

IBM FlashSystem A9000  
IBM FlashSystem A9000R  
Version 12.0.3

*Open API Reference Guide*



**Note**

Before using this document and the product it supports, read the information in [“Notices” on page 75](#).

Publication number: SC27-8561-04. This publication applies to IBM FlashSystem A9000 and IBM FlashSystem A9000R version 12.0.3 and to all subsequent releases and modifications until otherwise indicated in a newer publication.

© **Copyright International Business Machines Corporation 2016, 2018.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>List of Figures.....</b>	<b>v</b>
<b>List of Tables.....</b>	<b>vii</b>
<b>About this guide.....</b>	<b>ix</b>
Who should use this guide.....	ix
Conventions used in this guide.....	ix
Related information and publications.....	ix
Getting information, help, and service.....	ix
IBM Publications Center.....	x
Sending or posting your comments.....	x
<b>Chapter 1. Introduction.....</b>	<b>1</b>
Storage Management Initiative Specifications (SMI-S) overview.....	1
Common Information Model (CIM) agent overview.....	1
Common Information Model (CIM) concepts.....	2
Common Information Model (CIM) agent components.....	3
Open API element definitions.....	4
Common Information Model (CIM) agent security.....	5
<b>Chapter 2. Installation and configuration.....</b>	<b>7</b>
Multitenancy feature support information.....	7
<b>Chapter 3. Communication concepts and methods.....</b>	<b>9</b>
Common Information Model (CIM) agent communication concepts.....	9
Common Information Model (CIM) agent communication methods.....	9
Obtaining a single class from the target namespace.....	10
Obtaining a single instance from the target namespace.....	11
Deleting a single instance from the target namespace.....	11
Creating an instance in the target namespace.....	12
Modifying an existing instance in the target namespace.....	12
Enumerating classes within a defined target namespace.....	13
Enumerating the names of subclasses of a class defined within the target namespace.....	14
Enumerating instances of a defined class within the target namespace.....	14
Enumerating the names of instances of a class within a target namespace.....	15
Processing a query against the target namespace.....	16
Enumerating classes or instances that are associated with a specific Common Information Model (CIM) object.....	16
Enumerating the names of classes or instances associated with a specific Common Information Model (CIM) object.....	17
Enumerating the association objects that refer to a specific target class or instance.....	18
Enumerating the names of the association objects that refer to a specific target class or instance.....	19
Retrieving a single property value from an instance in the target namespace.....	20
Setting a single property value within an instance in the target namespace.....	20
Retrieving a single qualifier declaration from the target namespace.....	21
Creating or modifying a qualifier declaration in the target namespace.....	21
Enumerating qualifier declarations from the target namespace.....	22
Common Information Model (CIM) agent communication methods that cannot be used.....	22
Return error codes.....	22

<b>Chapter 4. Functional profiles, diagrams, and methods.....</b>	<b>27</b>
Block Server performance profile.....	27
Block Server Performance object model.....	28
Block Server Performance methods.....	30
Block Services profile.....	31
Block Services object model.....	32
Block Services methods.....	35
iSCSI Target Ports profile.....	39
iSCSI Target Ports object model.....	40
iSCSI Target Ports methods.....	42
Masking and Mapping profile.....	44
Masking and Mapping object model.....	46
Masking and Mapping methods.....	48
Indication profile.....	52
Replication Services profile.....	54
Replication Services object model.....	55
Replication Services methods.....	57
Replication Services indications.....	67
Job Control profile.....	67
Job Control object model.....	68
Thin provisioning profile.....	68
Thin Provisioning object model.....	69
Thin Provisioning methods.....	69
Thin Provisioning indications.....	71
<b>Chapter 5. Conformance tests.....</b>	<b>73</b>
<b>Notices.....</b>	<b>75</b>
Trademarks.....	76
<b>Glossary.....</b>	<b>77</b>
<b>Index.....</b>	<b>85</b>



---

## List of Figures

1. How a CIM agent works.....	2
2. The MOF compiler stores the model in the CIMOM data store.....	3
3. Block Server Performance SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	29
4. Block Services SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	32
5. Block Services Package with Settings and Capabilities model .....	33
6. iSCSI Target Ports SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	40
7. Masking and mapping physical model in IBM FlashSystem A9000 and A9000R systems.....	45
8. Masking and mapping object model in SMI-S.....	46
9. Replication Services (Local) SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	55
10. Replication Services (Remote) SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	56
11. Sample local group information retrieval.....	58
12. Sample remote group information retrieval.....	61
13. Association classes for mirrored volumes and consistency groups.....	61
14. Job Control SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	68
15. Thin Provisioning SMI-S model for IBM FlashSystem A9000 and A9000R systems.....	69



---

## List of Tables

1. Functional groups for the CIM agent.....	10
2. GetClass method parameters.....	10
3. GetInstance method parameters.....	11
4. DeleteInstance method parameters.....	12
5. CreateInstance method parameters.....	12
6. ModifyInstance method parameters.....	13
7. EnumerateClasses method parameters.....	13
8. EnumerateClassNames method parameters.....	14
9. EnumerateInstances method parameters.....	14
10. EnumerateInstanceNames method parameters.....	15
11. ExecuteQuery method parameters.....	16
12. Associators method parameters.....	16
13. Associators method parameters.....	17
14. References method parameters.....	18
15. ReferenceNames method parameters.....	19
16. GetProperty method parameters.....	20
17. SetProperty method parameters.....	20
18. GetQualifier method parameters.....	21
19. SetQualifier method parameters.....	21
20. Return error codes for the CIMOM.....	23
21. Block Server Performance metrics.....	27
22. Synchronous actions.....	34
23. iSCSI terminology and CIM class names.....	41
24. Masking and mapping classes.....	46
25. ExposePaths use cases, parameters, and parameter values.....	50
26. HidePaths use cases, parameters, and parameter values.....	51
27. Indication types and object classes.....	53
28. Mapping IBM FlashSystem A9000 and A9000R terminology to SMI terminology.....	56
29. Replication Service methods.....	57
30. Group management classes.....	58
31. Extrinsic methods for group management.....	58
32. Replication management classes.....	60
33. Extrinsic methods for replication management.....	62
34. Operations, operation descriptions, and corresponding WaitForCopyState states.....	64
35. Replication Services profile indications.....	67
36. Replication Service methods.....	69
37. Thin Provisioning profile indications.....	72
38. SNIA CTP results.....	73



## About this guide

---

This publication introduces the IBM FlashSystem A9000 and IBM FlashSystem A9000R Open Application Programming Interface (API), which is referred to as the Common Information Model (CIM) agent. This publication can assist you in writing your CIM-based applications for the IBM FlashSystem A9000 and A9000R Open API.

This publication supports the IBM FlashSystem A9000 and A9000R Open API microcode version 12.0.1.

## Who should use this guide

---

This publication is for system administrators and system and application programmers, or whomever is responsible for implementing the IBM FlashSystem A9000 and A9000R Open API and configuring the Common Information Model (CIM) agent.

This publication assumes that you understand the general concepts of the operating system and Internet capabilities for your enterprise.

## Conventions used in this guide

---

These notices are used to highlight key information.

---

**Note:** These notices provide important tips, guidance, or advice.

---

---

**Important:** These notices provide information or advice that might help you avoid inconvenient or difficult situations.

---



---

**Attention:** These notices indicate possible damage to programs, devices, or data. An attention notice appears before the instruction or situation in which damage can occur.

---

## Publications and related information

---

You can find additional information and publications related to IBM FlashSystem® A9000 and A9000R on the following information sources.

- [IBM FlashSystem A9000 on the IBM Knowledge Center](http://ibm.com/support/knowledgecenter/STJKMM) (ibm.com/support/knowledgecenter/STJKMM)
- [IBM FlashSystem A9000R on the IBM Knowledge Center](http://ibm.com/support/knowledgecenter/STJKN5) (ibm.com/support/knowledgecenter/STJKN5)
- [Storage Networking Industry Association \(SNIA\) website](http://www.snia.org) (www.snia.org)
- [Distributed Management Task Force \(DMTF\) website](http://www.dmtf.org) (www.dmtf.org)

## Getting information, help, and service

---

If you need help, service, technical assistance, or want more information about IBM products, you can find various sources to assist you. You can view the following websites to get information about IBM products and services and to find the latest technical information and support.

- [IBM website](http://ibm.com) (ibm.com®)
- [IBM Support Portal website](http://www.ibm.com/storage/support) (www.ibm.com/storage/support)
- [IBM Directory of Worldwide Contacts website](http://www.ibm.com/planetwide) (www.ibm.com/planetwide)

## IBM Publications Center

---

The IBM Publications Center is a worldwide central repository for IBM product publications and marketing material.

The [IBM Publications Center website](http://ibm.com/shop/publications/order) ([ibm.com/shop/publications/order](http://ibm.com/shop/publications/order)) offers customized search functions to help you find the publications that you need. You can view or download publications at no charge.

## Sending or posting your comments

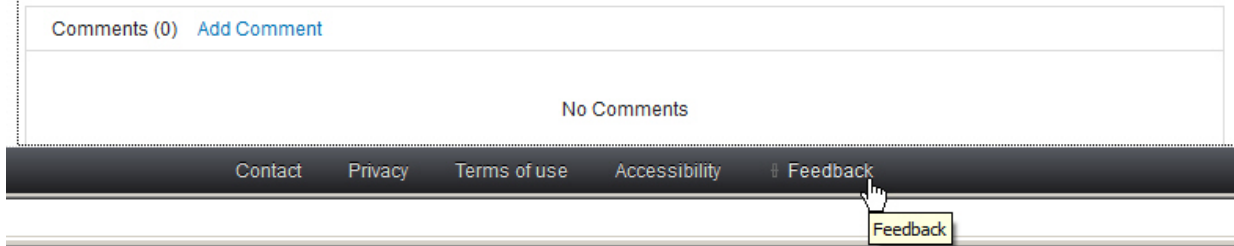
---

Your feedback is important in helping to provide the most accurate and highest quality information.

To submit any comments about this guide:

- Go to IBM Knowledge Center ([ibm.com/support/knowledgecenter](http://ibm.com/support/knowledgecenter)), drill down to the relevant page, and then click the **Feedback** link that is located at the bottom of the page.

By adding a comment, you accept our [IBM Knowledge Center Terms of Use](#). Your comments entered on this IBM Knowledge Center site do not represent the views or opinions of IBM. IBM, in its sole discretion, reserves the right to remove any comments from this site. IBM is not responsible for, and does not validate or confirm, the correctness or accuracy of any comments you post. IBM does not endorse any of your comments. All IBM comments are provided "AS IS" and are not warranted by IBM in any way.



The feedback form is displayed and you can use it to enter and submit your comments privately.

- You can post a public comment on the Knowledge Center page that you are viewing, by clicking **Add Comment**. For this option, you must first log in to IBM Knowledge Center with your IBM ID.
- You can send your comments by email to [starpubs@us.ibm.com](mailto:starpubs@us.ibm.com). Be sure to include the following information:
  - Exact publication title and product version
  - Publication form number (for example: SC01-0001-01)
  - Page, table, or illustration numbers that you are commenting on
  - A detailed description of any information that should be changed

---

**Note:** When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

---

---

# Chapter 1. Introduction

The IBM FlashSystem A9000 and A9000R Open Application Programming Interface (API) is a non-proprietary storage-management client application.

The Open API uses the Storage Management Initiative Specification (SMI-S), as defined by the Storage Networking Industry Association (SNIA) to view LUN information.

You can use the Open API to integrate configuration-management support into storage resource management (SRM) applications so that you can use your existing SRM application and infrastructures to configure and manage IBM FlashSystem A9000 and A9000R systems. The Open API presents another option for managing your systems by complementing the use of IBM Hyper-Scale Manager and the command-line interface (CLI). The Open API is an embedded component of IBM FlashSystem A9000 and A9000R systems.

You can implement the Open API without using a separate middleware application, like the IBM System Storage Common Information Model (CIM) agent, which provides a CIM-compliant interface. The Open API uses the CIM technology to manage proprietary devices as open system devices through storage management applications. The Open API is used by storage management applications to communicate with IBM FlashSystem A9000 and A9000R systems.

---

## Storage Management Initiative Specifications (SMI-S) overview

The Storage Management Initiative Specifications (SMI-S) are the leading storage-management API standards that are widely adopted by storage vendors. Using the SMI-S API, customers or independent software vendors (ISVs) can develop their own software solutions for managing heterogeneous storage deployments. For example, Microsoft is using SMI-S in Microsoft System Center Virtual Machine Manager 2012 to provide rapidly-integrated management of advanced disk arrays without needing any storage vendor add-on.

All IBM storage products support the SMI-S standard, which defines the Common Information Model (CIM) of each storage system management and is used as the base model for ISV and customer solution development. The built-in support for SMI-S in IBM storage systems, including IBM FlashSystem A9000 and A9000R, facilitates the development of different management solutions such as monitoring and control of physical storage, configuration and provisioning of SAN storage, and copying or protection of data through remote mirroring and snapshots.

The SMI-S standard is driven by the Storage Networking Industry Association (SNIA), and the CIM standard is driven by the Distributed Management Task Force (DMTF).

---

## Common Information Model (CIM) agent overview

A Common Information Model (CIM) agent provides a means by which a device can be managed by common building blocks rather than proprietary software. If a device is CIM-compliant, software that is also CIM-compliant can manage the device. Vendor applications can manage CIM-compliant devices in a common way, rather than using device-specific programming interfaces. You can perform tasks in a consistent manner across devices and vendor applications.

A CIM agent consists of the components that are shown in Figure 1 on page 2. The main components are the CIM object manager (CIMOM), the service location protocol (SLP), and the device provider. A device can be a storage system such as IBM FlashSystem A9000 or IBM FlashSystem A9000R. The CIM agent registers itself with the SLP Service Agent (SLP SA) to enable discovery by the client application.

The SLP DA is a directory service daemon that a client application calls to locate the CIMOM. The client application and the CIMOM communicate through CIM messages. The CIMOM and device provider communicate through method calls made from the CIMOM to the provider. The device provider communicates with the device through proprietary calls.

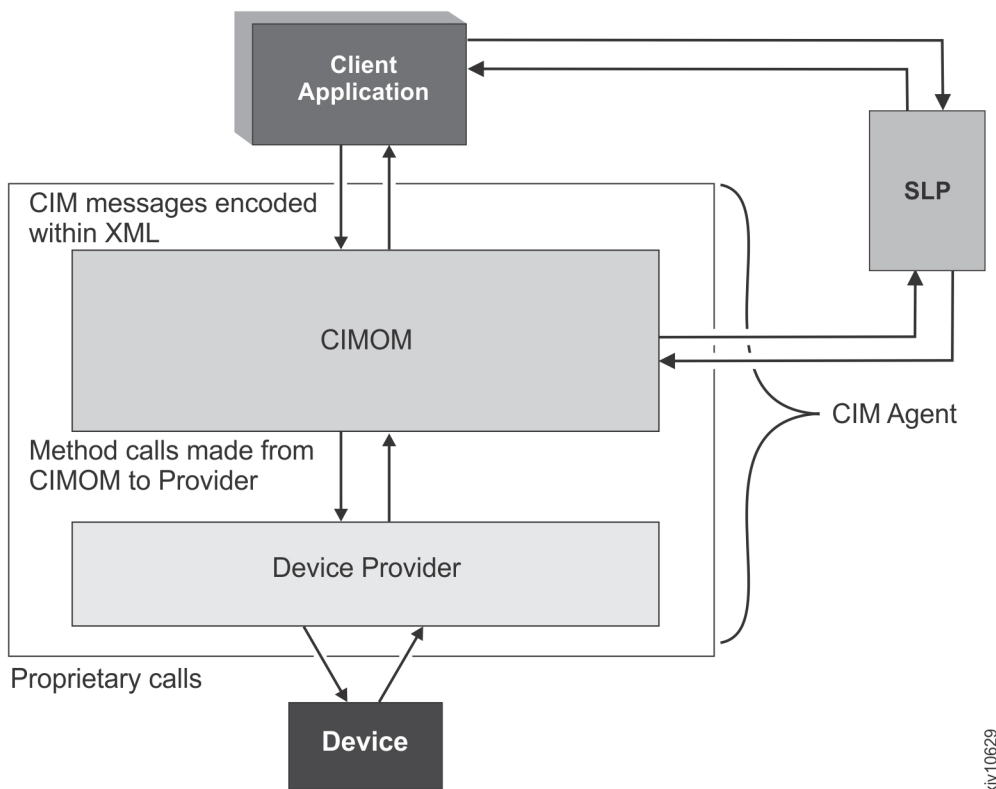


Figure 1: How a CIM agent works

The CIMOM supports the following specifications and standards:

- Distributed Management Task Force (DMTF) Specification for CIM Operations over HTTP, Version 1.4
- CIM Specification, version 2.40.0
- Storage Networking Industry Association (SNIA) Storage Management Initiative Specification (SMI-S) and the Shared Storage Model (SSM), a framework for describing storage architectures, version 1.6

These specifications allow a CIM agent to act as an open-system standards interpreter. Other CIM-compliant storage resource management applications (IBM and non-IBM) can interoperate with each other.

When you install, configure, and enable the CIM agent on a host server or an administrative workstation within your network, that host server or workstation can communicate with your IBM FlashSystem A9000 or IBM FlashSystem A9000R through the CIM agent. CIM-compliant applications like the CIM agent can manage the data on your system.

The following sites provide more information about the CIM standards:

- DMTF Common Information Model (CIM) Standards  
<http://www.dmtf.org/standards/cim/>
- Storage Networking Industry Association Standards  
[http://www.snia.org/tech\\_activities/standards/curr\\_standards/smi/](http://www.snia.org/tech_activities/standards/curr_standards/smi/)

## Common Information Model (CIM) concepts

The Common Information Model (CIM) is an open approach to the management of systems and networks.

The CIM provides a common conceptual framework applicable to all areas of management, which includes systems, applications, databases, networks, and devices. The CIM specification provides the language and the methodology that is used to describe management data.



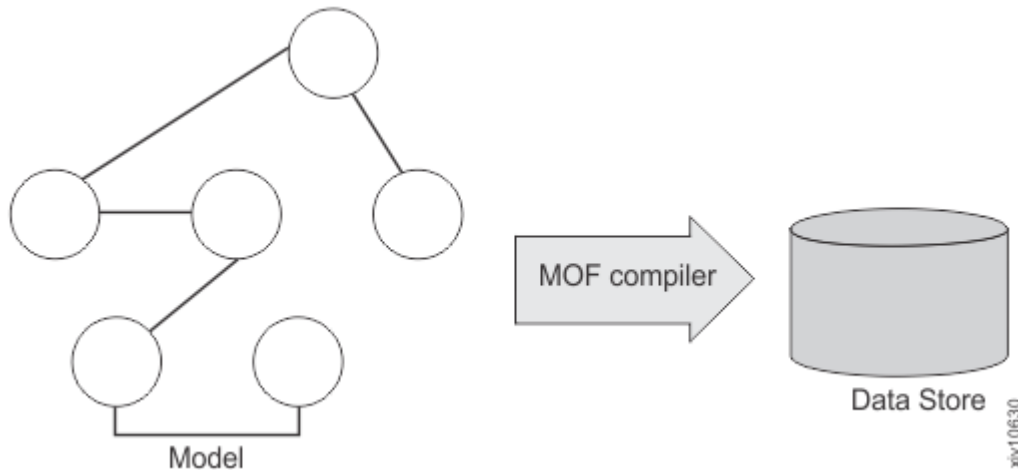
The CIM defines a set of classes with properties and associations that provide a conceptual framework. The framework can be used to organize data for a specific managed environment, such as data storage. CIM Schema 2.11 for Managing a Storage Array provides information about enabling management applications to manage data in a common way.

The CIM standards and the Distributed Management Task Force (DMTF) specification provide information about Web-based enterprise management (WBEM) operations over HTTP.

When the CIM object manager (CIMOM) first starts, it registers itself to the SLP. It provides information about its location (IP address and port) and the type of service it provides. A client application finds the location of the CIMOM by calling an SLP directory service. After this information is obtained, the client application opens direct communication with the CIMOM.

A client sends requests to a CIMOM in the context of a CIM model. The model is defined by the CIM schema and loaded into the repository of the CIMOM. [Figure 2 on page 3](#) shows how the schema is loaded into the data store of the CIMOM. The managed object format (MOF) compilation and creation of the data store is managed automatically during installation.

As requests arrive, the CIMOM validates and authenticates each request. Requests are either directed to the appropriate functional component of the CIMOM or directed to a device-specific handler called a provider.



*Figure 2: The MOF compiler stores the model in the CIMOM data store.*

A provider makes device-unique programming interface calls on behalf of the CIMOM to satisfy a client application request. Such requests generally map a CIM request to the API for a device. A request to get an instance of a class or a property of an instance, for example, might be directed to a provider. Then, the provider might make one or many requests of a device by using the unique API for the device. [Figure 1 on page 2](#) shows the communication structure between the device and the client application.

## Common Information Model (CIM) agent components

With a Common Information Model (CIM) agent, application programmers can use common building blocks rather than proprietary software or device-specific programming interfaces to manage CIM-compliant devices. Standardization of the way that applications manage storage provides easier storage management.

The following list describes a CIM agent and its components:

### **CIM agent**

An agent that interprets open-system data as it is transferred between the API and a device or a storage unit.

