user does not configure it.
string **SupportedProtocol**	Values { "CIM-XML", "WS-MAN", "CIM-RS" } The protocol in use at this entry point. In some cases there may be more than one, so it is not safe to assume that this is the only protocol in use at this entry point.
string **SupportedProtocolVersion**	The protocol version in use. This string is purposely not nailed down, to allow rapid movement to new protocol versions without requiring a spec update.
string **OwningOrg**	The owning organization for the protocol in use. Usually DMTF
string **BindTo**	An IP address to bind to. Leaving this NULL binds to all available NICs (not a good idea usually).

6. Standard Messages

Table 9 contains the standard error messages defined by SSM. In these entries, "msg" is the English language text for the error. Implementations targeted at other languages may use a localized version of this text. Placeholders in <arrow brackets> shall be filled in by the implementation at runtime.

Table 9 Standard error messages defined by SSM

Message ID		
COMPLETED_ WITH_ERRORS	msg	The operation <p1> completed with errors.
	<p1>	Method name in Class.MethodName() format.
JOB_STARTED	msg	A job was started to complete the call to <p1>. Its InstanceID is <p2>
	<p1>	Method name in Class.MethodName() format.
	<p2>	Job InstanceID
JOB_COMPLETED	msg	The job with InstanceID <p1> has completed
	<p1>	Job InstanceID
ACCESS_DENIED	msg	Access denied for property or method <p1>
	<p1>	Property or method name, in the format ClassName.PropertyOrMethodName()
NOT_SUPPORTED	msg	The operation <p1> is not supported on this device
	<p1>	Name of the operation or method. If a method, in Class.MethodName() format.
UNKNOWN_ERROR	msg	An unknown error has occurred
TIMEOUT	msg	The operation <p1> is not supported on this device
	<p1>	Name of the operation or method. If a method, in Class.MethodName() format.
INVALID_ARGUMENT	msg	Invalid argument <p1> to method <p2>
	<p1>	One of the argument names as given in the MOF
	<p2>	The name of the method that was invoked, in Class.MethodName() format.
INCOMPATIBLE_ARGUMENT	msg	The arguments <p1> and <p2> to method <p3> are incompatible
	<p1>	One of the argument names as given in the MOF
	<p2>	The other argument name, as given in the MOF
	<p3>	Name of the method that was invoked

Message ID		
INVALID_OFFSET	msg <p1> <p2>	The offset <p1> is invalid in operation <p2> The offending offset Name of the method or operation
DELETION_FAILED	msg <p1>	Deletion of object <p1> failed during operation <op> InstanceID of the object that couldn't be deleted Note: this message should be accompanied with an OBJECT_IN_USE message if that is the reason for the failure
FAILED	msg <p1>	The operation <p1> failed Name of the method or operation
INST_CREATION	msg <p1> <p2>	Object <p1> of type <p2> created InstanceID of the object Type of the object, e.g. "SSM_StorageVolume"
INST_MODIFICATION	msg <p1> <type>	Object <p1>of type <type> modified InstanceID of the object Type of the object, e.g. "SSM_StorageVolume"
INST_DELETION	msg <p1> <p2>	Object <p1> of type <p2> deleted InstanceID of the object Type of the object, e.g. "SSM_StorageVolume"
ELEMENT_OFFLINE	msg <p1> <p2>	Element <p1> of type <p2> has been taken offline InstanceID of the element Type of the element Possible values: "Fan", "CPU", "Board", "Enclosure", "Power Supply", "PhysicalPool", "Memory", "StorageVolume", "StoragePool".
ELEMENT_FAILED	msg <p1> <p2>	Element <p1> of type <p2> has died or been failed by the system InstanceID of the element Type of the element. Possible values: "Fan", "CPU", "Board", "Enclosure", "Power Supply", "PhysicalPool", "Memory", "StorageVolume", "StoragePool".