**CIM object manager (CIMOM)**

A common conceptual framework for data management. Receives, validates, and authenticates client application requests, and then directs requests to the appropriate functional component or to a device provider.

**client application**

A storage management API that initiates a request to a device or a data storage unit such as an IBM FlashSystem A9000 or IBM FlashSystem A9000R.

---

**Note:** A client application is not provided with the CIM agent, and it must be supplied by the customer

---

**Service Location Protocol (SLP)**

The SLP DA is a directory service that a client application calls to locate the CIMOM. The SLP SA is a service agent that enables discovery by a client application.

**storage unit (also known as a storage server)**

The final destination of a client application request and the processor of the request.

**storage unit provider**

A storage unit-specific handler that receives client application requests that are destined for its device or storage unit.

## Open API element definitions

---

The IBM FlashSystem A9000 and A9000R Open API elements include schemas, classes, properties, methods, indications, associations, references, and qualifiers, as described in the following list.

**Schema**

A group of classes that are defined to a single namespace. Within the Common Information Model (CIM) agent, the schemas that are supported are the ones that are loaded through the managed object format (MOF) compiler.

**Class**

The definition of an object within some hierarchy. Classes can have methods and properties and be the target of an association.

**Property**

A value that is used to characterize instances of a class.

**Method**

An implementation of a function on a class.

**Indication**

An object representation of an event.

**Association**

A class that contains two references which define a relationship between two objects.

**Reference**

A unique identifier of an object that is based on its key properties.

**Qualifier**

Additional information about other elements, classes, associations, indications, methods, method parameters, instances, properties, or references.

**Namespace**

A namespace defines the scope over which an IBM FlashSystem A9000 and A9000R Open API schema applies. IBM FlashSystem A9000 and A9000R Open API operations are always run within the context of a namespace. An IBM FlashSystem A9000 and A9000R Open API schema or version is loaded into a namespace when that schema is compiled by the MOF compiler. The namespace must be specified within the message that the client sends to the Open API.

Clients cannot create new namespaces. Attempts to do so result in errors.

### **Object name**

An object name consists of a namespace path and a model path. The namespace path provides access to the API implementation that is managed by the IBM FlashSystem A9000 and A9000R Open API. The model path provides navigation within the implementation. The following example shows an object name:

```
http://cimom.host.com/root/ibm:CIM_Class.key1=value1,key2=value2
```

where `http://cimom.host.com/root/ibm` is the namespace path  
and `:CIM_Class.key1=value1,key2=value2` is the model path.

## **Common Information Model (CIM) agent security**

---

The Common Information Model (CIM) agent can operate in both secure and unsecure modes.

### **Secure mode**

All requests between the client application and the CIM object manager (CIMOM) are XML encoded requests. They are sent over HTTP or HTTP over Secure Sockets Layer (SSL). The CIMOM, upon receiving a request, parses the request and processes it. Responses, when they are returned to the client application, are transformed into XML-encoded CIM status and returned in HTTP responses to the client. The CIM agent runs in secure mode by using SSL by default.

### **Unsecure mode**

Some vendor software cannot communicate with the CIM agent in a secure mode. You can still use this vendor software by configuring the CIM agent to run with only basic user name and password security. See the configuration instructions for your operating system for the instructions for configuring the CIM agent for this less secure mode.



---

## Chapter 2. Installation and configuration

The Common Information Model (CIM) agent is preinstalled and embedded on the administrative module, and it is enabled by default and preconfigured. You can manage IBM FlashSystem A9000 and A9000R systems from the CIM agent that is bundled with the administrative module.

Additional details regarding the preinstalled and preconfigured CIM agent:

- The embedded CIM agent does not require configuration changes to manage IBM FlashSystem A9000 and A9000R systems.
- The CIM object manager (CIMOM) is installed and running on all three administrative modules. The clients can connect to any of the administrative modules, and the same results are provided to the CIMOM.
- You can use the `cim_enable` and `cim_disable` commands through the command-line interface (CLI) to enable or disable the CIM agent. While the CIM agent is enabled, a watchdog process monitors it to ensure it is always running.

The CIM agent embedded on the administrative module has the following limitations:

- The CIM agent can support only IBM FlashSystem A9000 and A9000R systems on which the administrative module is located. The CIM agent is not able to manage any other IBM FlashSystem A9000 or IBM FlashSystem A9000R system.
- The CIM agent must use secure connections over port 5989.
- The CIM agent uses the IBM FlashSystem A9000 or IBM FlashSystem A9000R system user account to authenticate. To manage accounts, you must use IBM Hyper-Scale Manager or the CLI.

---

**Note:** CIM uses XML format when communicating with the IBM FlashSystem A9000 and A9000R systems, so the user name and password can't include the following special characters: `&`, `<`, `>`, `'`, `"`

If you must use these characters in your user name and password, manually replace them with their XML equivalents, as shown below:

`& => &amp;`;

`< => &lt;`;

`> => &gt;`;

`' => &apos;`;

`" => &quot;`;

---

### Multitenancy feature support information

---

For IBM FlashSystem A9000 and A9000R, the multitenancy feature is supported.

To support this feature, IBM FlashSystem A9000 and A9000R implement the concept of a domain that will link users to their dedicated pools, hosts, mirror targets, and other resources. A user can manage only the storage resources to which they are associated, without the ability to modify or monitor other system resources. The domain restricts the set of objects a user can manage to those associated with the domain. If a user is associated with one or more domains, they are a domain user, otherwise, they are a global user. This means that users in the same category but different domains will manage different resources.

The Common Information Model (CIM) agent for IBM FlashSystem A9000 and A9000R, however, does *not* support the multitenancy feature. CIM only supports the global user with the domain policy access value of `OPEN`. CIM requires a global Storage administrator/Application administrator/Read-only user who can manage all the resources in the system.

See [IBM FlashSystem A9000 and A9000R: Architecture, Implementation, and Usage](http://www.redbooks.ibm.com/abstracts/sg247659.html) (www.redbooks.ibm.com/abstracts/sg247659.html) and the *IBM FlashSystem A9000 Product Overview* or *IBM FlashSystem A9000R Product Overview* for more information about the architecture of IBM FlashSystem A9000 and A9000R systems.

---

## Chapter 3. Communication concepts and methods

Refer to the following topics for information about Common Information Model (CIM) agent communication concepts and methods.

- [“Common Information Model \(CIM\) agent communication concepts” on page 9](#)
- [“Common Information Model \(CIM\) agent communication methods” on page 9](#)

---

### Common Information Model (CIM) agent communication concepts

The Common Information Model (CIM) agent uses client communication and intrinsic and extrinsic methods of communication.

#### Client communication

A client application communicates with the CIM agent through operation request messages that are encoded within XML. The CIM agent returns responses with operation response messages. Requests and responses are sub-elements of the <CIM MESSAGE> element.

A <MESSAGE> sent to the CIM agent must contain an ID attribute. A response from the CIM agent returns this value and enables the client to track requests and their responses.

The CIM agent supports simple requests and simple responses. *Simple requests* are operation request messages that contain the <SIMPLEREQ> XML tag. *Simple responses* are operation response messages that contain the <SIMPLERSP> XML tag. A client application determines that the CIM agent supports only simple operation requests and responses by examining the output of the OPTIONS method.

#### Intrinsic and extrinsic methods

All operations on the CIM agent are completed by running one or more methods. A method is either an intrinsic method or an extrinsic method.

*Intrinsic methods* are supported by the CIM agent itself. These methods are included within <IMETHODCALL> XML tags that are sent in messages to the CIM agent.

*Extrinsic methods* are defined by the schema that is supported by the CIM agent. These methods are included within <METHODCALL> XML tags that are sent in messages to the CIM agent. Client applications can call on the CIM agent by using these methods. These methods fall within certain functional groups that might be supported by the CIM agent.

---

### Common Information Model (CIM) agent communication methods

Client application calls to intrinsic methods can result in Common Information Model (CIM) agent calls to the device provider. This result happens when the device provider surfaces the classes or instances that are referenced in the calls.

The CIM agent returns IMETHODRESPONSE or METHODRESPONSE elements to the client application when the intrinsic or extrinsic methods are used. These elements are contained within a MESSAGERESPONSE XML tag.

#### Functional groups

Table 1 on page 10 describes the functional groups that are supported by the CIM agent. This information is also returned to the client that makes an OPTIONS request to the CIM agent.

<i>Table 1: Functional groups for the CIM agent</i>		
<b>Functional group</b>	<b>Parameters</b>	<b>Supported or not supported</b>
Basic read	GetInstance EnumerateInstances EnumerateInstanceNames GetProperty	Supported
Basic write	SetProperty	Not Supported
Schema manipulation	CreateClass ModifyClass DeleteClass	Not Supported
Instance manipulation	CreateInstance ModifyInstance DeleteInstance	Not Supported
Association traversal	Associators AssociatorNames References RefernceNames	Supported
Qualifier read	GetQualifier EnumerateQualifiers	Supported
Qualifier manipulation	SetQualifier DeleteQualifier	Not Supported
Query execution	ExecQuery	Supported

The most current information for the communication methods is in the MOF documentation. The MOF documentation is in the `mo\` folder in the CIM agent installation directory.

## Obtaining a single class from the target namespace

The `GetClass` method returns a single class from the target namespace.

### Parameters

[Table 2 on page 10](#) describes the parameters of the `GetClass` method.

<i>Table 2: GetClass method parameters</i>		
<b>Parameter</b>	<b>Type</b>	<b>Description</b>
ClassName	String	Defines the name of the class you want to retrieve.
LocalOnly	Boolean	TRUE returns all properties, methods, and qualifiers that are overridden within the definition of the class.



<i>Table 2: GetClass method parameters (continued)</i>		
Parameter	Type	Description
IncludeQualifiers	Boolean	TRUE returns all qualifiers for the class, its properties, methods, or method parameters. FALSE returns no qualifiers.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN attribute of the class.

### Return values

If successful, a single class is returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_FAILED

## Obtaining a single instance from the target namespace

The GetInstance method returns a single instance from the target namespace.

### Parameters

Table 3 on page 11 describes the parameters of the GetInstance method.

<i>Table 3: GetInstance method parameters</i>		
Parameter	Type	Description
InstanceName	String	Defines the name of the instance to retrieve.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN attribute of the class.

### Return values

If successful, a single class is returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_FAILED

## Deleting a single instance from the target namespace

The DeleteInstance method deletes a single instance from the target namespace.

### Parameters

Table 4 on page 12 describes the parameters of the DeleteInstance method.

Table 4: DeleteInstance method parameters		
Parameter	Type	Description
InstanceName	String	Defines the name of the instance you want to delete.

#### Return values

The named instance is deleted or one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_FAILED

### Creating an instance in the target namespace

The CreateInstance method creates an instance in the target namespace. To use this method, the instance cannot exist.

The CreateInstance method is a standard Common Information Model (CIM) method. The IBM FlashSystem A9000 and A9000R Open API does not have any features that use this method.

#### Parameters

[Table 5 on page 12](#) describes the parameters of the CreateInstance method.

Table 5: CreateInstance method parameters		
Parameter	Type	Description
Instance	Object	The instance to be created. The instance must be based on a class that is already defined in the target namespace.

#### Return values

If successful, the specified instance is created. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_ALREADY\_EXISTS
- CIM\_ERR\_FAILED

### Modifying an existing instance in the target namespace

The ModifyInstance method modifies an existing instance in the target namespace. The instance must exist.

#### Parameters

[Table 6 on page 13](#) describes the parameters of the ModifyInstance method.

Table 6: <i>ModifyInstance</i> method parameters		
Parameter	Type	Description
Instance	Object	Defines the modified instance.

### Return values

If successful, the specified instance is updated. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_FAILED

## Enumerating classes within a defined target namespace

The EnumerateClasses method enumerates classes within a defined target namespace.

### Parameters

Table 7 on page 13 describes the parameters of the EnumerateClasses method.

Table 7: <i>EnumerateClasses</i> method parameters		
Parameter	Type	Description
ClassName	String	Defines the name of the class for which subclasses are to be returned. If this field is NULL, all base classes within the target namespace are returned.
DeepInheritance	Boolean	TRUE returns all subclasses of the specified class. FALSE returns only immediate child subclasses.
LocalOnly	Boolean	TRUE returns all properties, methods, and qualifiers, that are overridden within the definition of the class.
IncludeQualifiers	Boolean	TRUE returns all qualifiers for the class, its properties, methods, or method parameters. FALSE returns no qualifiers.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN of the class.

### Return values

If successful, zero or more classes (CIMClass) are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED

- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## Enumerating the names of subclasses of a class defined within the target namespace

The EnumerateClassNames method enumerates the names of subclasses of a class that is defined within the target namespace.

### Parameters

Table 8 on page 14 describes the parameters of the EnumerateClassNames method.

Table 8: EnumerateClassNames method parameters		
Parameter	Type	Description
ClassName	String	Defines the name of the class for which subclass names are to be returned. If this field is NULL, all base class names within the target namespace are returned.
DeepInheritance	Boolean	TRUE returns all subclass names of the specified class. FALSE returns only immediate child subclass names.

### Return values

If successful, zero or more class names are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## Enumerating instances of a defined class within the target namespace

The EnumerateInstances method enumerates instances of a defined class within the target namespace.

### Parameters

Table 9 on page 14 describes the parameters of the EnumerateInstances method.

Table 9: EnumerateInstances method parameters		
Parameter	Type	Description
ClassName	String	Defines the name of the class for which instances are to be returned.

<i>Table 9: EnumerateInstances method parameters (continued)</i>		
<b>Parameter</b>	<b>Type</b>	<b>Description</b>
DeepInheritance	Boolean	TRUE returns all instances and all properties of the instance, including the properties added by subclassing. FALSE returns only properties that are defined for the specified class.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN attribute of the class within the instance.

### **Return values**

If successful, zero or more instances (Objects) are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## **Enumerating the names of instances of a class within a target namespace**

The EnumerateInstanceNames method enumerates the names of the instances of a class within a target namespace.

### **Parameters**

[Table 10 on page 15](#) describes the parameter of the EnumerateInstanceNames method.

<i>Table 10: EnumerateInstanceNames method parameters</i>		
<b>Header</b>	<b>Header</b>	<b>Description</b>
ClassName	String	Defines the name of the class for which instance names are returned.

### **Return values**

If successful, zero or more names of instances are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## Processing a query against the target namespace

The ExecuteQuery method processes a query against the target namespace.

### Parameters

Table 11 on page 16 describes the parameters of the ExecuteQuery method.

Table 11: ExecuteQuery method parameters		
Parameter	Type	Description
QueryLanguage	String	Defines the query language in which the query parameter is expressed.
Query	String	Defines the query to be initiated.

### Return values

If successful, the method returns a table definition, followed by zero or more rows that correspond to the results of the query. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_NOT\_SUPPORTED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_QUERY\_LANGUAGE\_NOT\_SUPPORTED
- CIM\_ERR\_QUERY\_FEATURE\_NOT\_SUPPORTED
- CIM\_ERR\_INVALID\_QUERY
- CIM\_ERR\_FAILED

## Enumerating classes or instances that are associated with a specific Common Information Model (CIM) object

The Associators method enumerates classes or instances that are associated with a specific Common Information Model (CIM) object.

### Parameters

Table 12 on page 16 describes the parameters of the Associators method.

Table 12: Associators method parameters		
Parameter	Type	Description
ObjectName	String	Defines the class name or instance name that is the source of the association.
AssocClass	String	If not NULL, indicates that all objects must be associated with the source object through an instance of this class or one of its subclasses.
ResultClass	String	If not NULL, indicates that all returned objects must be instances of this class or one of its subclasses or be this class.

Table 12: Associators method parameters (continued)		
Parameter	Type	Description
Role	String	If not NULL, indicates that each return object must be associated with the source object that plays the specified role. The name of the property in the association class that refers to the source object must match the value of this parameter.
ResultRole	String	If not NULL, indicates that each returned object must be associated with the source object that plays the specified role. That is, the name of the property in the association class that refers to the returned object must match the value of this parameter.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN attribute of the class.

### Return values

If successful, zero or more classes (CIMClass) or instances (Objects) are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## Enumerating the names of classes or instances associated with a specific Common Information Model (CIM) object

The AssociatorNames method enumerates the names of the classes or instances that are associated with a specific Common Information Model (CIM) object.

### Parameters

[Table 13 on page 17](#) describes the parameters of the AssociatorNames method.

Table 13: Associators method parameters		
Parameter	Type	Description
ObjectName	String	Defines the class name or instance name that is the source of the association.
AssocClass	String	If not NULL, indicates that returned objects are associated with the source object through an instance of this class or one of its subclasses.
ResultClass	String	If not NULL, indicates that all returned object paths must identify instances of this class or one of its subclasses or must be this class.

*Table 13: Associators method parameters (continued)*

Parameter	Type	Description
Role	String	If not NULL, the name of the property in the association class that refers to the source object must match the value of this parameter.
ResultRole	String	If not NULL, the name of the property in the association class that refers to the return object must match the value of this parameter.

### Return values

If successful, zero or more class paths (CIMObjectPath) are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_FAILED

## Enumerating the association objects that refer to a specific target class or instance

The References method enumerates the association objects that refer to a particular target class or instance.

### Parameters

Table 14 on [page 18](#) describes the parameters of the References method.

*Table 14: References method parameters*

Parameter	Type	Description
ObjectName	String	Defines the class name or instance name whose referring objects are to be returned.
ResultClass	String	If not NULL, indicates that all returned objects must be instances of this class or one of its subclasses or must be this class.
Role	String	If not NULL, must be a valid property name. Each returned object must refer to the target object through a property whose name matches the value of this parameter.
IncludeClassOrigin	Boolean	TRUE returns the CLASSORIGIN attribute of the class.



### Return values

If successful, zero or more classes (CIMClass) or instances (Objects) are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_FAILED

## Enumerating the names of the association objects that refer to a specific target class or instance

The ReferenceNames method enumerates the names of the association objects that refer to a particular target class or instance.

### Parameters

Table 15 on page 19 describes the parameters of the ReferenceNames method.

Table 15: ReferenceNames method parameters		
Parameter	Type	Description
ObjectName	String	Defines the class name or instance name whose referring objects are to be returned.
ResultClass	String	If not NULL, indicates that all returned objects must be instances of this class or one of its subclasses or must be this class.
Role	String	If not NULL, must be a valid property name. Each returned object must refer to the target object through a property whose name matches the value of this parameter.

### Return values

If successful, the return value specifies the value of the requested property. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_NO\_SUCH\_PROPERTY
- CIM\_ERR\_FAILED

## Retrieving a single property value from an instance in the target namespace

The GetProperty method retrieves a single property value from an instance in the target namespace.

### Parameters

Table 16 on page 20 describes the parameters of the GetProperty method.

Table 16: GetProperty method parameters		
Parameter	Type	Description
InstanceName	String	Defines the name of the instance.
Property	String	The name of the property whose value is to be returned from the instance.

### Return values

If successful, the return value specifies the value of the requested property. Otherwise, one of the following return codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_NO\_SUCH\_PROPERTY
- CIM\_ERR\_FAILED

## Setting a single property value within an instance in the target namespace

The SetProperty method sets a single property value within an instance in the target namespace. But the IBM FlashSystem A9000 and A9000R Open API Common Information Model (CIM) agent does not have any features that use this method.

### Parameters

Table 17 on page 20 describes the parameters of the SetProperty method.

Table 17: SetProperty method parameters		
Parameter	Type	Description
InstanceName	String	Defines the name of the instance.
Property	String	The name of the property whose value is to be returned from the instance.

### Return values

If successful, the instance is updated. Otherwise, one of the following return codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER

- CIM\_ERR\_INVALID\_CLASS
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_NO\_SUCH\_PROPERTY
- CIM\_ERR\_TYPE\_MISMATCH
- CIM\_ERR\_FAILED

## Retrieving a single qualifier declaration from the target namespace

The GetQualifier method retrieves a single qualifier declaration from the target namespace.

### Parameters

Table 18 on page 21 describes the parameters of the GetQualifier method.

Table 18: GetQualifier method parameters		
Parameter	Type	Description
QualifierName	String	Defines the qualifier whose declaration is to be returned.

### Return values

If successful, the value of the qualifier is returned. Otherwise, one of the following return codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_FAILED

## Creating or modifying a qualifier declaration in the target namespace

The SetQualifier method creates or updates a qualifier declaration in the target namespace.

### Parameters

Table 19 on page 21 describes the parameters of the SetQualifier method.

Table 19: SetQualifier method parameters		
Parameter	Type	Description
QualifierDeclaration	Void	Defines the qualifier declaration to be added to the target namespace.

### Return values

If successful, the qualifier is updated in the target namespace. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER

- CIM\_ERR\_NOT\_FOUND
- CIM\_ERR\_FAILED

## Enumerating qualifier declarations from the target namespace

The EnumerateQualifiers method enumerates qualifier declarations from the target namespace.

There are no parameters for this method.

### Return values

If successful, zero or more qualifier declarations are returned. Otherwise, one of the following error codes is returned:

- CIM\_ERR\_ACCESS\_DENIED
- CIM\_ERR\_INVALID\_NAMESPACE
- CIM\_ERR\_INVALID\_PARAMETER
- CIM\_ERR\_FAILED

## Common Information Model (CIM) agent communication methods that cannot be used

The following Common Information Model (CIM) agent communication methods cannot be used because they are not supported by IBM FlashSystem A9000 and A9000R:

- [“DeleteClass method \(not supported\)” on page 22](#)
- [“CreateClass method \(not supported\)” on page 22](#)
- [“ModifyClass method \(not supported\)” on page 22](#)
- [“DeleteQualifier method \(not supported\)” on page 22](#)

### DeleteClass method (not supported)

The DeleteClass method deletes a single class from the target namespace, but it is not supported and cannot be used. The CIM\_ERR\_NOT\_SUPPORTED error code is returned to the client application if a request to process this operation is received.

### CreateClass method (not supported)

The CreateClass method creates a class from the target namespace, but it is not supported and cannot be used. The CIM\_ERR\_NOT\_SUPPORTED error code is returned to the client application if a request to process this operation is received.

### ModifyClass method (not supported)

The ModifyClass method modifies an existing class, but it is not supported and cannot be used. The CIM\_ERR\_NOT\_SUPPORTED error code is returned to the client application if a request to process this operation is received.

### DeleteQualifier method (not supported)

The DeleteQualifier method deletes a single class from the target namespace, but it is not supported and cannot be used. The CIM\_ERR\_NOT\_SUPPORTED error code is returned to the client application if a request to process this operation is received.

## Return error codes

---

The Common Information Model object manager (CIMOM) returns status to the client application.

The return status is sent to the client application in one of the following ways:

- Through HTTP status messages
- Through error codes that are contained within <METHODRESPONSE> or <IMETHODRESPONSE> XML tags.

Table 20 on page 23 describes the vendor-specific status codes that the CIMOM might return. For CIM standard return codes, see the CIM schema.

<i>Table 20: Return error codes for the CIMOM</i>		
<b>Code</b>	<b>Symbolic Name</b>	<b>Definition</b>
1	CIM_ERR_FAILED	A general error occurred that is not covered by a more specific error code.
2	CIM_ERR_ACCESS_DENIED	Access to a CIM resource was not available to the client.
3	CIM_ERR_INVALID_NAMESPACE	The target namespace does not exist.
4	CIM_ERR_INVALID_PARAMETER	One or more parameter values that are passed to the method were not valid.
5	CIM_ERR_INVALID_CLASS	The specified class does not exist.
6	CIM_ERR_NOT_FOUND	The requested object was not found.
7	CIM_ERR_NOT_SUPPORTED	The requested operation is not supported.
8	CIM_ERR_CLASS_HAS_CHILDREN	The operation cannot be carried out on this class because it has subclasses.
9	CIM_ERR_CLASS_HAS_INSTANCES	The operation cannot be carried out on this class because it has instances.
10	CIM_ERR_INVALID_SUPERCLASS	The operation cannot be carried out because the specified superclass does not exist.
11	CIM_ERR_ALREADY_EXISTS	The operation cannot be carried out because the object exists.
12	CIM_ERR_NO_SUCH_PROPERTY	The specified property does not exist.
13	CIM_ERR_TYPE_MISMATCH	The value that is supplied is not compatible with the type that is specified.
14	CIM_ERR_QUERY_LANGUAGE_NOT_SUPPORTED	The query language is not recognized or supported.
15	CIM_ERR_INVALID_QUERY	The query is not valid for the specified query language.
16	CIM_ERR_METHOD_NOT_AVAILABLE	The extrinsic method cannot be run.
17	CIM_ERR_METHOD_NOT_FOUND	The specified extrinsic method does not exist.
20	CIM_ERR_LOW_ON_MEMORY	There is not enough memory.
21	XMLERROR	An XML error occurred.
22	CIM_ERR_LISTNER_ALREADY_DEFINED	The listener is already defined.
23	CIM_ERR_INDICATION_NOT_COLLECTED	The indications are not collected.
24	CIM_ERR_NO_METHOD_NAME	The method name is null.
25	CIM_ERR_INVALID_QUALIFIER_DATATYPE	The data type qualifier is not valid.
26	CIM_ERR_NAMESPACE_NOT_IN_MANAGER	The namespace value is not found.
27	CIM_ERR_INSTANTIATE_FAILED	The instantiation failed.

Table 20: Return error codes for the CIMOM (continued)		
Code	Symbolic Name	Definition
28	CIM_ERR_FAILED_TO_LOCATE_INDICATION_HANDLER	The indication handler is not found.
29	CIM_ERR_IO_EXCEPTION	An I/O exception occurred.
30	CIM_ERR_COULD_NOT_DELETE_FILE	The file cannot be deleted.
31	INVALID_QUALIFIER_NAME	The qualifier name is null.
32	NO_QUALIFIER_VALUE	The qualifier value is null.
33	NO_SUCH_QUALIFIER1	There is no such qualifier.
34	NO_SUCH_QUALIFIER2	There is no such qualifier.
35	QUALIFIER_UNOVERRIDABLE	The qualifier cannot be overwritten.
36	SCOPE_ERROR	A scope error occurred.
37	TYPE_ERROR	A type error occurred.
38	CIM_ERR_MISSING_KEY	The key is missing.
39	CIM_ERR_KEY_CANNOT_MODIFY	The key cannot be modified.
40	CIM_ERR_NO_KEYS	There are no keys found.
41	CIM_ERR_KEYS_NOT_UNIQUE	The keys are not unique.
100	CIM_ERR_SET_CLASS_NOT_SUPPORTED	The set class operation is not supported.
101	CIM_ERR_SET_INSTANCE_NOT_SUPPORTED	The set instance operation is not supported.
102	CIM_ERR_QUALIFIER_NOT_FOUND	The qualifier value is not found.
103	CIM_ERR_QUALIFIERTYPE_NOT_FOUND	The qualifier type is not found.
104	CIM_ERR_CONNECTION_FAILURE	The connection failed.
105	CIM_ERR_FAIL_TO_WRITE_TO_SERVER	There is a fail to write to the server.
106	CIM_ERR_SERVER_NOT_SPECIFIED	The server is not specified.
107	CIM_ERR_INDICATION_ERROR	There is an indication processing error.
108	CIM_ERR_FAIL_TO_WRITE_TO_CIMOM	A write operation to the CIMOM failed.
109	CIM_ERR_SUBSCRIPTION_EXISTS	A subscription exists.
110	CIM_ERR_INVALID_SUBSCRIPTION_DEST	The subscription destination is not valid.
111	CIM_ERR_INVALID_FILTER_PATH	The filter path is not valid.
112	CIM_ERR_INVALID_HANDLER_PATH	The handler path is not valid.
113	CIM_ERR_NO_FILTER_INSTANCE	The filter instance is not found.
114	CIM_ERR_NO_HANDLER_INSTANCE	The handler instance is not found.
115	CIM_ERR_UNSUPPPORTED_FILTER	The filter that is referenced in the subscription is not supported.
116	CIM_ERR_INVALID_TRUSTSTORE	The CIMOM cannot be connected to because there is a bad or missing truststore or an incorrect truststore password.

<i>Table 20: Return error codes for the CIMOM (continued)</i>		
<b>Code</b>	<b>Symbolic Name</b>	<b>Definition</b>
117	CIM_ERR_ALREADY_CONNECTED	The CIMOM cannot be connected to because it is already connected.
118	CIM_ERR_UNKNOWN_SERVER	The server is unknown. The CIMOM cannot accept connections.
119	CIM_ERR_INVALID_CERTIFICATE	The correct certificate cannot be found in the truststore. The CIMOM cannot accept connections.





# Chapter 4. Functional profiles, diagrams, and methods

The following sections detail the functional profiles, diagrams, and methods of the Common Information Model (CIM).

The functional diagrams show specific functions that the CIM agent provides and illustrate the architecture of the CIM agent for IBM FlashSystem A9000 and A9000R systems. The method sections describe the uses of and parameters associated with the methods associated with each profile and matching diagram(s).

## Block Server performance profile

Block server performance is the Storage Management Initiative Specification (SMI-S) subprofile that describes how to present performance statistics. It uses a number of metrics with definitions that are standardized for all storage systems that are represented by SMI-S. These metrics are defined in the CIM\_BlockStorageStatisticalData class definition.

For the Common Information Model (CIM) agent for IBM FlashSystem A9000 and A9000R systems, statistics are only provided for the volumes and hosts.

Table 21 on page 27 provides details about the specific metrics that are supported by IBM FlashSystem A9000 and A9000R arrays and their components. In addition, the table provides a list of ElementType values that correspond to the components of IBM FlashSystem A9000 and A9000R arrays.

Table 21: Block Server Performance metrics			
	Equivalent CIM class	Corresponding ElementType value	Properties that are supplied in associated CIM_BlockStorage StatisticalData instances
Volume	IBMTSDS_SEVolumeStatistics	8	TotalIOs KBytesTransferred KBytesRead KBytesWritten ReadIOs ReadHitIOs WriteIOs WriteHitIOs IOTimeCounter ReadIOTimeCounter WriteIOTimeCounter ReadHitIOTimeCounter WriteHitIOTimeCounter

Table 21: Block Server Performance metrics (continued)			
	Equivalent CIM class	Corresponding ElementType value	Properties that are supplied in associated CIM_BlockStorage StatisticalData instances
Host	IBMTSDS_HostStatistics	4	TotalIOs KBytesTransferred KBytesRead KBytesWritten ReadIOs ReadHitIOs WriteIOs WriteHitIOs IOTimeCounter ReadIOTimeCounter WriteIOTimeCounter ReadHitIOTimeCounter WriteHitIOTimeCounter

## Block Server Performance object model

Two major categories of classes in the object model determine the way clients retrieve performance statistics.

The following categories are used:

- The first category is a set of classes where each instance of the class represents a single performance statistics record. For example, the statistics for a single volume.
- The second is a set of classes that are required to use an extrinsic method to retrieve a string that contains a batch of performance statistics.

See [Figure 3 on page 29](#) for the Block Server Performance Storage Management Initiative Specification (SMI-S) model for IBM FlashSystem A9000 and A9000R systems.

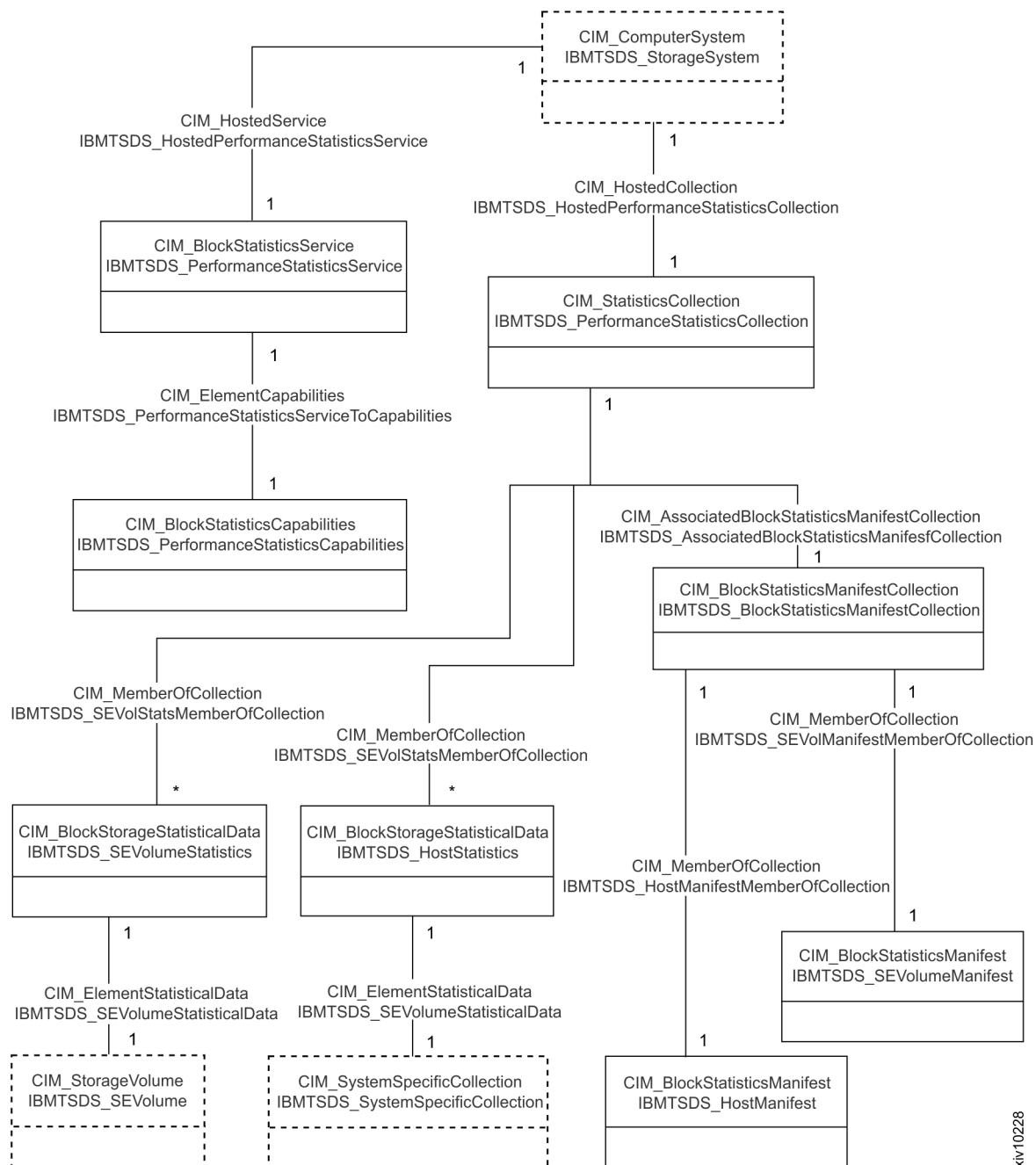


Figure 3: Block Server Performance SMI-S model for IBM FlashSystem A9000 and A9000R systems

The classes that inherit from CIM\_BlockStorageStatisticalData IBMTSDS\_SEVolumeStatistics and IBMTSDS\_HostStatistics are of the first category. Each instance contains properties that describe the performance statistics of a single instance of IBMTSDS\_SEVolume or IBMTSDS\_SystemSpecificCollection.

IBMTSDS\_PerformanceStatisticsCollection does not fall under either of the categories, but is a class that associates all of the Block Server Performance-related classes for a particular system.

The rest of the classes are under the second category. They are used for batch gathering of statistics. `IBMTSDS_PerformanceStatisticsService` contains the `GetStatisticsCollection` method, which you use to get a string representation of a batch of statistics instances. `IBMTSDS_PerformanceStatisticsCapabilities` defines what the performance statistics-related capabilities of the Common Information Model (CIM) agent are.

The classes that inherit from CIM\_BlockStatisticsManifest define filters for the statistics that are returned by GetStatisticsCollection.

IBMTSDS\_BlockStatisticsManifestCollection is a group of BlockStatisticsManifest instances that gets passed into the GetStatisticsCollection method. The BlockStatisticsManifest classes contain Boolean properties, one for each property in the corresponding BlockStorageStatisticalData class.

If a BlockStorageStatisticalData had a statistics property XXX, then the corresponding BlockStatisticsManifest class would have a boolean property called IncludeXXX. If IncludeXXX was set to true, GetStatisticsCollection would return the data for XXX. If IncludeXXX was set to false, GetStatisticsCollection would not return the data for XXX.

The IBM FlashSystem A9000 and A9000R CIM agent does not allow clients to pick and choose which properties it is interested in. Therefore, all of the IncludeXXX properties for the XXX properties that the CIM agent supports are set to true.

All statistics attributes are in units of kilobytes, milliseconds, or just a count. For example, the number of I/O operations. However, the statistics are just running counters. They do not provide information about rates. For example, I/O operations per second. When the counters reach an internal limit, they roll back to zero. If the client application is monitoring these statistics at a constant rate to calculate I/O rates, it must know when the counters roll back to zero.

## Block Server Performance methods

The following sections describe functional methods associated with the Block Server Performance profile and object model, including uses of and parameters used by each method.

- [“Obtaining performance statistics data” on page 30](#)
- [“Obtaining volume or host statistics” on page 31](#)

### Obtaining performance statistics data

The IBMTSDS\_PerformanceStatisticsService.GetStatisticsCollection method returns a string representation of a set of performance statistics data.

#### Parameters

The following list describes the parameters of the IBMTSDS\_PerformanceStatisticsService.GetStatisticsCollection method.

#### ElementTypes

An array of values that indicates the type of element that returns statistics. Volumes and hosts use the standard value in the Common Information Model (CIM) schema. If this parameter is left null, the default value 4 and 8 are used. See the BlockServerPerformance.mof file for details.

#### ManifestCollection (required)

A reference to an instance of the collection that represents a collection of BlockStatisticsManifest instances to use to filter the output. This parameter is not a reference to BlockStatisticsManifest. It is a ManifestCollection because the method can return statistics for multiple ElementTypes values at the same time, and each ElementType instance has its own BlockStatisticsManifest. The CIM agent currently supplies one instance of ManifestCollection, which is the default ManifestCollection.

#### Statistics

An output parameter that is an array of strings that represents a batch of performance data. Each array element is an instance of a statistics class. Each array element is formatted as a semicolon-separated list of values. The order of the returned values matches the properties definition order in the corresponding CIM\_BlockStatisticsManifest class in mof. For example, the statistics output parameter might display as follows:

```
IBM.2810-6000095-100916;8;20110118225313.890995+480;3111;166920;20864419;1426;  
1422;1119849;1078871;67382;1685;1673;19744570;19692176;99538;  
IBM.2810-6000095-100930;8;20110118225313.891038+480;8386;504537;76662400;1426;  
1422;1182723;1133669;67382;6960;6940;75479677;75360317;437155;
```

where each row is an array element. The `ElementType` value of each row is 8, which refers to volumes. So there are two instances of `VolumeStatistics` with attributes of each in semicolon-separated strings, in the order they are defined in the `IBMTSDS_SEVolumeManifest` class in `BlockServerPerformance.mof`.

```
{  
StatisticTime;TotalIOs;KBytesTransferred;IOTimeCounter;ReadIOs;ReadHitIOs;  
ReadIOTimeCounter;ReadHitIOTimeCounter;KBytesRead;WriteIOs;WriteHitIOs;  
WriteIOTimeCounter;WriteHitIOTimeCounter;KBytesWritten; }
```

### Obtaining volume or host statistics

You can use the `IBMTSDS_PerformanceStatisticsService.GetStatisticsCollection` method to obtain volume or host statistics.

1. Use `enumerateInstanceNames` on `IBMTSDS_PerformanceStatisticsService` and then save the reference.
2. Use `enumerateInstancesNames` on `IBMTSDS_BlockStatisticsManifestCollection` and then save the reference.
3. Use `enumerateInstances` on `IBMTSDS_VolumeManifest` (or `CIM_BlockStatisticsManifest` and look for the `ElementType` value 8) or `IBMTSDS_SystemSpecificCollection` (or `CIM_BlockStatisticsManifest` and look for the `ElementType` value 4), and then save `BulkFormat`.
4. Use `invokeMethod` on `GetStatisticsCollection` with the `ElementType` value set to 8 for volume or to 4 for host (if `ElementType` is null, `ElementType` values of 4 and 8 are assigned), the `StatisticsFormat` value set to 2, the `ManifestCollection` value obtained in the second step, and the `PerformanceStatisticsCollection` instance value that is obtained in the first step.
5. Check the output statistics (`Statistics`). For each array element in the statistics output, string tokenize on semicolon, and save the properties in a table. Check the properties.

## Block Services profile

---

The IBM FlashSystem A9000 and A9000R system architecture provides a means for you to customize the underlying resources.

The following definitions describe the various layers of abstraction that make this customization possible:

### The primordial storage pool

Logical entity that contains all the available unformatted or unprepared disk capacity on an array. Device Concrete `StoragePool` instances are allocated from `Primordial StoragePool`. IBM FlashSystem A9000 and A9000R arrays have only one `Primordial StoragePool` instance per array.

### Concrete storage pools

Logical entities that are allocated from `Primordial StoragePool`. Concrete `StoragePool` instances enable storage administrators to manage relationships between volumes and snapshots and to define separate capacity provisioning and snapshot requirements for separate applications and departments. Storage pools are not tied to specific physical resources, nor are they part of the data-distribution scheme.

IBM FlashSystem A9000 and A9000R support two types of concrete storage pools: `VirtualPool` and `SnapshotPool`. A `VirtualPool` instance is allocated from `Primordial StoragePool` directly. It contains volumes and at most one `SnapshotPool` instance.

A `SnapshotPool` instance is allocated from a `VirtualPool` instance and contains snapshots of volumes in the `VirtualPool` instance in which this `SnapshotPool` instance is located. A `VirtualPool` instance without its own `SnapshotPool` instance cannot contain snapshots

### Storage volumes

Logical units that can be mapped to a host or a cluster. Volumes are created from storage pools and are managed within the context of storage pools.



As required by SMI-S, each instance of storage pool has a corresponding instance of Storage Capabilities. Each instance of Storage Capabilities is associated with the instances of Storage Setting that are valid for creating volumes from the associated storage pool. See [Figure 5 on page 33](#) for the Block Services package with settings and capabilities model.