Message ID		
ELEMENT_ONLINE	msg < p1> <p2>	Element < p1> of type <p2> has been brought online InstanceID of the element Type of the element Possible values: "Fan", "CPU", "Board", "Enclosure", "Power Supply", "PhysicalPool", "Memory", "StorageVolume", "StoragePool".
FILESYSTEM_IN_USE	msg <p1> <p2>	The operation <p1> failed because the FileSystem <p2> is in use Name of the method that was invoked GivenName of the FileSystem
CANNOT_SATISFY_SIZE	msg <p1>	The operation <p1> failed because the size requested cannot be satisfied Name of the method that was invoked
THIN_TYPE_ NOT_SUPPORTED	msg <p1>	The operation <p1> failed because the thin provisioning type requested is not supported Name of the method that was invoked
CANNOT_SET_GIVENNAME	msg <p1> <p2>	The operation <p1> failed because the GivenName cannot be set on object <p2> Name of the method that was invoked InstanceID of the affected object
CANNOT_SET_ COPYPRIORITY	msg <p1>	The operation <p1> failed because the CopyPriority cannot be set on object <id> Name of the method that was invoked
NO_MATCH	msg <p1> <p2>	No <p1> matches the input pattern for operation <p2> Object type being matched against Name of the method that was invoked
STORAGEPOOL_ TOO_SMALL	msg <p1> <p2>	The operation <p1> could not be performed because the StoragePool <p2> is too small Name of the method that was invoked InstanceID of the StoragePool
DEFAULT_STORAGEPOOL_ NOT_CREATED	msg <p1>	The operation <p1> could not be performed because no default StoragePool could be found or created Name of the method that was invoked
ILLEGAL_OPERATION	msg <p1>	The operation <p1> either is or attempted to do something illegal or not permitted Name of the method that was invoked

Message ID		
ILLEGAL_SETTING	msg	The operation <p1> failed because the supplied setting is illegal
	<p1>	Name of the method that was invoked
INCOMPATIBLE_POOLS	msg	The operation <p1> failed because the pools <p2> and <p3> supplied to the call are incompatible
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of one of the incompatible pools
	<p3>	InstanceID of the other incompatible pool
MULTIPLE_POOLS_NOT_SUPPORTED	msg	The operation <p1> failed because multiple input pools are not supported
	<p1>	Name of the method that was invoked
POOL_NESTING_TOO_DEEP	msg	The operation <p1> failed because an attempt was made to nest pools too deeply (this is a system limit).
	<p1>	Name of the method that was invoked
POOL_IN_USE	msg	The operation <p1> failed because the pool <p2> is in use
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of the pool
STORAGEVOLUME_IN_USE	msg	The operation <p1> failed because the storage volume <p2> is in use
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of the StorageVolume
OBJECT_IN_USE	msg	The object <p1> is in use
	<p1>	InstanceID of the object
NON-BASIC_STORAGEPOOL	msg	The operation <p1> failed because a Basic StoragePool is required and <p2> is not a Basic pool
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of the non-basic pool
PROVISIONING_SCHEME_NOT_SUPPORTED	msg	The operation <p1> failed because the requested provisioning scheme is not supported
	<p1>	Name of the method that was invoked
COULD_NOT_SET_THRESHOLD	msg	The operation <p1> failed because the requested threshold(s) were out of bounds or otherwise malformed
	<p1>	Name of the method that was invoked

Message ID		
COULD_NOT_ CREATE_REPLICA	msg <p1> <p2>	The operation <p1> failed because a replica of element <p2> could not be created Name of the method that was invoked InstanceID of the element (volume, pool, etc.)
COULD_NOT_CREATE_ REPLICATION_ RELATIONSHIP	msg <p1> <p2> <p3>	The operation <p1> failed because a replication relationship between <p2> and <p3> could not be created Name of the method that was invoked InstanceID of the proposed source member InstanceID of the proposed target member
INSUFFIENT_SPACE_ IN_TARGETPOOL	msg <p1> <p2>	The operation <p1> failed because there is insufficient space in the target pool <p2> Name of the method that was invoked InstanceID of the target pool
LOST_COMMUNICATION_WITH _SOURCE	msg <p1> <p2> <p3>	The provider has lost communication with the source object <p1> (type <p2>) during a <p3> operation. InstanceID of the source object Type of the source object, e.g. SSM_StorageVolume The operation in progress: Copy or Move
LOST_COMMUNICATION_WITH _TARGET	msg <p1> <p2> <p3>	The provider has lost communication with the target object <p1> (type <p2>) during a <p3> operation. InstanceID of the target object Type of the target object, e.g. SSM_StorageVolume The operation in progress: Copy or Move
INVALID_SOURCE_TYPE	msg <p1> <p2>	Invalid source object type (<p1>) for operation <p2> Type of the source object, e.g. SSM_StorageVolume The operation in progress
INVALID_TARGET_TYPE	msg <p1> <p2>	Invalid target object type (<p1>) for operation <p2> Type of the target object, e.g. SSM_StorageVolume The operation in progress
UNSUPPORTED_ SOURCE_TYPE	msg <p1> <p2>	Unsupported source object type (<p1>) for operation <p2> Type of the source object, e.g. SSM_StorageVolume The operation in progress