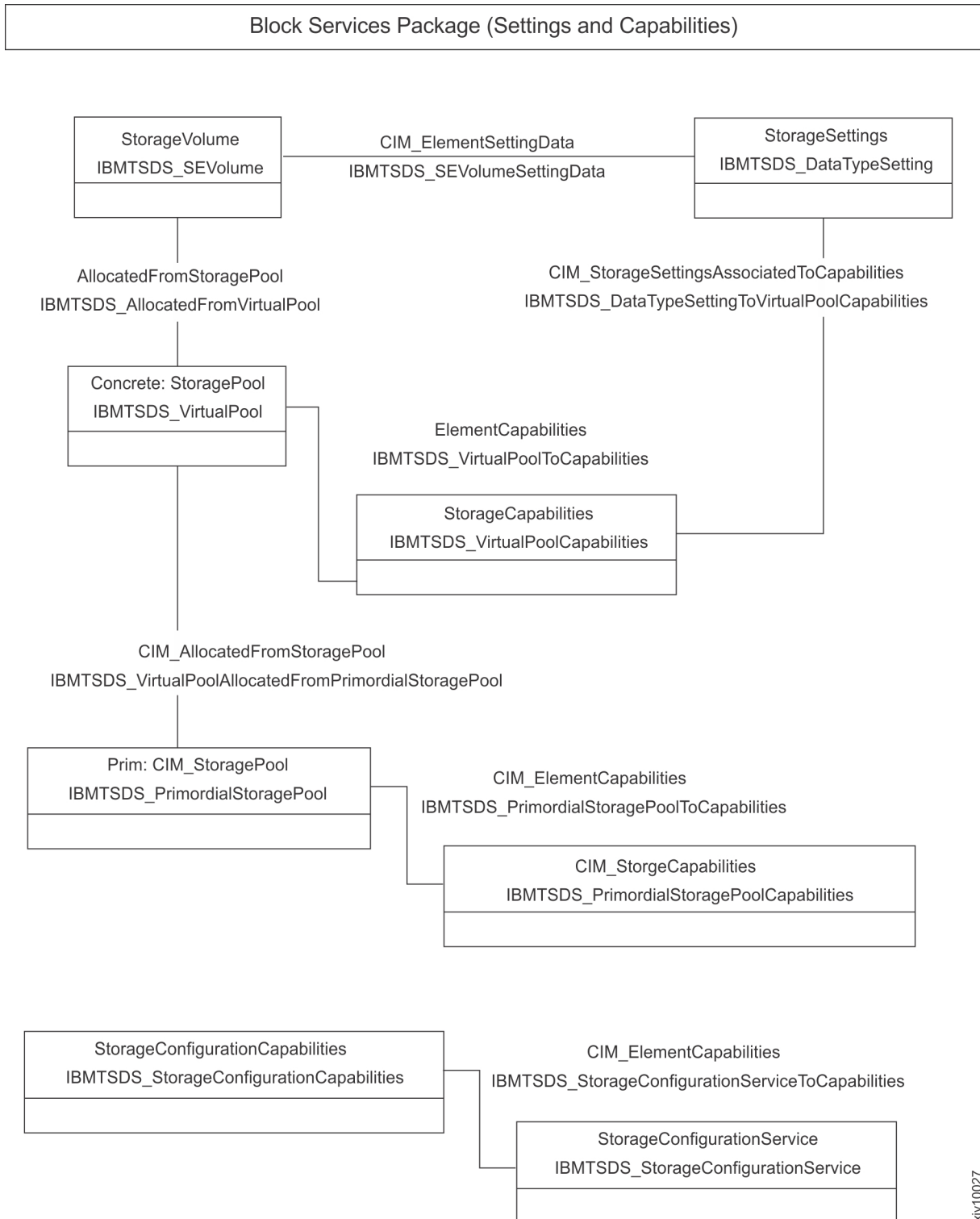


Figure 5: Block Services Package with Settings and Capabilities model

xiv10027

StorageSetting is a class that contains properties to specify the quality of service, such as DataRedundancy or parity layout. When you create a StorageVolume or Concrete StoragePool instance, a StorageSetting instance is supplied as the Goal parameter for the appropriate method.

The IBM FlashSystem A9000 and A9000R CIM agent has four CIM\_StorageSetting instances defined:

**IBMTSDS\_VirtualPoolSetting.InstanceID="IBMTSDS:IBM XIV Virtual Storage Pool Setting"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyStoragePool method to create or modify a VirtualPool instance.

**IBMTSDS\_SnapshotPoolSetting.InstanceID="IBMTSDS:IBM XIV Snapshot Pool Setting"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyStoragePool method to create or modify a SnapshotPool instance.

**IBMTSDS\_DataTypeSetting.InstanceID="IBMTSDS:XIVBlockSize"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyElementFromStoragePool and CreateOrModifyElementsFromStoragePool methods to create or modify StorageVolume instances with size in terms of GB.

The system allocates the soft volume size as the minimum number of discrete 1 GB increments needed to meet the requested volume size.

**IBMTSDS\_DataTypeSetting.InstanceID="IBMTSDS:SystemBlockSize"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyElementFromStoragePool and CreateOrModifyElementsFromStoragePool methods to create or modify StorageVolume instances with size in terms of blocks.

The volume's capacity is indicated as a discrete number of 512-byte blocks. The system allocates the soft volume size that is used within the StoragePool instance as the minimum number of discrete 1 GB increments to meet the requested size. However, the size that is reported to hosts is equivalent to the precise number of blocks defined.

For more information about the block services object model, see the BlockServices.mof file.

## Extrinsic Methods

Use an associator's CIM client request and the association class CIM\_ElementCapabilities between an instance of CIM\_StorageConfigurationService and the result class CIM\_StorageConfigurationCapabilities.

The SupportedSynchronousActions and SupportedAsynchronousActions properties identify whether the action is initiated synchronously or asynchronously. The IBM FlashSystem A9000 and A9000R CIM agent supports the following synchronous actions in the IBMTSDS\_StorageConfigurationService class:

Table 22: Synchronous actions	
Synchronous actions	IBM FlashSystem A9000 and A9000R CIM functions
StoragePool Creation	CreateOrModifyStoragePool
StoragePool Modification	CreateOrModifyStoragePool
StoragePool Deletion	DeleteStoragePool
StorageVolume Creation	CreateOrModifyElementFromStoragePool CreateOrModifyElementsFromStoragePool
StorageVolume Modification	CreateOrModifyElementFromStoragePool CreateOrModifyElementsFromStoragePool
StorageVolume Expansion	CreateOrModifyElementFromStoragePool CreateOrModifyElementsFromStoragePool



Table 22: Synchronous actions (continued)

Synchronous actions	IBM FlashSystem A9000 and A9000R CIM functions
StorageVolume Shrinking	CreateOrModifyElementFromStoragePool CreateorModifyElementsFromStoragePool
StorageVolume Deletion	Return to StoragePool
StorageVolume Deletion	Return Elements to StoragePool

The following functions can be used to determine what sizes of Concrete StoragePool and StorageVolume instances can be created:

**IBMTSDS\_PrimordialStoragePool.GetSupportedSizeRange**

Determines the supported physical size of Concrete StoragePool instances that can be created in Primordial StoragePool.

**IBMTSDS\_VirtualPool.GetSupportedSizeRange**

Determines the supported soft size of StorageVolume instances that can be created in a Concrete StoragePool instance.

## Block Services methods

The following sections describe functional methods associated with the Block Services profile and object model, including uses of and parameters used by each method.

- [“Creating or modifying a virtual pool or snapshot pool” on page 35](#)
- [“Deleting a storage pool” on page 36](#)
- [“Creating, modifying, and/or moving multiple volumes in a single method call” on page 36](#)
- [“Creating, modifying, and/or moving a single volume” on page 37](#)
- [“Deleting a single volume” on page 38](#)
- [“Deleting multiple volumes concurrently” on page 38](#)
- [“Determining sizes to use to create elements from the primordial storage pool” on page 38](#)
- [“Determining sizes to use to create elements from the virtual storage pool” on page 39](#)
- [“Creating a concrete storage pool” on page 39](#)
- [“Creating a storage volume” on page 39](#)

### Creating or modifying a virtual pool or snapshot pool

The IBMTSDS\_StorageConfigurationService.CreateOrModifyStoragePool method creates or modifies a VirtualPool instance or a SnapshotPool instance. The Primordial StoragePool instance cannot be created or modified.

### Parameters

The following list describes the parameters of the IBMTSDS\_StorageConfigurationService.CreateOrModifyStoragePool method.

#### Pool (required for modification)

As an input parameter, specifies whether you want to create or modify a pool. If you specify a reference to a pool, it indicates that you want to modify the pool. If the parameter is null, it indicates that you want to create pool.

#### InPools

InPools specifies from which pool to create a pool. To create a VirtualPool instance, only the object reference of IBMTSDS\_PrimordialStoragePool is allowed. All VirtualPool instances are created in Primordial StoragePool. To create a SnapshotPool instance, only the object reference of IBMTSDS\_VirtualPool is allowed because a SnapshotPool instance is created in a VirtualPool instance.

---

**Note:** The Common Information Model (CIM) schema defines this input parameter to be an array of strings that represent CIM object paths (COPs), and not actual references to objects.

---

### Goal

The Goal parameter represents the StorageSetting instance of the pool to be created. To create a VirtualPool instance, only the object reference IBMTSDS\_VirtualPoolSetting is allowed; to create a SnapshotPool instance, only the object reference IBMTSDS\_SnapshotPoolSetting is allowed. The Goal parameter also specifies the snapshot size value of the pool. See the BlockServer.mof file for details.

### ElementName

The ElementName property provides a means for you to set a meaningful name for the pool to be created or modified. If specified, limit it to 63 characters which can include letters, digits, blanks, -, \_, . and ~ characters. Blanks cannot be the beginning and ending characters. If not specified during pool creation, a random pool name is generated in the form of pool<random integer>.

---

**Note:** The name of the pool must be unique in the system. It cannot be a name that is already assigned to one of the other pools in the system.

---

### Size (required for creation)

As an input parameter, Size specifies the requested size of the pool. Null is not allowed for pool creation. As an output parameter, Size specifies the size achieved.

### Deleting a storage pool

The IBMTSDS\_StorageConfigurationService.DeleteStoragePool method is used to delete a single storage pool. This method requires a single parameter. Specify the reference to the Concrete StoragePool instance that is to be deleted. Only a single StoragePool instance can be deleted at a time.

---

**Note:** You must delete all volumes in the current pool before you delete a pool, and the Primordial StoragePool instance cannot be deleted.

---

### Creating, modifying, and/or moving multiple volumes in a single method call

IBMTSDS\_StorageConfigurationService.CreateOrModifyElementsFromStoragePool is a vendor-extension method that allows for the creation / modification / moving of multiple volumes in a single method call.

For large numbers of volumes, this method can be more efficient than calling CreateOrModifyElementFromStoragePool several times in a loop. The InPool, Goal, and ElementType input parameters are the same as in CreateOrModifyElementFromStoragePool.

### Parameters

The following definitions summarize the parameters that are different from CreateOrModifyElementFromStoragePool.

#### TheElements (required for volume modification or volume moving)

If this value is not null, it indicates that you want to modify each of the volumes specified.

#### Quantity (required for volume creation)

As input, represents the number of StorageVolume instances to be created. This parameter must be null when you modify StorageVolume instances. As output, represents the number of volumes that are created if successful. Or, if creation was not successful because of too many volumes and not enough capacity on the VirtualPool instance, it represents the number of volumes that can be created.

#### ElementNames

An array of the element names to assign to the various volumes that are being created or modified. Volume names can be supplied in two different ways:

- The first way supplies each volume name with one element of the value. The length must be equal to the value for Quantity when you create volumes. If not NULL, the length must be the same as the length specified in the TheElements parameter when you modify volumes. For example, to create

two volumes 'testVolume\_3' and 'testVolume\_4', set the ElementNames parameter's value as [ 'testVolume\_3' , 'testVolume\_4' ] and leave the value of the FirstSuffix parameter as null.

- The second way supplies the leading string of volumes names as the only one element. In this case, names of all volumes have the same format [LeadingString]\_[IncrementingNumber]. [LeadingString] is the value of this parameter, and the first [IncrementingNumber] is the value of the FirstSuffix parameter. The incremental number is 1. To create two volumes 'testVolume\_3' and 'testVolume\_4' in the first way, set the ElementNames parameter's value as [ 'testVolume' ] and set the value of the FirstSuffix parameter as 3.

### FirstSuffix

Represents the starting suffix number of volume names when you use the second way described to create volumes or modify volume names. Users can supply the starting [IncrementingNumber] with this parameter. If it is null, the default value is 1. When you use the first way described to create volumes or modify volume names, the value must be null.

---

**Note:** The FirstSuffix parameter is not in the Common Information Model (CIM) or Storage Management Initiative Specification (SMI-S) schema. This parameter is an IBM extension.

---

### ReturnCodes

Each volume that is created or modified can have a different error that is associated with it. This output parameter is an array of each of the individual return codes for each attempt.

### Creating, modifying, and/or moving a single volume

IBMTSDS\_StorageConfigurationService.CreateOrModifyElementFromStoragePool is the Common Information Model (CIM) and Storage Management Initiative Specification (SMI-S) standard method for creating or modifying a StorageVolume instance from a Concrete StoragePool instance.

This method can create or modify one volume at a time. This method is also used for moving a volume from one pool to another when the InPool parameter is specified with reference of the target pool.

### Parameters

The following definitions summarize how the IBMTSDS\_StorageConfigurationService.CreateOrModifyElementFromStoragePool method handles different values for the input parameters:

#### TheElement (*required for modification*)

As an input parameter, if the value is null, it indicates that you want to create a volume. If the value is not null, it indicates that you are trying to modify the specified volume. As an output parameter, TheElement references the volume that was created or modified.

#### ElementType

An enumeration that indicates what type of element is being created or modified. Only volume values are supported. So the value of ElementType is 2 (StorageVolume) or 5 (ThinlyProvisionedStorageVolume). See the BlockServer.mof file for details.

#### InPool (*required for creation and moving*)

A reference to the StoragePool instance which the volume is to be created in or moved to. It must be an instance of IBMTSDS\_VirtualPool.

#### Goal

A reference to the StorageSetting instance representing the DataType value of the volume. It must be a reference to an instance of IBMTSDS\_DataTypeSetting. IBMTSDS\_DataTypeSetting has two instances: IBMTSDS:SystemBlockSize and IBMTSDS:XIVBlockSize. In volume creation, if this parameter is null, the default value IBMTSDS:XIVBlockSize is used. In volume modification, if the Size parameter is specified, but this parameter is null, the default value IBMTSDS:XIVBlockSize is used, and the volume is resized.

#### Size

In creating or modifying a volume, this parameter specifies the wanted size (as input) and the size achieved (as an output parameter). In modifying the size of a volume, only size expansion is supported by default. If you want to shrink a volume, set the ForceShrink parameter as true.

## ForceShrink

In modifying a volume, this parameter specifies a volume is shrunk if the Size parameter is smaller than the actual size of the volume. If this parameter is null or set to false, shrinking a volume fails. To shrink a volume, set this parameter to true. The default value is false.



**Attention:** Shrinking a volume can cause data loss.

---

The ForceShrink parameter is not in the CIM or SMI-S schema. This parameter is an IBM extension.

## Locked

In modifying a volume/snapshot, this parameter specifies the volume/snapshot locking status to be locked or unlocked. If this parameter is null, the volume/snapshot locking status is not modified. If this parameter is set to 0, the volume/snapshot is unlocked. If this parameter is set to 1, the volume/snapshot is locked. Other values are invalid. When a volume/snapshot is locked, and other properties are modified in the same method, the volume/snapshot is locked in the last step. When a volume/snapshot is unlocked and other properties are modified in the same method, the volume/snapshot is unlocked in the first step.

---

**Note:** The Locked parameter is not in the CIM or SMI-S schema. This parameter is an IBM extension.

---

## ElementName

A descriptive name to assign to the volume.

---

**Note:** The volume name must be unique in the system. It cannot be a name that is already assigned to one of the other volumes in the system.

---

## Deleting a single volume

The IBMTSDS\_StorageConfigurationService.ReturnToStoragePool method is used to delete a single volume. The only input parameter that is required is a reference to the volume that you want to delete. Only one volume can be deleted at a time.

## Deleting multiple volumes concurrently

The IBMTSDS\_StorageConfigurationService.ReturnElementsToStoragePool method is used to delete multiple volumes concurrently. You can pass in an array of references of volumes to delete.

There is a ReturnCodes output parameter for the errors that can be generated for each individual volume that is being deleted.

---

**Note:** The ReturnElementsToStoragePool method is not in the Common Information Model (CIM) or Storage Management Initiative Specification (SMI-S) schema. This method is an IBM extension.

---

## Determining sizes to use to create elements from the primordial storage pool

The IBMTSDS\_PrimitiveStoragePool.GetSupportedSizeRange method is used to find out what sizes to use to create elements from the primordial storage pool.

CIM\_StoragePool includes the standard methods GetSupportedSizes and GetSupportedSizeRange to perform this task, but for IBMTSDS\_PrimitiveStoragePool, only GetSupportedSizeRange is supported. It can be used to retrieve the supported physical size of a virtual pool which can be created. The Goal input parameter represents the pool data type to use to calculate the possible valid sizes. It returns a range and a divisor of valid sizes.

---

**Note:** If there is no spare space to create a minimum pool, the returned minimum and maximum size of GetSupportedSizeRange for the storage pool is zero.

---

### Determining sizes to use to create elements from the virtual storage pool

The IBMTSDS\_VirtualPool.GetSupportedSizeRange method is used to find out what sizes to use to create elements from the virtual storage pool.

This method can be used to retrieve the supported size of virtual pool which can be created. The Goal input parameter represents the pool data type used to calculate the possible valid sizes. It returns a range and a divisor of valid sizes.

---

**Note:** If there is no spare space to create a minimum pool, the returned minimum and maximum size of GetSupportedSizeRange for the storage pool is zero.

---

### Creating a concrete storage pool

You can use the CreateOrModifyStoragePool method to create a Concrete StoragePool instance.

1. Call IBMTSDS\_PrimordialStoragePool.GetSupportedSizeRange with ElementType set to StoragePool. If the return code is 0 (success), present these values to the user and allow the user to pick a size.
2. Use InvokeMethod on IBMTSDS\_StorageConfigurationService.CreateOrModifyStoragePool and specify the ElementName, Size, and Goal input parameters.
3. Save the pool output parameter. It is the reference to the VirtualPool instance that was created.

### Creating a storage volume

You can use the CreateOrModifyElementFromStoragePool method to create a StorageVolume instance.

1. Use the associatorNames parameter from the Concrete StoragePool instance that was created to CIM\_StorageCapabilities to get a matching instance of IBMTSDS\_VirtualPoolCapabilities.
2. Use the associatorNames parameter from the IBMTSDS\_VirtualPoolCapabilities instance to CIM\_StorageSetting to get valid instances of IBMTSDS\_DataTypeSetting and allow user to pick the data type to use to create the volume.
3. Call IBMTSDS\_VirtualPool.GetSupportedSizeRange on the virtual pool that is created in Step 1 using the data type that the user picked as the Goal input parameter. If the return code is 0 (success), present these values to the user and allow the user to pick a size.
4. Use InvokeMethod on IBMTSDS\_StorageConfigurationService.CreateOrModifyElementFromStoragePool. Use the IBMTSDS\_VirtualPool instance created in Step 1, the IBMTSDS\_DataTypeSetting value, Size value, and ElementName value specified as the InPools, Size, Goal, and ElementName input parameters.
5. Save the TheElement output parameter and display it to the user, informing the user know that is the instance of the volume that was created.

## iSCSI Target Ports profile

---

The iSCSI Target Ports subprofile provides a standard model for representing iSCSI elements as Common Information Model (CIM) objects.

A client can obtain information about iSCSI capabilities and settings of an IBM FlashSystem A9000 and A9000R storage array. A client can also obtain information that is provided by iSCSI-capable arrays and the storage volumes that are exposed by those arrays on an iSCSI network:

- Nodes
- Network Portals
- SCSI Ports

iSCSI sessions and settings are not currently configurable through the IBM FlashSystem A9000 and A9000R Storage Management Initiative Specification (SMI-S) provider.

# iSCSI Target Ports Object Model

The Common Information Model (CIM) agent is designed to allow for the retrieval of information for each layer of abstraction.

See [Figure 6 on page 40](#) for the iSCSI Target Ports Storage Management Initiative Specification (SMI-S) model for IBM FlashSystem A9000 and A9000R systems.

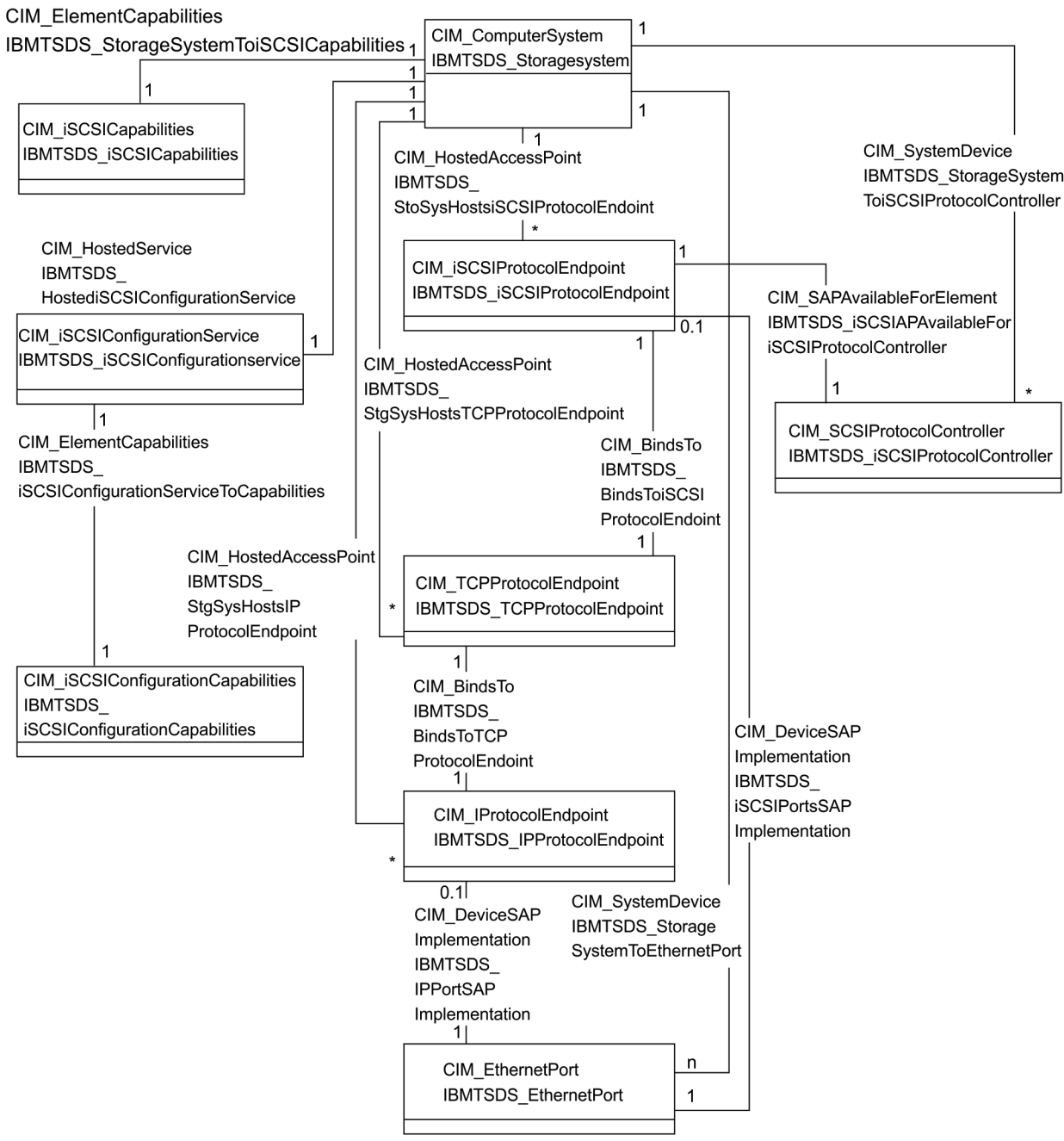


Figure 6: iSCSI Target Ports SMI-S model for IBM FlashSystem A9000 and A9000R systems

The following table provides a map of terminology from iSCSI standards and CIM class names on the previous UML diagram. iSCSI Session, Connection, and Portal Group are not supported by IBM FlashSystem A9000 and A9000R SMI-S.

xiv10229

Table 23: iSCSI terminology and CIM class names

iSCSI term	CIM class name	Description
Network entity	ComputerSystem	The Network Entity represents a device that is accessible from the IP network. A Network Entity has one or more Network Portals. Each Network Portal can be used to gain access to the IP network by some iSCSI Nodes that are contained in that Network Entity.
Node	SCSIProtocolController	The iSCSI Node represents a single iSCSI Target. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible through one or more Network Portals. An iSCSI Node is identified by its iSCSI Name. Separating the iSCSI Name from the addresses allows multiple iSCSI nodes to use the same address, and the same iSCSI node to use multiple addresses.
SCSI port	iSCSIProtocolEndpoint	A SCSI Port with an iSCSI service delivery subsystem. A collection of Network Portals that together acts as a SCSI Target or target.
Network Portal	TCPProtocolEndpoint, IPProtocolEndpoint, EthernetPort	A component of a Network Entity that has a TCP/IP network address. It might be used by an iSCSI Node within that Network Entity for the connections within one of its iSCSI sessions. A Network Portal in a Target is identified by its IP address.

### Network entities

Each IBMTSDS\_StorageSystem instance is considered a separate network entity, associated to an instance of CIM\_iSCSICapabilities through the CIM\_ElementCapabilities association; it is an iSCSI network entity visible on an iSCSI network.

Instances of CIM\_iSCSICapabilities contain the iSCSI specification versions and the authentication mechanisms that are supported by a network entity, which can be used to determine the capabilities of a storage array.

### iSCSI nodes

iSCSI Node instances are represented by instances of CIM\_SCSIProtocolController. This class can have many subclasses. The Name and NameFormat properties can be used to determine whether a CIM\_SCSIProtocolController instance is an iSCSI Node instance.

All CIM\_SCSIProtocolController instances that represent iSCSI Node instances are required to have a Name property whose value is an iSCSI Name value. Therefore, the NameFormat property must have a value of 3 (which maps to iSCSI Name).

On the IBM FlashSystem A9000 and A9000R arrays, iSCSI Node instances are represented by instances of IBMTSDS\_iSCSIProtocolController. IBMTSDS\_iSCSIProtocolController is a subclass of CIM\_SCSIProtocolController, which is associated to the first-level system (IBMTSDS\_StorageSystem) through CIM\_SystemDevice.

For IBM FlashSystem A9000 and A9000R arrays, each IBMTSDS\_StorageSystem entry has one Node entry for each IP Interface parameter. The iSCSI Name values for those nodes follow the IQN (iSCSI qualified name) format. The NameFormat and Name properties of the IBMTSDS\_iSCSIProtocolController instances are shown bolded in the following example.

```
instance of IBMTSDS_iSCSIProtocolController { SystemCreationClassName =
"IBMTSDS_StorageSystem"; SystemName = "IBM.2810-6000095";
CreationClassName = "IBMTSDS_iSCSIProtocolController"; Name =
"iqn.2005-10.com.xivstorage:000095.M_7_1"; NameFormat = 3; ElementName =
"M_7_1"; DeviceID = "IBM.2810-6000095-100955"; Caption =
"iSCSI Node of IP Interface M_7_1"; HealthState = 0; };
```

### iSCSI ports

Each iSCSI Node instance on an IBM FlashSystem A9000 and A9000R array is accessed through a single CIM\_iSCSIProtocolEndpoint instance. The CIM\_iSCSIProtocolEndpoint instances can be discovered by finding all iSCSI nodes, then following the CIM\_SAPAvailableForElement association from the iSCSI node (IBMTSDS\_iSCSIProtocolController) to CIM\_iSCSIProtocolEndpoint.

To find all of the CIM\_iSCSIProtocolEndpoint instances for a network entity, follow the CIM\_HostedAccessPoint association from the IBMTSDS\_StorageSystem instance to the CIM\_iSCSIProtocolEndpoint instances.

### Network Portal

There is a single network portal per IP Interface on an IBM FlashSystem A9000 and A9000R array, just as there is one iSCSI node and iSCSI port per IP Interface.

Network portals are represented by using the CIM\_TCPProtocolEndpoint, CIM\_IPProtocolEndpoint and CIM\_EthernetPort classes, which are associated through the CIM\_BindsTo association:

- The CIM\_IPProtocolEndpoint instance contains the IP address of the network portal in the IPv4Address property. The IPv6Address property is not currently supported.
- The CIM\_TCPProtocolEndpoint instance contains the TCP Port of the Network Portal in the PortNumber property.
- The CIM\_EthernetPort instance contains the physical properties of the Network Portal in the PermanentAddress, DeviceID, FullDuplex, Speed, and OtherIdentifyingInfo properties.

### iSCSI Target Ports methods

The following sections describe functional methods associated with the iSCSI Target Ports profile and object model, including uses of and parameters used by each method.

- [“Creating an IP endpoint” on page 42](#)
- [“Modifying an IP endpoint” on page 43](#)
- [“Deleting an IP endpoint” on page 44](#)

#### Creating an IP endpoint

You can use IBMTSDS\_iSCSIConfigurationService.CreateIPProtocolEndpoint method to create an IP endpoint.

1. Obtain the reference (CIMObjectPath) of an IBMTSDS\_iSCSIConfigurationService instance. The instance is associated with the IBMTSDS\_StorageSystem instance in which you receive the performance statistics by traversing the IBMTSDS\_HostedISCSIConfigurationService association.
2. Invoke the IBMTSDS\_iSCSIConfigurationService.CreateIPProtocolEndpoint method to create an IP endpoint with specified values of parameters ElementName, ModuleNumber, PortNumber, IPv4Address, IPv4Gateway, IPv4SubnetMask, and MTU.
3. After it is successfully completed, check the value of IPEndpoint representing the IP endpoint that was created.



### **Parameters**

The following list describes the parameters of the IBMTSDS\_iSCSIConfigurationService.CreateIPProtocolEndpoint method.

#### **ElementName (required)**

A meaningful name for the IP endpoint. It cannot be empty or null.

#### **ModuleNumber (required)**

An identifier that indicates the module that contains the IP endpoint. It cannot be 0 or null.

#### **PortNumber (required)**

Port identifier for this IP endpoint on this module. The value cannot be a value other than 1, 2, 3, or 4 because each module has four ports.

#### **IPv4Address (required)**

IPv4 address for this port.

#### **IPv4Gateway (required)**

IPv4 gateway for this port.

#### **IPv4SubnetMask (required)**

IPv4 subnet mask for this port.

#### **MTU**

MTU for this port. If it is null, default value 4500 is used.

#### **IPEndpoint**

The reference to a point that was created.

### **Modifying an IP endpoint**

You can use IBMTSDS\_iSCSIConfigurationService.ModifyIPProtocolEndpoint method to create an IP endpoint.

1. Obtain the reference (CIMObjectPath) of an IBMTSDS\_iSCSIConfigurationService instance that is associated with the IBMTSDS\_StorageSystem instance in which you receive the performance statistics by traversing the IBMTSDS\_HostedISCSIConfigurationService association.
2. Obtain the reference (CIMObjectPath) of an IBMTSDS\_IPEndPoint instance which is to be modified.
3. Invoke the IBMTSDS\_iSCSIConfigurationService.ModifyIPProtocolEndpoint method to modify an IP endpoint with values of parameters ElementName, IPv4Address, IPv4Gateway, IPv4SubnetMask, MTU, and IPEndpoint.
4. After it is successfully completed, the IP endpoint will be modified.

### **Parameters**

The following list describes the parameters of the IBMTSDS\_iSCSIConfigurationService.ModifyIPProtocolEndpoint method.

---

**Note:** The method ModifyIPProtocolEndpoint is not in the Common Information Model (CIM)/Storage Management Initiative Specification (SMI-S) schema but an IBM extension.

---

#### **ElementName**

A new meaningful name for the IP endpoint.

#### **ModuleNumber**

An identifier that indicates the new module that contains the IP endpoint.

#### **PortNumber**

The new port identifier for this IP endpoint. If it is set, the value cannot be a value other than 1, 2, 3, or 4 because each module has four ports.

#### **IPv4Address**

A new IPv4 address for this port.

#### **IPv4Gateway**

A new IPv4 gateway for this port.

**IPv4SubnetMask**

A new IPv4 subnet mask for this port.

**MTU**

A new MTU for this port.

**IPEndpoint**

The reference to a point to modify.

**Deleting an IP endpoint**

You can use the IBMTSDS\_iSCSIConfigurationService.DeleteIPProtocolEndpoint method to delete an IP endpoint.

1. Obtain the reference (CIMObjectPath) of an IBMTSDS\_iSCSIConfigurationService instance. Ensure that the reference is associated with the IBMTSDS\_StorageSystem instance in which you receive the performance statistics by traversing the IBMTSDS\_HostedISCSIConfigurationService association.
2. Obtain the reference (CIMObjectPath) of an IBMTSDS\_IPEndPoint instance which is to be deleted.
3. Invoke the IBMTSDS\_iSCSIConfigurationService.DeleteIPProtocolEndpoint method and set IPEndpoint as the reference obtained in step 2.
4. After it is successfully completed, the IP endpoint is deleted.

**Parameters**

The only input parameter that is required for the IBMTSDS\_iSCSIConfigurationService.DeleteIPProtocolEndpoint method is a reference to the PortocolEndPoint instance that you want to delete. Only one PortocolEndPoint instance can be deleted at a time.

---

**Note:** The method DeleteIPProtocolEndpoint is not in the Common Information Model (CIM)/Storage Management Initiative Specification (SMI-S) schema but an IBM extension.

---

## Masking and Mapping profile

---

You can control which client hosts can see and access storage volumes by mapping and masking client hosts. The Storage Networking Industry Association (SNIA) Storage Management Initiative Specification (SMI-S) specifications define the Mapping and Masking profile to support this function.

**LUN masking**

The process of configuring software in SAN nodes to determine which hosts have access to exported drives (volumes).

LUN masking for IBM FlashSystem A9000 and A9000R systems is storage-based port mapping.

**LUN mapping**

The process of creating a disk resource and defining its external access paths by using a LUN. The LUN is then mapped to an external ID descriptor. For example, a SCSI port or a target ID. To ensure uninterrupted data availability, map a logical volume to allow access from multiple ports, target IDs or both.

The IBM FlashSystem A9000 and A9000R system architecture for masking and mapping involves the following objects on two kinds of devices:

- Objects on IBM FlashSystem A9000 and A9000R systems:
  - Storage device I/O ports (FC ports and iSCSI ports)
  - Storage volumes
- Objects on the host system
  - Host ports (HBAs)
  - Hosts (host systems)
  - Clusters

Figure 7 on page 45 provides an example of how IBM FlashSystem A9000 and A9000R systems use masking and mapping.

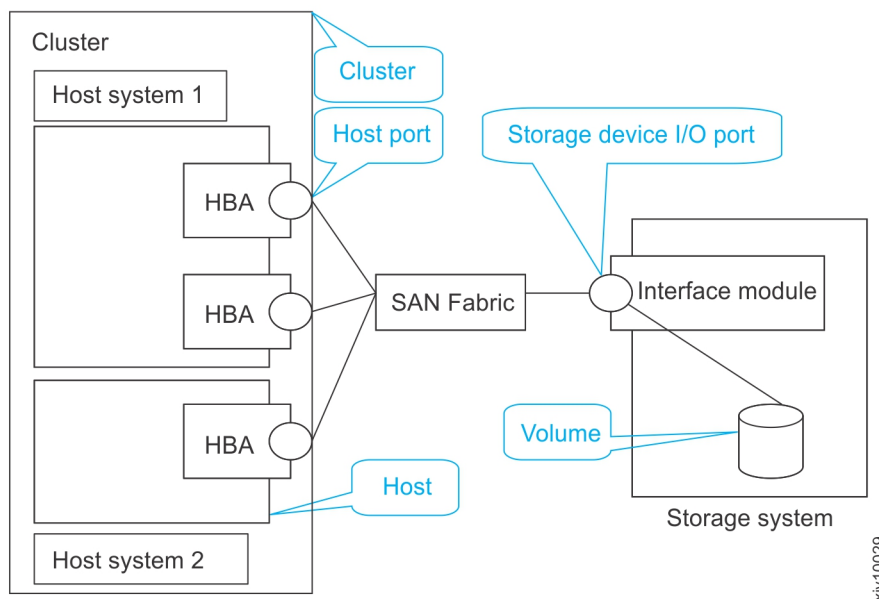


Figure 7: Masking and mapping physical model in IBM FlashSystem A9000 and A9000R systems

The following list defines each of the objects that are shown in [Figure 7 on page 45](#).

#### Storage device I/O ports

Ports on IBM FlashSystem A9000 and A9000R systems that represent a Fibre Channel port or an iSCSI port that is used for host I/O operations. Storage device I/O ports are sometimes referred to as target ports.

#### Volumes

Objects on IBM FlashSystem A9000 and A9000R systems that are mapped to host or a cluster. One volume cannot be mapped to multiple hosts that are not part of the same cluster.

#### Host ports

Objects that represent a single Fibre Channel port or an iSCSI port on the host system. The architecture does not track host systems. If a host system has multiple host ports, a host port object must be created for each port.

#### Hosts

Systems with a set of initiator ports that are used to access storage on the device. For each host system, you can create a host object, add host port objects, and then map the host to volumes through the IBM Hyper-Scale Manager GUI, command-line interface (CLI), or Common Information Model (CIM) agent.

#### Clusters

Groups of hosts. For IBM FlashSystem A9000 and A9000R systems, multiple hosts can see the same set of volumes. For a host system cluster, you can create a cluster object, add host objects to it, and then map the cluster object to multiple volumes through the IBM Hyper-Scale Manager GUI, CLI, or CIM agent. Clusters are required when more than one host can access the same volume.

After you set up the physical network and create a host object, add Fibre Channel ports or iSCSI ports to the host. Map the volumes to the host so that the volume can be used for I/O operations.

#### Mapping hosts to a volume

You can map a host to volumes. A host can contain several initiator ports that include Fibre Channel ports and iSCSI ports. In this case, create a host, add the initiator ports, and then map the host to a volume.

Through this mapping, the host system is able to access the mapped volume through the included initiator ports.

### Mapping clusters to a volume

You can map a cluster to a volume when several hosts must be mapped to the same volume. In this case, create a cluster and add hosts, and then map the cluster to volumes. Through this mapping, the volume is mapped to all hosts that are contained in the cluster. If a host is included in a cluster, it cannot be mapped to another volume separately.

See the *IBM FlashSystem A9000 and A9000R: Architecture, Implementation, and Usage* and the *IBM FlashSystem A9000 Product Overview* or *IBM FlashSystem A9000R Product Overview* for more information about the architecture of IBM FlashSystem A9000 and A9000R systems.

### Masking and Mapping object model

The Masking and Mapping profile provides an interface to specify which hosts can see which volumes and through which target ports.

For IBM FlashSystem A9000 and A9000R host and cluster mappings, specifying target ports for the view is not supported. Target ports cannot be selected when you configure host mappings, but can be configured by zoning configurations on the switch. The target ports that are involved in all views can be displayed by enumerating the instances of CIM\_ProtocolControllerForPort class. See [Figure 8](#) on [page 46](#).

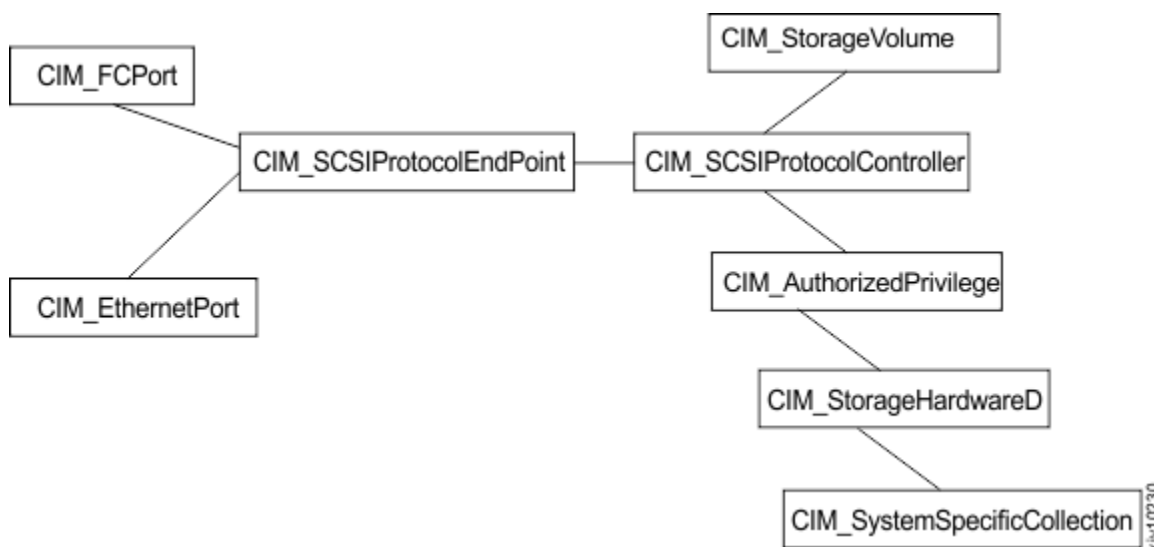


Figure 8: Masking and mapping object model in SMI-S

Table 24 on [page 46](#) describes the masking and mapping classes that are available. For more information about the masking and mapping object model, see the `MaskingMapping.mof` file. The MOF documentation is in the `mof` folder in the Common Information Model (CIM) agent installation directory.