Message ID		
UNSUPPORTED_ TARGET_TYPE	msg <p1> <p2>	Unsupported target object type (<p1>) for operation <p2> Type of the target object, e.g. SSM_StorageVolume The operation in progress
COPY_FAILED	msg <p1> <p2>	The Copy operation from source <p1> to target <p2> failed InstanceID of the source InstanceID of the target
MOVE_FAILED	msg <p1> <p2>	The Move operation from source <p1> to target <p2> failed InstanceID of the source InstanceID of the target
SNAPSHOT_FAILED	msg <p1> <p2>	The operation <p1> failed to take a snapshot of the source <p2> Name of the method or operation InstanceID of the source object
LOST_COMMUNICATION_WITH _REPLICA_SOURCE	msg <p1>	The provider has lost communication with the source device in a replication relationship <p1> InstanceID of the StorageReplica object that represents the replication relationship
LOST_COMMUNICATION_WITH _REPLICA_TARGET	msg <p1>	The provider has lost communication with the source device in the replication relationship <p1> InstanceID of the StorageReplica object that represents the replication relationship
ATTACH_FAILED	msg <p1> <p2>	The operation <p1> failed to attach the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
DETACH_FAILED	msg <p1> <p2>	The operation <p1> failed to detach the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
FRACTURE_FAILED	msg <p1> <p2>	The operation <p1> failed to fracture the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship

Message ID		
RESYNC_FAILED	msg <p1> <p2>	The operation <p1> failed to resynchronize the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
SUSPEND_FAILED	msg <p1> <p2>	The operation <p1> failed to suspend synchronization of the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
RESUME_FAILED	msg <p1> <p2>	The operation <p1> failed to resume synchronization of the source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
RESTORE_FAILED	msg <p1> <p2>	The operation <p1> failed to restore the source container from the target container in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
RESTART_FAILED	msg <p1> <p2>	The operation <p1> failed to restart the synchronization between source and target containers in the replication relationship <p2> Name of the method that was invoked InstanceID of the replication relationship
INCOMPATIBLE_VERSIONS	msg <p1> <p2>	The software versions "<p1>" and "<p2>" are incompatible The earlier version of the software, in string format The later version of the software, in string format
INVALID_VERSION_STRING	msg <p1>	The version string "<p1>" is invalid The version string
USE_GET_SUPPORTED_ SIZE_RANGE	msg	The operation GetSupportedSizes failed. Use GetSupportedSizeRange instead
USE_GET_SUPPORTED_ SIZES	msg	The operation GetSupportedSizeRange failed. Use GetSupportedSizes instead

Message ID		
INVALID_ELEMENT_TYPE	msg	The operation < p1> failed because the type of element <p2> is invalid
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of the offending element
INVALID_VIEW	msg	The operation <p1> failed because the view <p2> is invalid
	<p1>	Name of the method that was invoked
	<p2>	InstanceID of the StorageView
ELEMENT_OFFLINE	msg	The element <p1> has gone offline
	<p1>	InstanceID of the element
ELEMENT_ONLINE	msg	The element <p1> has gone online
	<p1>	InstanceID of the element
ELEMENT_FAILED	msg	The element <p1> has failed
	<p1>	InstanceID of the element
PASSWORD_CHANGED	msg	The password for account <p1> has been changed
	<p1>	InstanceID for the account
LOGIN	msg	Login from account <p1>
	<p1>	InstanceID for the account
LOGOUT	msg	Logout from account <acp1ct>
	<p1>	InstanceID for the account
ACCOUNT_CREATED	msg	Account <p1> created by Security Officer <p2>
	<p1>	Name of the account
	<p2>	Name of the Security Officer account
ACCOUNT_DELETED	msg	Account <p1> deleted by Security Officer <sec>
	<p1>	Name of the account
	<p2>	Name of the Security Officer account
MUST_RESET_PASSWORD	msg	The password on account <p1> must be reset by the account holder before management requests from this account can be honored.
	<p1>	Name of the account

Message ID		
AVAILABLE_SPACE_ALERT	msg	Available space on the <p1> <p2> (<p3>) has dropped to <p4>%.
	<p1>	The type of entity { "StorageSystem", "StoragePool", "StorageVolume", "FileSystem", "ObjectStore" }
	<p2>	GivenName of the entity
	<p3>	InstanceID of the entity
	<p4>	SpaceInfo.AvailableSpaceRatio of the entity
VENDOR_MESSAGE	msg	<p1> message <p2>: <p3>
	<p1>	Stock ticker symbol or unique name for the vendor; a system type (e.g. VNX) may be included
	<p2>	Message ID assigned by the vendor
	<p3>	Arbitrary message text
MESSAGE_DETAIL	msg	<p1>
	<p1>	Additional detail the vendor may wish to include in an error message header

7. References

TBD