Table 24: Masking and mapping classes		
CIM class name	Description	IBM FlashSystem A9000 and A9000R CIM class name
CIM_FCPort	A Fibre Channel port on the system	IBMTSDS_FCPort
CIM_EthernetPort	An Ethernet port on the system	IBMTSDS_EthernetPort

<i>Table 24: Masking and mapping classes (continued)</i>		
<b>CIM class name</b>	<b>Description</b>	<b>IBM FlashSystem A9000 and A9000R CIM class name</b>
CIM_SCSIProtocolEndPoint	The Fibre Channel or iSCSI port that is used in the mapping	IBMTSDS_SCSIProtocolEndpoint IBMTSDS_iSCSIProtocolEndpoint
CIM_SCSIProtocolController	A logical entity that represents a host and the relationship to the volumes	IBMTSDS_SCSIProtocolController
CIM_StorageVolume	A storage volume on the system	IBMTSDS_SEVolume
CIM_AuthorizedPrivilege	A logical entity that represents the access permissions for a set of volumes	IBMTSDS_Privilege
CIM_StorageHardwareID	An initiator port on the host system	IBMTSDS_StorageHardwareID
CIM_SystemSpecificCollection	A logical entity which represents a collection of StorageHardwareID (host)	IBMTSDS_SystemSpecificCollection
CIM_Cluster	A logical entity which represents a collection of SystemSpecificCollection (hosts)	IBMTSDS_Cluster

The IBM FlashSystem A9000 and A9000R CIM agent provides a full implementation of masking and mapping subprofile including the following items:

- iSCSI port support
- StorageHardwareID instance creation and deletion
- HardwareIDCollection instance creation and modification
- ExposePaths and HidePaths methods

### **iSCSI port support**

By default, IBM FlashSystem A9000 and A9000R ships equipped with six iSCSI ports. Three of the interface modules support iSCSI, with two ports in each module. You can refer to IBM FlashSystem A9000 and A9000R Redbook for detailed information. The IBM FlashSystem A9000 and A9000R CIM agent include the full iSCSI support by implementation of the iSCSI target ports profile and the masking and mapping profile.

### **StorageHardwareID instance manipulation**

A StorageHardwareID instance represents the host-side initiator that logs in to a storage area network. The IBM FlashSystem A9000 and A9000R SMI-S provider supports host-side initiators that contain Fibre Channel or iSCSI host ports.

A StorageHardwareIDManagementService instance allows a CIM client to locate StorageHardwareID instances known to a storage array, and create, modify, and delete them.

- The CreateStorageHardwareID method creates a StorageHardwareID instance. On IBM FlashSystem A9000 and A9000R, a host is created which contains the specified initiator port.
- The DeleteStorageHardwareID method deletes a StorageHardwareID instance. On IBM FlashSystem A9000 and A9000R, the initiator port is removed from the host that contains it.

## HardwareIDCollection instance manipulation

A HardwareIDCollection instance represents a collection of StorageHardwareID instances (host-side initiators) that log in to a storage area network.

A StorageHardwareIDManagementService instance allows a CIM client to locate HardwareIDCollection instances known to a storage array, create a HardwareIDCollection instance, and remove HardwareID instances from a HardwareIDCollection instance.

- The CreateHardwareIDCollection method creates a HardwareIDCollection instance with specified ElementName and Setting values.
- The AddHardwareIDsToCollection method adds StorageHardwareID instances to a HardwareIDCollection instance.

---

**Note:** The methods ModifyHardwareIDCollection, DeleteHardwareIDCollection, and RemoveHardwareIDsFromCollection are not in the CIM or SMI-S schema. These methods are an IBM extension.

---

## ExposePaths and HidePaths methods

An SCSIProtocolController instance represents a view of volumes that can be assigned access to a StorageHardwareID instance. You can use the ControllerConfigurationService method to locate SCSIProtocolController instances or the ExposePaths and HidePaths methods from StorageHardwareID instances to locate SCSIProtocolController instances.

A privilege determines the access rights between the StorageHardwareID instance and the SCSIProtocolController instance. The IBM FlashSystem A9000 and A9000R SMI-S provider supports read/write access. A privilege can be located by using the PrivilegeManagementService method.

## Masking and Mapping methods

The following sections describe functional methods associated with the Masking and Mapping profile and object model, including uses of and parameters used by each method.

- [“Creating a storage hardware ID” on page 48](#)
- [“Deleting a storage hardware ID” on page 49](#)
- [“Creating a hardware ID collection” on page 49](#)
- [“Adding hardware IDs to a collection” on page 49](#)
- [“Assigning volumes to a storage hardware ID” on page 50](#)
- [“Deleting multiple volumes concurrently” on page 38](#)
- [“Removing access to volumes from a storage hardware ID” on page 51](#)
- [“Deleting an existing protocol controller” on page 51](#)
- [“Example configuration procedures using the Masking and Mapping profile” on page 52](#)

### Creating a storage hardware ID

You can use the CreateStorageHardwareID method to create a StorageHardwareID instance. This method creates a host in the IBM FlashSystem A9000 or IBM FlashSystem A9000R device that contains the initiator port that is specified by StorageID value.

### Parameters

The following list describes the parameters of the CreateStorageHardwareID method.

#### Setting

An input parameter that represents the operating system of the port. It must be an instance of IBMTSDS\_StorageClientSettingData. Each instance represents a different operating system. Only the values Standard, HPUX, and z/VM are supported by the IBM FlashSystem A9000 and A9000R Common Information Model (CIM) agent. If it is left null, Standard is used by default.

**IDType**

An input parameter that represents the type of port. To create a HardwareID instance representing a Fibre Channel port, specify 2; to create a HardwareID instance representing an iSCSI port, specify 5.

**StorageID**

An input parameter that represents the ID of the port. If the IDType value is 2, specify it as the port's WWN; if the IDType value is 5, specify it as the port's IQN.

**HardwareID**

An output parameter; the reference to the created StorageHardwareID instance.

**Deleting a storage hardware ID**

Use the DeleteStorageHardwareID method to delete StorageHardwareID instances.

**Parameters**

The following list describes the parameters of the DeleteStorageHardwareID method.

**HardwareID**

An input parameter that represents a reference to the StorageHardwareID instance to delete.

**Creating a hardware ID collection**

You can use the CreateHardwareIDCollection method to create HardwareIDCollection instances.

**Parameters**

The following list describes the parameters of the CreateHardwareIDCollection method.

**ElementName (required)**

An input parameter that represents a meaningful name for the HardwareIDCollection instance that is being created. The ElementName must be unique in the system. It cannot be a name that is already assigned to one of the other HardwareIDCollection instances in the system. If not specified, the Common Information Model (CIM) generates a random name.

**HardwareIDs**

An input parameter that represents the ID of the ports that are contained by the HardwareIDCollection instance. If specified, each ID must be the WWN of a Fibre Channel port or the IQN of an iSCSI port. If not specified, an empty HardwareIDCollection instance is created.

**HardwareIDCollection**

An output parameter that represents the reference to the created HardwareIDCollection instance.

**Adding hardware IDs to a collection**

You can use the AddHardwareIDsToCollection method to add StorageHardwareID instances (representing initiator ports) into a SystemSpecificCollection instance (representing a host). The operation fails if the port already belongs to another host. The operation succeeds if the port already belongs to the specified host, but it has no effect.

**Parameters**

The following list describes the parameters of the AddHardwareIDsToCollection method.

**HardwareIDs (required)**

An input parameter that represents an array of strings that contain StorageID values of StorageHardwareID instances that become members of the HardwareIDCollection instance.

**HardwareIDCollection (required)**

An input parameter that represents the reference to the SystemSpecificCollection instance into which the HardwareID values are added.

### Assigning volumes to a storage hardware ID

Use the IBMTSDS\_ControllerConfigurationService.ExposePaths method to assign volumes to a StorageHardwareID instance representing a host or cluster in an IBM FlashSystem A9000 or IBM FlashSystem A9000R system.

ExposePaths initiates the mapping and masking operation in one method call. It produces a list of volumes to a list of initiators, through one or more SCSIProtocolController instances (SPCs). It supports creating or modifying SPCs depending on different specified parameters.

The following table gives an overview of ExposePaths use cases and associated parameters and parameter values.

Table 25: ExposePaths use cases, parameters, and parameter values						
Use cases	LUNames	Initiator PortIDs	Target PortIDs	Device Numbers	Device Accesses	ProtocolControllers (on input)
Create a view	Optional	Optional	NULL	Optional	Optional	NULL
Add LUNs to a view	Mandatory		NULL	Optional	Optional	contains a single SPC reference
Add initiator IDs to a view	NULL	Mandatory	NULL	NULL	NULL	contains a single SPC reference

### Parameters (detailed)

The following list describes the parameters of the IBMTSDS\_ControllerConfigurationService.ExposePaths method in more detail.

#### LUNames (required)

A string array input parameter that represents the volumes to map to the SPC. They are not references to volume instances, but strings that match the Name property of the IBMTSDS\_SEVolume (CIM\_StorageVolume) instance, which is the WWN of an IBM FlashSystem A9000 or IBM FlashSystem A9000R volume.

#### InitiatorPortIDs (required if StorageHardwareID has no volumes previously mapped)

A string array input parameter that represents the initiator ports to be added to an SPC. They are not references to StorageHardwareID instances, but the StorageID property of IBMTSDS\_StorageHardwareID instance.

#### DeviceNumbers (required)

A string array input parameter that represents the LUN IDs. Each item in this array must be a number 0 - 511. The number of items is the same as the number of items that are specified by the LUNames parameter. If not specified, the Common Information Model (CIM) assigns unused LUN IDs for the operation.

#### DeviceAccesses

A unit16 array input parameter that represents the access rights to give to the StorageHardwareID instance and to the volumes specified in LUNames. Read/write access is supported. It must be an array the same size as that specified for LUNames, where each value is 2.

#### ProtocolControllers (required if adding volumes to a StorageHardwareID that was already previously mapped to volumes)

As an input parameter, it represents the SCSIProtocolController instance to be modified. As an output parameter, it represents the SCSIProtocolController instance that is being modified or created.



### Removing access to volumes from a storage hardware ID

The IBMTSDS\_ControllerConfigurationService.HidePaths method is the inverse of the ExposePaths method; it is used to remove access to volumes from a StorageHardwareID instance.

The following table gives an overview of HidePaths use cases and associated parameters and parameter values.

Table 26: HidePaths use cases, parameters, and parameter values				
Use case	LUNames	InitiatorPortIDs	TargetPortIDs	ProtocolControllers (on input)
Remove LUs from a view	Mandatory	NULL	NULL	contains a single SPC reference
Remove initiator IDs from a view	NULL	Mandatory	NULL	contains a single SPC reference
Hide full paths from a view	Mandatory	Mandatory	NULL	contains a single SPC reference

### Parameters (detailed)

The following list describes the parameters of the IBMTSDS\_ControllerConfigurationService.HidePaths method in more detail.

#### LUNames (required for removing access to volumes)

A string array input parameter that represents the volumes to unmap from the StorageHardwareID instance. They are not references to volume instances, but are strings that match the Name property of the IBMTSDS\_SEVolume (CIM\_StorageVolume) instance.

#### InitiatorPortIDs (required for removing access to initiator ports)

A string array input parameter that represents the initiator port that you want to remove from the SCSIProtocolController view.

#### ProtocolControllers (required)

As an input parameter, it represents the SCSIProtocolController instance to be modified. As an output parameter, it represents the SCSIProtocolController instance that was modified or deleted.

### Deleting an existing protocol controller

The IBMTSDS\_ControllerConfigurationService.DeleteProtocolController method deletes an existing ProtocolController. In IBM FlashSystem A9000 and A9000R systems, the corresponding host and all the host ports that are contained by the host are deleted.

### Parameters

The following list describes the parameters of the IBMTSDS\_ControllerConfigurationService.DeleteProtocolController method.

#### ProtocolController (required)

An input parameter that represents the ProtocolController instance to be deleted.

#### DeleteUnits

An input parameter that represents the requirement to delete the volumes that are mapped to the ProtocolController instance. If true is specified, the mapped volumes are deleted.

### **Example configuration procedures using the Masking and Mapping profile**

The following sections provide example procedures to perform various configuration tasks using the Masking and Mapping profile, including the common tasks of configuring host and LUN mapping in an IBM FlashSystem A9000 or IBM FlashSystem A9000R system.

#### **Mapping FC and iSCSI ports to volumes (example):**

1. Obtain the WWN of the host FC ports, the IQN of the host iSCSI ports, and the names of the volumes to be mapped to.
2. Use InvokeMethod on IBMTSDS\_ControllerConfigurationService.ExposePaths, with InitiatorPortIDs set to the IDs of the host ports, LUNames set to the volume names, and DeviceAccesses set to 2.

#### **Adding volumes to an existing SCSI protocol controller (SPC) (example):**

1. Obtain the host reference and the names of the volumes to be added.
2. Find the IBMTSDS\_SCSIProtocolController instance that matches the selected host.
3. Use InvokeMethod on IBMTSDS\_ControllerConfigurationService.ExposePaths, with LUNames set to the volume names, DeviceAccesses set to 2, and ProtocolControllers set to the SCSIProtocolController instance name.

#### **Adding initiator ports to an existing SPC (example):**

1. Obtain the IDs of the initiator ports.
2. Find the IBMTSDS\_SCSIProtocolController instance.
3. Use InvokeMethod on IBMTSDS\_ControllerConfigurationService.ExposePaths, with InitiatorPortIDs set to the initiator port IDs, DeviceAccesses set to 2, and ProtocolControllers set to the SCSIProtocolController instance name.

#### **Removing volumes from an SPC (example):**

1. Obtain the SPC reference and names of the volumes to be removed.
2. Find the IBMTSDS\_SCSIProtocolController instance that matches the selected host.
3. Use InvokeMethod on IBMTSDS\_ControllerConfigurationService.HidePaths, with LUNames set to the volume names and ProtocolControllers set to the SCSIProtocolController instance name.

#### **Removing initiator ports from an SPC (example):**

1. Obtain the SPC reference and IDs of the initiator port.
2. Find the IBMTSDS\_SCSIProtocolController instance.
3. Use InvokeMethod on IBMTSDS\_ControllerConfigurationService.HidePaths, with InitiatorPortIDs set to the IDs of the initiator ports and ProtocolControllers set to the SCSIProtocolController instance name.

## **Indication profile**

---

The Storage Management Initiative Specification (SMI-S) supports two types of indications: Lifecycle indications and alert indications.

Lifecycle indications are used to convey changes in the model and are concerned only with the creation, modification, or deletion of Common Information Model (CIM) instances. Alert indications are used to draw the attention of subscribing client applications to the occurrence of an event. Lifecycle indications are implemented in all IBM FlashSystem A9000 and A9000R CIM agent releases.

### **Supported lifecycle indications by class**

Table 27 on page 53 identifies the class of objects that can be monitored by using the specified indication type for IBM FlashSystem A9000 and A9000R arrays.

Table 27: Indication types and object classes	
Class name	Supported lifecycle indications
IBMTSDS_SEVolume	IBMTSDS_InstCreation IBMTSDS_InstDeletion IBMTSDS_InstModification
IBMTSDS_VirtualPool	IBMTSDS_InstCreation IBMTSDS_InstDeletion IBMTSDS_InstModification
IBMTSDS_ProtocolControllerForSEUnit	IBMTSDS_InstCreation IBMTSDS_InstDeletion
IBMTSDS_SCSIProtocolController	IBMTSDS_InstCreation IBMTSDS_InstDeletion
IBMTSDS_SystemSpecificCollection	IBMTSDS_InstCreation IBMTSDS_InstDeletion
IBMTSDS_StorageHardwareID	IBMTSDS_InstCreation IBMTSDS_InstDeletion
IBMTSDS_iSCSIProtocolEndpoint	IBMTSDS_InstCreation IBMTSDS_InstDeletion

The lifecycle indications are sent following an IBM FlashSystem A9000 or IBM FlashSystem A9000R CIM agent configuration function being successfully completed. If the SMI-S client starts the CIM agent ExposePaths method to map a volume to a host, a related InstCreation indication with the new IBMTSDS\_ProtocolControllerForSEUnit instance is sent from the provider.

In addition, the CIM agent periodically fetches events and sends related lifecycle indications. If a StorageVolume instance is created on an array through the IBM FlashSystem A9000 or IBM FlashSystem A9000R GUI, an SMI-S client receives a related InstCreation indication with the new IBMTSDS\_SEVolume instance from the provider. By default, the IBM FlashSystem A9000 and A9000R event fetch interval is 30 seconds.

### Alert indications

The IBM FlashSystem A9000 and A9000R CIM agent periodically fetches non-internal events and sends one alert indication for each event. The alert indication comes in the form of class IBMTSDS\_AlertIndication. By default, the IBM FlashSystem A9000 and A9000R events fetching interval is 30 seconds.

---

**Note:** The IBM FlashSystem A9000 and A9000R CIM agent has a limitation of values of properties:

- IndicationTime of IBMTSDS\_InstCreation class
- IndicationTime of IBMTSDS\_InstDeletion class
- IndicationTime of IBMTSDS\_InstModification class
- EventTime of IBMTSDS\_AlertIndication class
- IndicationTime of IBMTSDS\_AlertIndication class

The above properties are representing a timestamp in format `yyyymmddhhmmss.mmmmmmsutc`, where:

- `yyyy` is the four-digit year
- `mm` is the month within the year (starting with 01)
- `dd` is the day within the month (starting with 01)
- `hh` is the hour within the day (24-hour clock, starting with 00)
- `mm` is the minute within the hour (starting with 00)
- `ss` is the second within the minute (starting with 00)
- `mmmmmm` is the microsecond within the second (starting with 000000)
- `s` is a "+" or "-", indicating that the value is a timestamp, and indicating the sign of the UTC (Universal Coordinated Time) correction field. A "+" is used for time zones east of Greenwich, and a "-" is used for time zones west of Greenwich.
- `utc` is the offset from UTC in minutes (using the sign indicated by `s`)

Fields other than `utc` are correct which are representing the UTC timestamp and the `utc` field as 000. Occasionally the `utc` property has values other than 000 (such as 540). Values other than 000 are be ignored.

---

## Replication Services profile

---

The Replication Services profile allows a storage system to copy data from a source element to a target element. The copy operations can be initiated on elements from the same storage system or across a connection to a different storage system.

Elements that are used in a copy operation can be grouped to facilitate the copy operation on many elements at the same time. Furthermore, the elements of a group can be declared as consistent.

Two types of synchronization views are supported. A target element can be synchronized to the current view of the source element, or it can be synchronized to a point-in-time view. Snapshots and clones represent a point-in-time view, while a mirror represents a current view.

Two copy modes are supported: synchronous and asynchronous. In synchronous mode, the writer waits for acknowledgment that a write to the source element was processed by the target element. Until it receives this acknowledgment, it does not accept another I/O from the host.

In asynchronous mode, the writer does not wait for this acknowledgment and continues processing I/Os, enabling writes to be sent to the target element later.

The Replication Services profile supports local and remote replication. Local replication specifies that both the source elements and the target elements are contained in a single managed system, such as an array platform.

Remote replication specifies that the source elements and the target elements are contained in separate systems.

For remote replication, the client may interact with both the source system and the target system, but the client only invokes the replication methods to a single replication service.

# Replication Services object model

Data can be copied from a source element or group of elements to a target element. The data can be copied within the same storage system, or across a connection to a different storage system.

See [Figure 9 on page 55](#) and [Figure 10 on page 56](#).

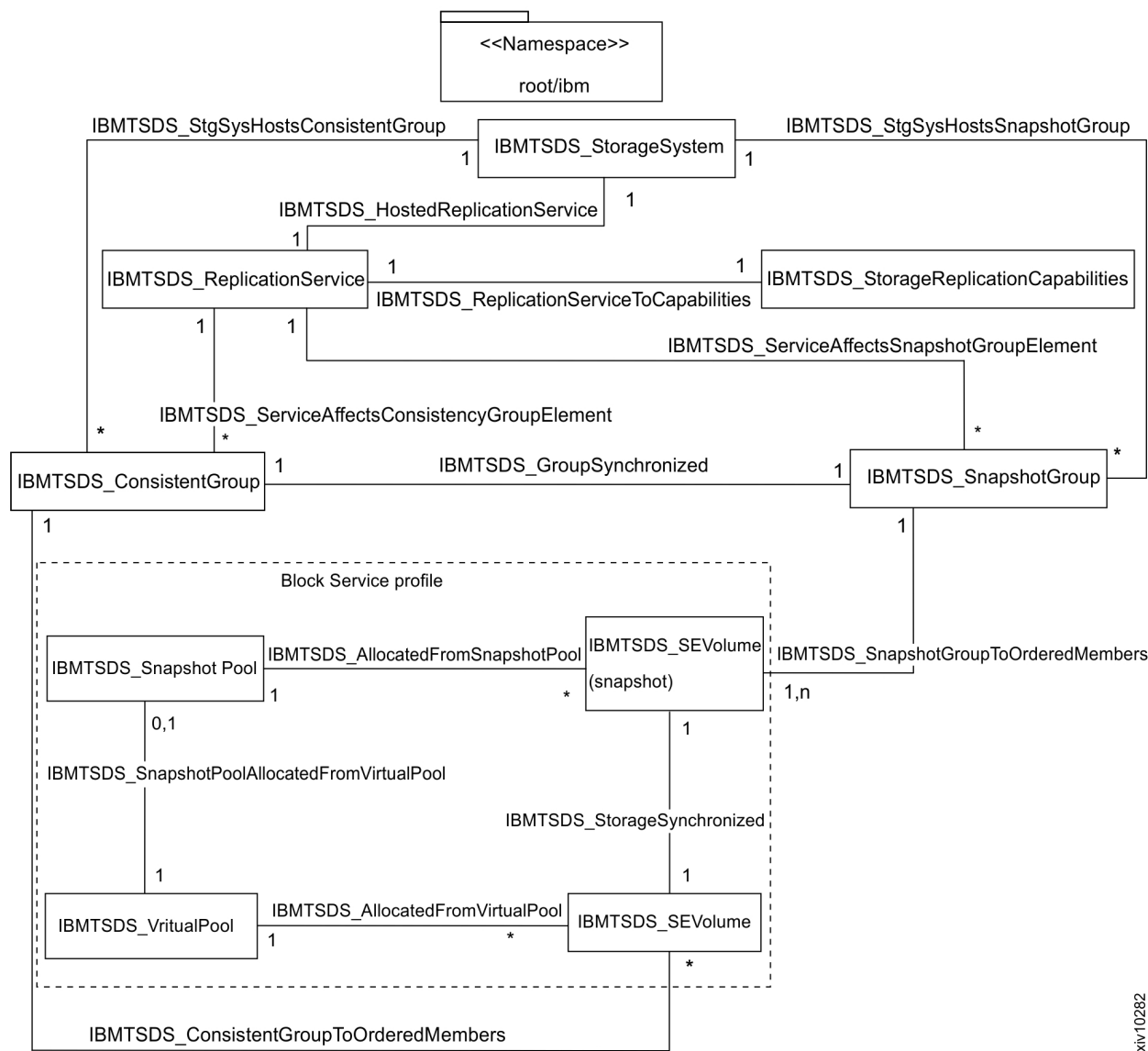


Figure 9: Replication Services (Local) SMI-S model for IBM FlashSystem A9000 and A9000R systems

xiv10282

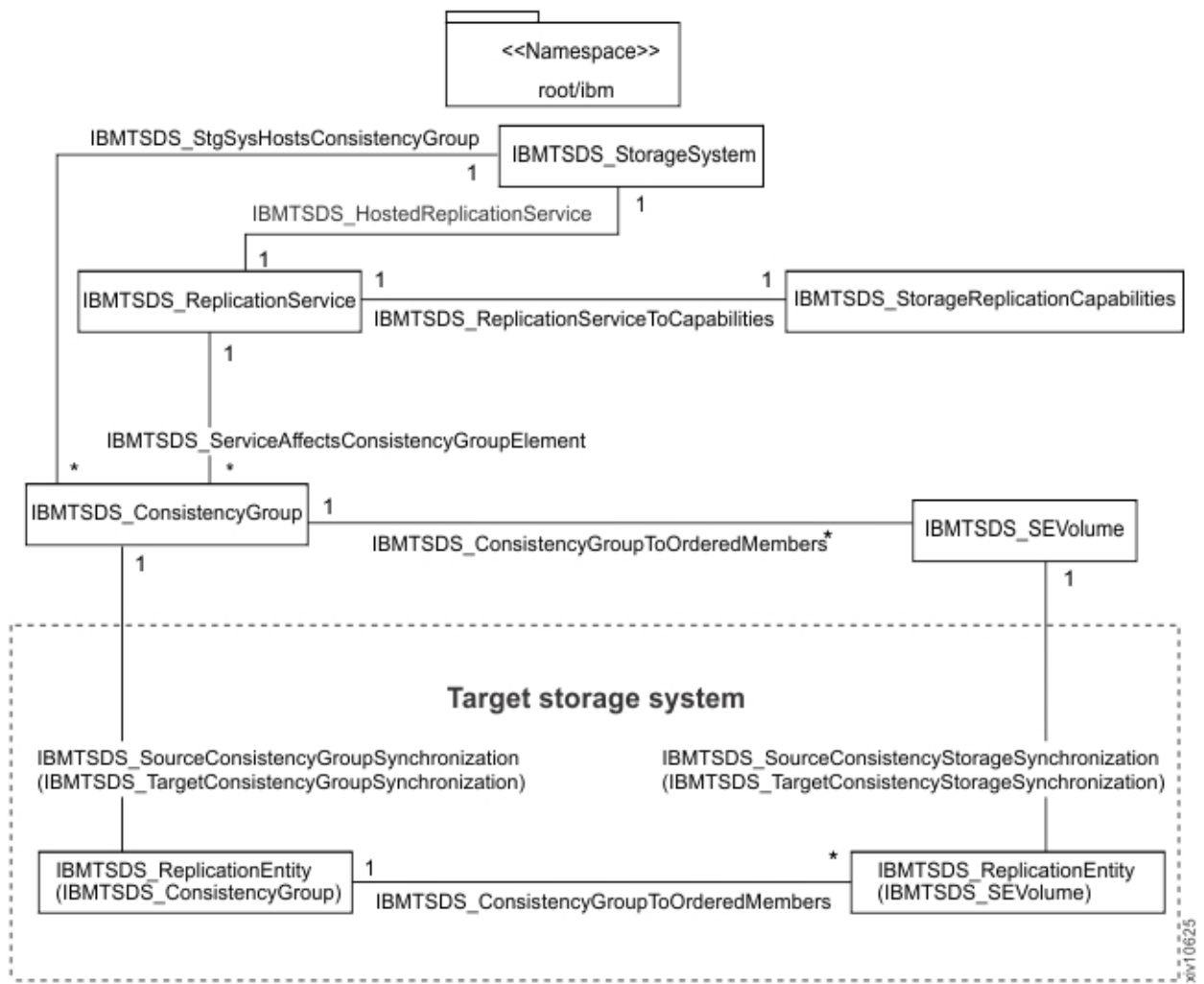


Figure 10: Replication Services (Remote) SMI-S model for IBM FlashSystem A9000 and A9000R systems

### Replication Service capabilities

Use IBM FlashSystem A9000 and A9000R terminology to map to Storage Management Initiative Specification (SMI-S) specific terminology:

Table 28: Mapping IBM FlashSystem A9000 and A9000R terminology to SMI terminology			
IBM FlashSystem A9000 and A9000R Terminology	SMI Terminology	Description	IBM FlashSystem A9000 and A9000R CIM Agent support
Snapshot	Asynchronous Snapshot Local	A "Point-in-Time", associated virtual copy of the source element. The target element enables visibility into a session.	version 12.0 or later
Volume copy	Synchronous Clone Local	A full, "Point-In-Time", unassociated local copy of the source element.	version 12.0 or later

Table 28: Mapping IBM FlashSystem A9000 and A9000R terminology to SMI terminology (continued)

IBM FlashSystem A9000 and A9000R Terminology	SMI Terminology	Description	IBM FlashSystem A9000 and A9000R CIM Agent support
Synchronous Remote Mirroring	Synchronous Mirror Remote	A synchronized remote copy of the source element.	version 12.0.1 or later
Asynchronous Remote Mirroring	Asynchronous Mirror Remote	An asynchronous remote copy of the source element.	version 12.0.2 or later

The single instance of the class ReplicationServiceCapabilities and its methods describe the various capabilities of the service. You can examine the ReplicationServiceCapabilities instance and invoke its methods to determine the specific capabilities of a replication service implementation.

Table 29: Replication Service methods

Method	Description
ConvertSyncTypeToReplicationType	Translates CopyType, Mode, and Local/Remote to the corresponding ReplicationType
ConvertReplicationTypeToSyncType	Translates ReplicationType to the corresponding CopyType, Mode, and Local/Remote
GetSupportedFeatures	Determines supported features
GetSupportedGroupFeatures	Determines supported group features
GetSupportedCopyStates	Determines the supported copy states
GetSupportedGroupCopyStates	Determines supported group copy states
GetSupportedWaitForCopyStates	Determines supported wait for copy states
GetSupportedConsistency	Determines supported consistency
GetSupportedOperations	Determines supported operations
GetSupportedGroupOperations	Determines supported group operations
GetSupportedListOperations	Determines supported list operations
GetSupportedMaximum	Determines supported maximum
GetDefaultConsistency	Determines default consistency
GetDefaultGroupPersistency	Determines default group persistency
GetSynchronizationSupported	For the supplied element, this method returns the supported synchronization operations in a series of parallel output arrays.
GetSupportedOperationsForSynchronization	For the supplied synchronized association, this method returns the supported operations in a series of parallel output arrays.

## Replication Services methods

The following sections describe functional methods associated with the Replication Services profile and object model, including uses of and parameters used by each method.

- [“Group management” on page 58](#)
- [“Creating a consistency group” on page 59](#)

- [“Deleting a consistency group” on page 59](#)
- [“Adding volumes to a consistency group” on page 59](#)
- [“Removing volumes from a consistency group” on page 59](#)
- [“Replication management” on page 60](#)
- [“Creating a snapshot or clone” on page 62](#)
- [“Restoring a snapshot to a volume or manipulating a mirror consistency group” on page 63](#)
- [“Modifying a list of synchronizations using a batch operation” on page 64](#)
- [“Retrieving target elements” on page 65](#)
- [“Retrieving snapshot or mirror relationships” on page 65](#)
- [“Retrieving strings of references to snapshots, snapshot groups, or mirror relationships” on page 66](#)
- [“Retrieving peer systems” on page 66](#)
- [“Creating new storage objects that are replicas of specified source storage objects” on page 66](#)

### Group management

The CIM agent supports group manipulation of consistency groups and snapshot groups.

See [Table 30 on page 58](#), [Figure 11 on page 58](#), and [Table 31 on page 58](#).

Table 30: Group management classes		
IBM FlashSystem A9000 and A9000R	Storage Management Initiative Specification (SMI-S) class	IBM FlashSystem A9000 and A9000R CIM class
Consistency group	CIM_ReplicationGroup	IBMTSDS_ConsistencyGroup
Snapshot group	CIM_ReplicationGroup	IBMTSDS_SnapshotGroup
Association between Consistency group and volumes in it	CIM_OrderedMemberOfCollection	IBMTSDS_ConsistencyGroupToOrderedMembers
Association between Snapshot group and snapshots in it	CIM_OrderedMemberOfCollection	IBMTSDS_SnapshotGroupToOrderedMembers

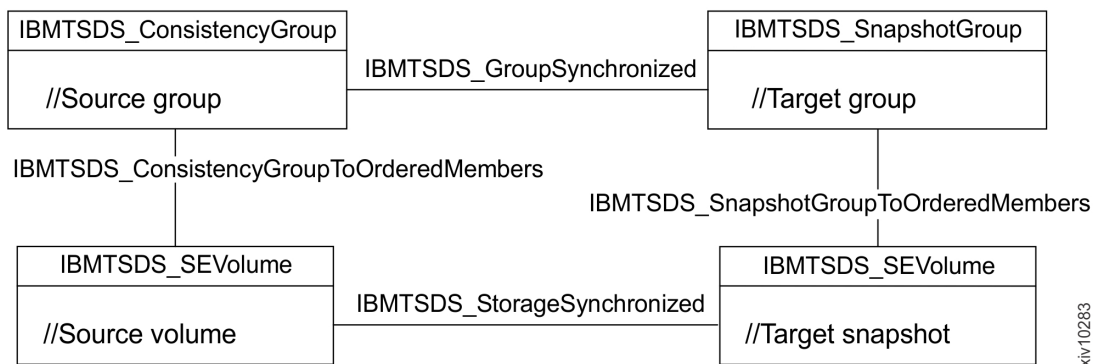


Figure 11: Sample local group information retrieval

Table 31: Extrinsic methods for group management	
Method	Description
CreateGroup	Creates a consistency group



Table 31: Extrinsic methods for group management (continued)

Method	Description
DeleteGroup	Deletes an existing consistency group
AddMembers	Adds volumes to a consistency group
RemoveMembers	Removes volumes from a consistency group

### Creating a consistency group

The IBMTSDS\_ReplicationService. CreateGroup method creates a consistency group.

#### Parameters

The following list describes the parameters of the IBMTSDS\_ReplicationService. CreateGroup method.

#### GroupName

As an input, refers to the name of the group that is created. If not specified, a random name is generated.

#### Members

As an input, refers the members to be added to the group that is created. If not specified, an empty group is created.

#### ReplicationGroup

As an output, refers to the group that is created.

### Deleting a consistency group

The IBMTSDS\_ReplicationService. DeleteGroup method deletes an existing consistency group.

#### Parameters

The following list describes the parameters of the IBMTSDS\_ReplicationService. DeleteGroup method.

#### RemoveElements

As an input, a value of true indicates that members in this group are removed before you delete the group. If one or more elements in the group are in a replication relationship, its value is ignored.

#### ReplicationGroup

As an input, refers to the consistency group that you want to delete.

### Adding volumes to a consistency group

The IBMTSDS\_ReplicationService. AddMembers method adds volumes to a consistency group.

#### Parameters

The following list describes the parameters of the IBMTSDS\_ReplicationService. AddMembers method.

#### ReplicationGroup

As an input, refers to the consistency group to which members are added.

#### Members

As an input, refers to volumes to be added to the group. All specified volumes must be in the same pool of the consistency group.

### Removing volumes from a consistency group

The IBMTSDS\_ReplicationService. RemoveMembers method removes volumes from a consistency group.

#### Parameters

The following list describes the parameters of the IBMTSDS\_ReplicationService. RemoveMembers method.

**ReplicationGroup**

As an input, refers to the consistency group from which members are removed.

**Members**

As an input, refers to volumes to be removed from the group.

**DeleteOnEmptyElement**

As an input, a value of true indicates that the consistency group is deleted if all members are removed.

**Replication management**

IBM FlashSystem A9000 and A9000R support local replication manipulation, including local snapshot and clone, and also supports remote replication manipulation, including volume and consistency group mirroring.

See the following tables and figures for information about replication management:

<i>Table 32: Replication management classes</i>		
<b>IBM FlashSystem A9000 and A9000R</b>	<b>SMI class</b>	<b>IBM FlashSystem A9000 and A9000R CIM class</b>
Volume, snapshot	CIM_StorageVolume	IBMTSDS_SEVolume
Remote volume / consistency group	CIM_ReplicationEntity	IBMTSDS_ReplicationEntity
Association between volume and its snapshot	CIM_StorageSynchronized	IBMTSDS_StorageSynchronized
Association between consistency group and its snapshot group	CIM_GroupSynchronized	IBMTSDS_GroupSynchronized
Association between mirrored volumes	CIM_StorageSynchronized	IBMTSDS_SourceConsistencyStorageSynchronized/ IBMTSDS_TargetConsistencyStorageSynchronized
Association between mirrored consistency groups	CIM_GroupSynchronized	IBMTSDS_SourceConsistencyGroupSynchronized/ IBMTSDS_TargetConsistencyGroupSynchronized

See [Figure 12](#) on [page 61](#).

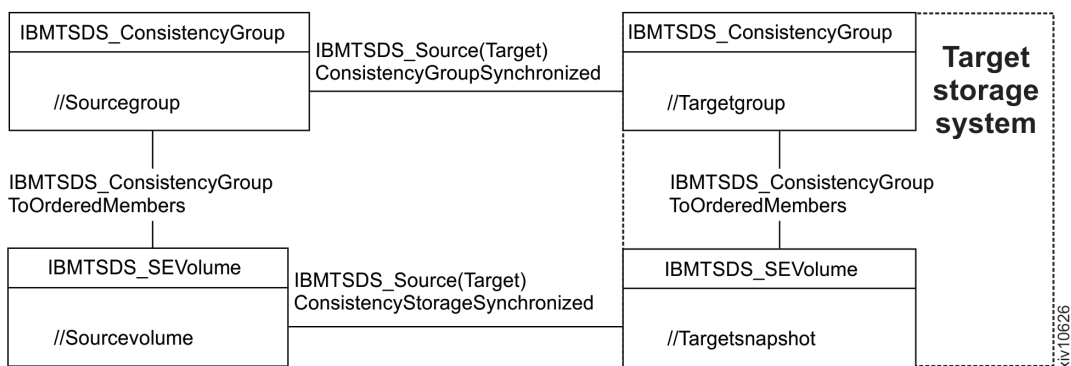


Figure 12: Sample remote group information retrieval

**Note:** The IBMTSDS\_SourceConsistencyStorageSynchronized and IBMTSDS\_TargetConsistencyStorageSynchronized classes are association classes which associate the source volume with the target volume.

They are essentially the same association class, but differ in how they get data from the Primary IBM FlashSystem A9000 or IBM FlashSystem A9000R and the Secondary IBM FlashSystem A9000 or IBM FlashSystem A9000R respectively; see Figure 13 on page 61.

At the same time, the IBMTSDS\_SourceConsistencyGroupSynchronized and IBMTSDS\_TargetConsistencyGroupSynchronized classes associate the source consistency group with the target consistency group.

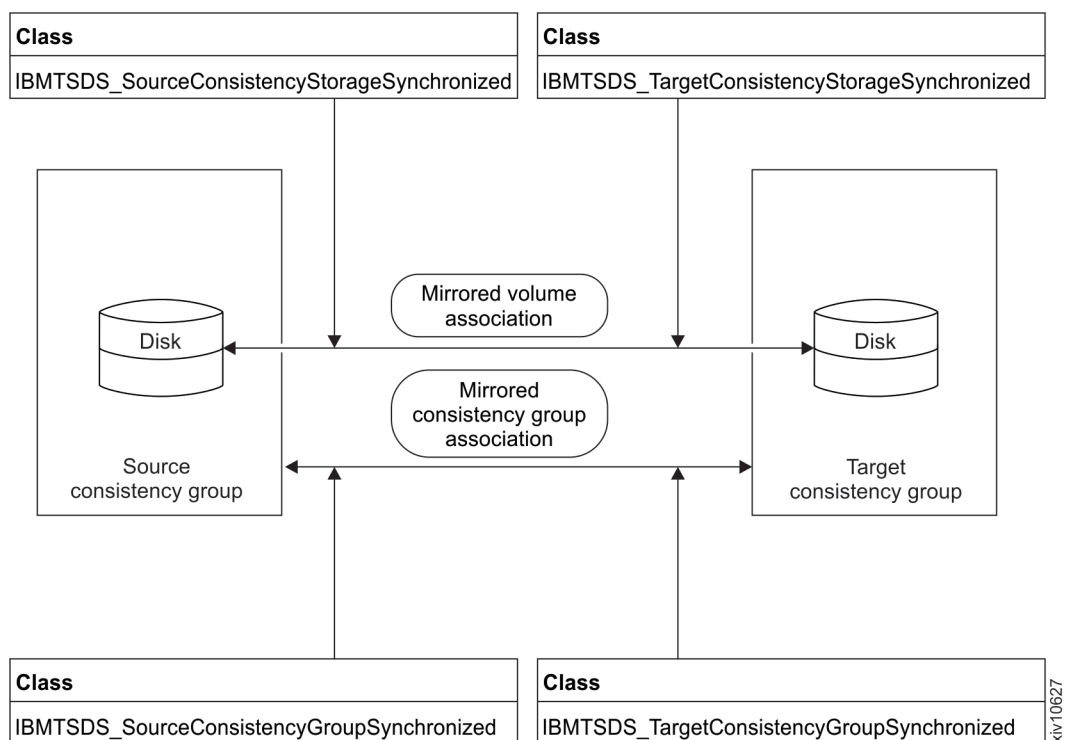


Figure 13: Association classes for mirrored volumes and consistency groups

<i>Table 33: Extrinsic methods for replication management</i>	
<b>Method</b>	<b>Description</b>
CreateElementReplica	Create a snapshot of a volume, or a duplication snapshot of a snapshot, or a clone of a volume.
CreateGroupReplica	Create a snapshot group of a consistency group.
ModifyReplicaSynchronization	For local replication service, restore a snapshot or a snapshot group; or delete a snapshot or a snapshot group.
ModifyListSynchronization	Restore a list of snapshots or snapshot groups; or delete snapshots or snapshot groups.
GetAvailableTargetElements	Return snapshots of a specified volume or snapshot.
GetPeerSystems	Get all of the peer systems.
CreateGroupReplicaFromElements	Create consistency group and volumes mirror.
GetReplicationRelationships	Get all of available synchronization relationships.
GetReplicationRelationshipInstances	Get all of available synchronization relationship instances as strings of references.

### **Creating a snapshot or clone**

The IBMTSDS\_ReplicationService. CreateElementReplica method creates a snapshot of a volume, duplicate snapshot of a snapshot, clone of a volume, and clone of a snapshot.

### **Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. CreateElementReplica method.

#### **ElementName**

As an input, refers the name for the volume or clone being created. If not specified, a random name is generated.

#### **SyncType**

As an input, refers the type of replica to be created. To create a snapshot, specify 7; to create a clone, specify 8.

#### **Mode**

As an input, refers the mode of replica to be created. Only 3 is supported, which means an asynchronous replica is created if specified.

#### **SourceElement**

As an input, refers the source volume or snapshot for which the replica is created.

#### **TargetElement**

As an input, refers to a target element to use if specified. This parameter cannot be specified when you are creating a snapshot. As an output, refers to the replica that is created.

#### **TargetPool**

As an input, refers the pool in which the replica is being created. For snapshot creation, if specified, it must be an instance of IBMTSDS\_SnapshotPool which is in the same pool of the source element. For clone creation, if specified, it can be an instance of IBMTSDS\_VirtualPool representing any pool in the device. If not specified, the Common Information Model (CIM) selects the first available pool on the device to create the clone.

**WaitForCopyState**

As an input, refers to the copy state the replica must reach before the method returns. Only 4 is supported if specified.

**Creating a snapshot group for a consistency group**

The IBMTSDS\_ReplicationService. CreateGroupReplica method creates a snapshot group for a consistency group.

**Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. CreateGroupReplica method.

**RelationshipName**

As an input, refers to the name of the snapshot group that is being created. If not specified, a random name is generated.

**SyncType**

As an input, refers the type of replica to be created. Only 7 is supported if specified because this method can create a snapshot of a consistency group.

**Mode**

As an input, refers the mode of replica to be created. Only 3 is supported if specified, which means an asynchronous snapshot group is created.

**SourceGroup**

As an input, refers to a consistency group for which the snapshot group is being created.

**TargetGroup**

As an output, refers to the snapshot group that is being created.

**Consistency**

As an input, refers to the group consistency. Only 3 is supported if specified.

**Synchronization**

As an output, refers to the created association between the source consistency group and the snapshot group that is created.

**TargetPool**

As an input, refers to the pool in which the replica is being created. The pool must be the same as the pool of the SourceGroup if specified.

**WaitForCopyState**

As an input, refers to the copy state the replica must reach before the method returns. Only 4 is supported if specified.

**Restoring a snapshot to a volume or manipulating a mirror consistency group**

For local replication service, use the IBMTSDS\_ReplicationService. ModifyReplicaSynchronization method to restore a snapshot to a volume, restore a snapshot group to a consistency group, delete a snapshot, and delete a snapshot group. For remote replication service, use the IBMTSDS\_ReplicationService. ModifyReplicaSynchronization method to manipulate the mirror consistency group (for example, to split the mirrored consistency group or set up failover for the mirrored consistency group).

**Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. ModifyReplicaSynchronization method.

**Operation**

As an input, refers to the type of modification to be made to the replica. The following values/modifications are supported: 4 (Activate), 5 (AddSyncPair), 7 (Deactivate), 8 (Detach), 10 (Failover), 12 (Fracture), 13 (RemoveSyncPair), 14 (ReSync Replica), 15 (Restore from Replica), 19 (Return To ResourcePool) and 20 (Reverse Roles), and 21 (Split).

## Synchronization

As an input, refers to the replica synchronization to be modified. For local, it must be an instance of IBMTSDS\_GroupSynchronized or IBMTSDS\_StorageSynchronized class. For remote, it must be an instance of IBMTSDS\_SourceConsistencyGroupSynchronized or IBMTSDS\_SourceConsistencyStorageSynchronized.

## WaitForCopyState

As an input, refers to the copy state the replica must reach before the method returns. The values 3, 4, 8, and 10 are supported, if specified. Different operations correspond with different WaitForCopyState values, as shown in [Table 34 on page 64](#).

Table 34: Operations, operation descriptions, and corresponding WaitForCopyState states		
Operation	Operation Description (from profile)	Corresponding WaitForCopyState Value
Activate	Activate an Inactive or Prepared StorageSynchronized association.	Unsynchronized
Detach	Remove the association between the source and target elements. Does not delete the target element.	N/A
Failover	Enable the read and write operations from the host to the target element. This operation is useful for situations when the source element is unavailable.	Failedover
Fracture/Split	Separate the target element from the source element.	Inactive
Resync Replica	Resynchronize a fractured target element. Or, from a Broken or Aborted relationship. To continue from the Broken state, the problem should be corrected first before resyncing the replica. Also, to continue from the Aborted state.	Synchronized
Reverse Roles	Switch the source and the target element roles. The source element may need to be Read Only. See GetSupportedFeatures in capabilities.	Unsynchronized
Add/Remove SyncPair	Add/Remove mirrored volumes into/from mirrored consistency group.	Synchronized

**Note:** The ReverseRoles and ResyncReplica operations may take some time to execute, so a Job will be returned (4096).

## Modifying a list of synchronizations using a batch operation

The IBMTSDS\_ReplicationService.ModifyListSynchronization method is a batch operation of the IBMTSDS\_ReplicationService.ModifyReplicaSynchronization method, which modifies a list of synchronizations.

For a list of applicable parameters, see [“Restoring a snapshot to a volume or manipulating a mirror consistency group” on page 63](#).

**Retrieving target elements**

The IBMTSDS\_ReplicationService. GetAvailableTargetElements method retrieves all of the candidate target elements for the supplied source element.

**Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. GetAvailableTargetElements method.

**SourceElement**

As an input, refers to the source volume. It must be an instance of IBMTSDS\_SEVolume class.

**SyncType**

As an input, refers to type of target elements. Only 7 is supported if specified.

**Mode**

As an input, refers to the mode of target elements. Only 3 is supported if specified.

**Candidates**

As an output, refers to the target elements of the source volume.

**Retrieving snapshot or mirror relationships**

The IBMTSDS\_ReplicationService. GetReplicationRelationships method retrieves all of the snapshot or mirror relationships on an IBM FlashSystem A9000 or IBM FlashSystem A9000R device.

**Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. GetReplicationRelationships method.

**Type**

As an input, refers to the replication type. To get snapshots and mirrored volumes, specify 2; to get snapshot groups, specify 3; if not specified, all snapshots and snapshot groups are returned.

**SyncType**

As an input, refers to type of target elements. To get snapshots and snapshot groups, specify 7. To get mirror relationships, specify 7; if no value is specified, all snapshots, snapshot groups, and mirror relationships are returned.

**Mode**

As an input, refers to mode of target elements. 2 and 3 are supported if specified.

**Locality**

As an input, refers to locality of target elements. To get mirrored volumes and consistency groups, specify 3. To get snapshots and snapshot groups, specify 2. If no value is specified, all snapshots, snapshot groups, and mirror relationships are returned.

**CopyState**

As an input, refers to copy state of target elements. Only 4 is supported if specified.

**Synchronizations**

As an input, refers to references of the returned snapshots, snapshot groups, or mirror relationships.

**Retrieving strings of references to snapshots, snapshot groups, or mirror relationships**

Use the IBMTSDS\_ReplicationService. GetReplicationRelationshipInstances to retrieve strings of references to snapshots, snapshot groups, or mirror relationships.

**Retrieving peer systems**

Use the IBMTSDS\_ReplicationService. GetPeerSystems method to retrieve (or start a job to retrieve) all of the peer systems. A peer system is a system that is known and visible to the replication service. This method will return all the connected IBM FlashSystem A9000 or IBM FlashSystem A9000R systems.

**Creating new storage objects that are replicas of specified source storage objects**

Use the IBMTSDS\_ReplicationService. CreateGroupReplicaFromElements method to create (or start a job to create) new storage objects that are replicas of the specified source storage objects (SourceElements).

If 0 is returned, the function completed successfully, and no ConcreteJob instance is created. If 4096/0x1000 is returned, a ConcreteJob is started, and a reference to it is returned in the Job output parameter. This method combines the functionality of the CreateGroup and CreateGroupReplica methods, in one call. This method creates mirrors for volumes and consistency groups and then adds the mirrored volumes into the mirrored consistency groups.

**Parameters**

The following list describes the parameters of the IBMTSDS\_ReplicationService. CreateGroupReplicaFromElements method.

**SyncType**

As an input, refers to the type of target elements. Only 6 is supported.

**Mode**

As an input, refers to the mode of target elements. 2 and 3 are supported, if specified.

**SourceGroup**

As an input, refers to a source group. It must be an instance of IBMTSDS\_ConsistencyGroup class.

**SourceElements**

As an input, refers to the source volumes. It must be an instance of IBMTSDS\_SEVolume class.

**SourceGroupName**

As an input, refers to the name of the source group to be created. As an output, refers to the name of the source group that is being created.

**TargetGroup**

As an input, refers to a target group. It must be an instance of IBMTSDS\_ConsistencyGroup class. If not specified, a target group will be created in the target pool, which is specified by the TargetPool parameter. TargetGroup and TargetPool can't be NULL at the same time.

**TargetGroupName**

As an input, refers to the name of the target group to be created.

**Consistency**

As an input, only 3 is supported, if specified.

**TargetPool**

As an input, refers to the target pool that puts the target elements (the replicas). It must be an instance of IBMTSDS\_VirtualPool. If not specified, it will be gotten from the TargetGroup parameter. If TargetGroup and TargetPool are specified at the same time, and TargetGroup is not in the TargetPool parameter, TargetGroup will be moved to it.

**WaitForCopyState**

As an input, refers to the copy state the replica must reach before the method returns. Only 3 is supported, if specified.

**Job**

As an output, refers to the job.



## Replication Services indications

The IBM FlashSystem A9000 and A9000R Common Information Model (CIM) agent supports the following Replication Services profile indications.

Table 35: Replication Services profile indications	
Indication	Description
IBMTSDS_StorageSynchronized InstCreation	After a snapshot is created or snapshot group is created
IBMTSDS_GroupSynchronized InstCreation	After a snapshot group is created
IBMTSDS_GroupSynchronized InstDeletion	After a snapshot group is deleted
IBMTSDS_StorageSynchronized InstDeletion	After a snapshot or snapshot group is deleted
IBMTSDS_ConsistencyGroupInstDeletion	After a consistency group is deleted
IBMTSDS_ConsistencyGroupInstCreation	After a consistency group is created
IBMTSDS_ConsistencyGroupInstModification	After a consistency group is renamed
IBMTSDS_ConsistencyGroupToOrderedMembersInstCreation	After volume is added/removed into/ from consistency group
IBMTSDS_ConsistencyGroupToOrderedMembersInstDeletion	After all volumes are removed from the consistency group
IBMTSDS_SourceConsistencyGroupSynchronizedInstDeletion	After the mirror relationship is deleted
IBMTSDS_SourceConsistencyGroupSynchronizedInstCreation	After the mirror relationship is created
IBMTSDS_TargetConsistencyGroupSynchronizedInstCreation	After the mirror relationship is created
IBMTSDS_SourceConsistencyGroupSynchronizedInstModification	After the mirror relationship designation is changed
IBMTSDS_TargetConsistencyGroupSynchronizedInstModification	After the mirror relationship designation is changed

## Job Control profile

In some profiles, like the replication services profile, some of the methods described may take some time to execute. In these cases, a mechanism is needed to handle asynchronous execution of the method as a 'job.' The Job Control profile defines the constructs and behavior for job control.

When the Job Control profile is implemented, and a client executes a method that executes asynchronously, a reference to an instance of ConcreteJob is returned and the return value for the method is set to Method parameters checked - job started (4096). The ConcreteJob instance allows the progress of the method to be checked, and instance indications can be used to subscribe for job completion.

Although there are many methods, such as format volume and copy volume that are asynchronous jobs, in this work item, we only implement the job control for the volume/consistency group mirror.

In the IBM FlashSystem A9000 and A9000R Common Information Model (CIM) provider, the methods CreateGroupReplicaFromElements and ModifyReplicaSynchronization will output a job that will be associated to the elements whose references are created or modified as a side-effect of the job's execution via the AffectJobElement association.

The lifetime of a completed job instance, and thus the AffectedJobElement association to the appropriate Element instance, is currently implementation dependent. But, the set of AffectedJobElement associations to the Input and Output elements present when the job finishes execution will remain until the job is deleted.

## Job Control object model

The Job Control profile provides a mechanism to handle asynchronous execution of methods that take some time to execute.

For a diagram of how this mechanism works, see [Figure 14 on page 68](#).

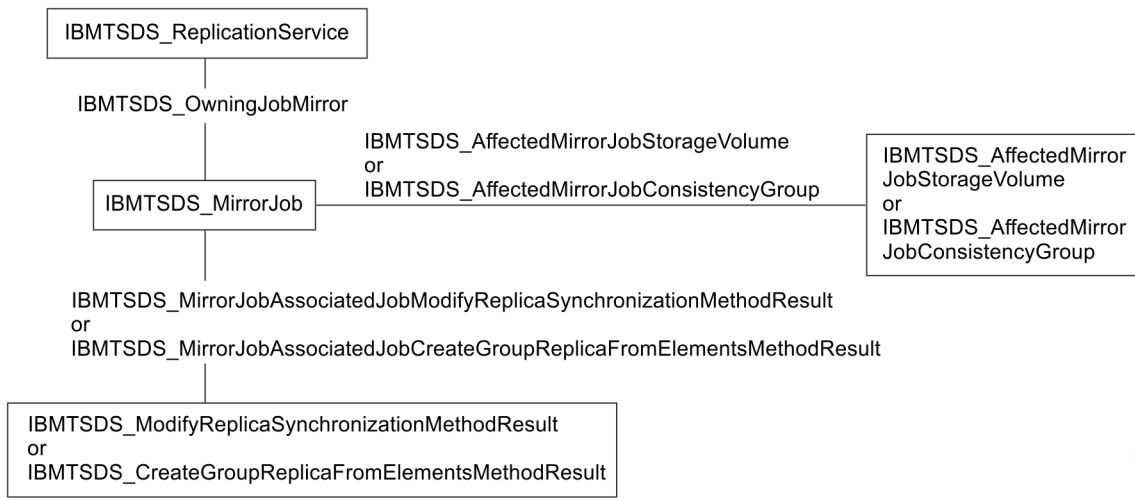


Figure 14: Job Control SMI-S model for IBM FlashSystem A9000 and A9000R systems

For IBMTSDS\_MirrorJob, if all the operations completed successfully, the JobState value will be 7 (Completed), and the OperationStatus value will be 2 (Ok), 17 (Completed). If not, the JobState value will be 10 (Exception), and the OperationStatus value will be 6 (Error), 17 (Completed). If the job finished, and the duration time was over 60 minutes, the job will be deleted automatically.

## Thin Provisioning profile

Thin provisioning is a capability of some Block Services implementations. It defers provisioning of backing store for regions of a volume until the regions are accessed (written) by the consumer (for example, host file system).

The alternatives (fully provisioned volumes) allocate all the requested capacity from the backing store at the time the volume is created. For thin provisioned volumes, the block server implementation tracks information about which regions are accessed. After a region is accessed, the backing storage is allocated.

The Thin Provisioning profile allows Storage Management Initiative Specification (SMI-S) clients to determine whether a storage system (and children such as pools and volumes) supports thin provisioning. Clients can also determine the difference between the exposed "virtual capacity" and actual committed physical storage and create thinly provisioned volumes and pools.

For IBM FlashSystem A9000 and A9000R, pools are provisioned with virtual capacity. It is not possible to provision a pool with a set amount of physical capacity.

From a volume perspective, you define the volume soft size, and it takes the size from the virtual size of the pool. Upon creation of a volume in a pool, no preallocation of hard capacity is done. Only soft capacity is taken from the virtual capacity of the pool.

## Thin Provisioning object model

Compared with the Block Services profile, the Thin Provisioning profile has more classes and some existing classes with modified properties.

See [Figure 15 on page 69](#) for the Thin Provisioning Storage Management Initiative Specification (SMI-S) model for IBM FlashSystem A9000 and A9000R systems.

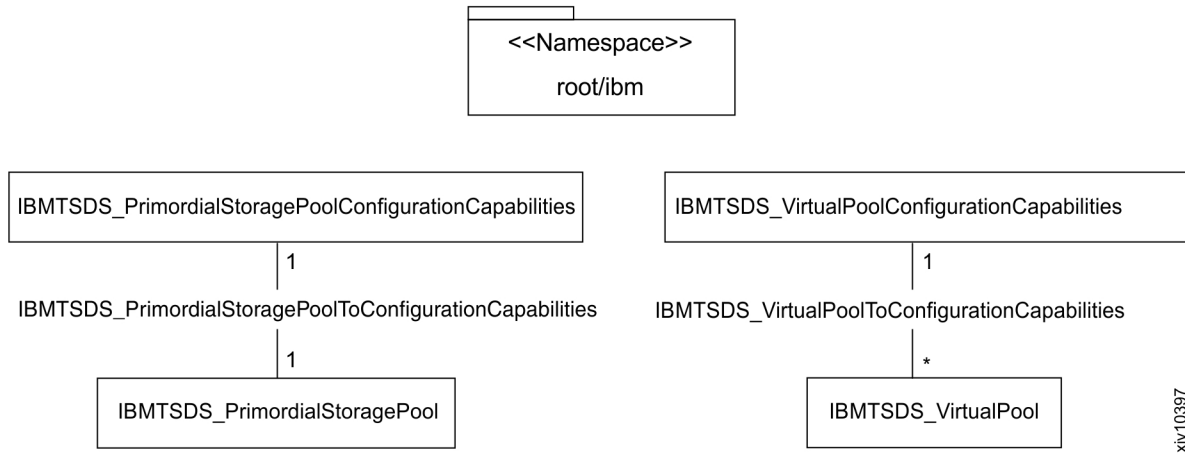


Figure 15: Thin Provisioning SMI-S model for IBM FlashSystem A9000 and A9000R systems

As with the Block Services profile, the CIM agent has four CIM\_StorageSetting instances, but IBMTSDS\_VirtualPoolSetting adds an instance type in the Thin Provisioning profile. One is the default, and the other is a newly created one.

### **IBMTSDS\_VirtualPoolSetting.InstanceID="IBMTSDS:IBM XIV Virtual Storage Pool Setting"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyStoragePool method to create or modify a regular thin provisioning VirtualPool instance.

### **IBMTSDS\_VirtualPoolSetting.InstanceID="IBMTSDS:IBM.2810-1310129-112714143137626"**

This StorageSetting instance is supplied as the Goal parameter for the CreateOrModifyStoragePool method to create or modify a thin provisioning VirtualPool instance.

For more information about the thin provisioning object model, see the BlockServices.mof file.

## Extrinsic Methods

Table 36: Replication Service methods	
Method	Description
CreateSetting	Creates an IBMTSDS_VirtualPoolSetting instance.
modify	Modifies the properties of the newly created IBMTSDS_VirtualPoolSetting instance.
delete	Deletes the newly created IBMTSDS_VirtualPoolSetting instance.
CreateOrModifyStoragePool	Creates or modifies a thin provisioning VirtualPool instance.

## Thin Provisioning methods

The following sections describe functional methods associated with the Thin Provisioning profile and object model, including uses of and parameters used by each method.

- [“Creating an IBMTSDS\\_VirtualPoolSetting instance” on page 70](#)
- [“Modifying the properties of a newly created IBMTSDS\\_VirtualPoolSetting instance” on page 70](#)

- [“Deleting a new IBMTSDS\\_VirtualPoolSetting instance” on page 70](#)
- [“Creating or modifying a thin provisioning virtual pool” on page 70](#)
- [“Creating a thin provisioning pool” on page 71](#)
- [“Thin Provisioning indications” on page 71](#)

### **Creating an IBMTSDS\_VirtualPoolSetting instance**

You can use the IBMTSDS\_PrimordialStoragePoolCapabilities. CreateSetting method to create an IBMTSDS\_VirtualPoolSetting instance. The newly created instance will expire after 600 seconds.

#### **Parameters**

The following list describes the parameters of the IBMTSDS\_PrimordialStoragePoolCapabilities. CreateSetting method.

#### **SettingType**

As an input, the value must be 2 (default), 3 (goal), or null.

#### **NewSetting**

As an output, refers to the newly created IBMTSDS\_VirtualPoolSetting instance.

### **Modifying the properties of a newly created IBMTSDS\_VirtualPoolSetting instance**

You can use the IBMTSDS\_VirtualPoolSetting. modify method to modify the properties of a newly created IBMTSDS\_VirtualPoolSetting instance.

#### **Parameters**

The following list describes the parameters of the IBMTSDS\_VirtualPoolSetting. modify method.

#### **ThinProvisionedInitialReserve**

As an input, represents the snapshot size of the newly created pool to be used as the Goal parameter for the CreateOrModifyStoragePool method.

#### **ThinProvisionedPoolType**

As an input, just 7 is supported.

#### **Cop**

As an input, represents the Common Information Model (CIM) object paths (COPs) of the newly created IBMTSDS\_VirtualPoolSetting instance to modify.

### **Deleting a new IBMTSDS\_VirtualPoolSetting instance**

You can use the IBMTSDS\_VirtualPoolSetting. delete method to delete the newly created IBMTSDS\_VirtualPoolSetting instance.

---

**Note:** The default instance of IBMTSDS\_VirtualPoolSetting cannot be deleted.

---

#### **Parameters**

The following list describes the parameters of the IBMTSDS\_VirtualPoolSetting. delete method.

#### **Cop**

As an input, represents the Common Information Model (CIM) object paths (COPs) of the newly created IBMTSDS\_VirtualPoolSetting instance to delete.

### **Creating or modifying a thin provisioning virtual pool**

You can use the IBMTSDS\_StorageConfigurationService. CreateOrModifyStoragePool method to create or modify a thin provisioning VirtualPool instance.

#### **Parameters**

The following list describes the parameters of the IBMTSDS\_StorageConfigurationService. CreateOrModifyStoragePool method.

**Pool (required for modification)**

As an input parameter, specifies whether you want to create or modify a pool. If you specify a reference to a pool, indicates that you want to modify the pool. If the parameter is left null, indicates that you want to create a pool.

**InPools**

The InPools parameter specifies which pool to create the pool from. To create a VirtualPool instance, only the object reference of IBMTSDS\_PrimordialStoragePool is allowed for this parameter since all VirtualPool instances are created in Primordial StoragePool.

To create a SnapshotPool instance, only the object reference of IBMTSDS\_VirtualPool is allowed for this parameter, if specified, since a SnapshotPool instance is created in a VirtualPool instance.

---

**Note:** The Common Information Model (CIM) schema defines this input parameter to be an array of strings that represent CIM object paths (COPs), and not actual references to objects.

---

**Goal**

The Goal parameter represents the StorageSetting instance of the pool to be created. To create a VirtualPool instance, only the object reference of IBMTSDS\_VirtualPoolSetting is allowed.

To create a SnapshotPool instance, only the object reference of IBMTSDS\_SnapshotPoolSetting is allowed. See the BlockServer.mof file for details.

**ElementName**

The ElementName property provides a means for you to set a meaningful name for the pool that is being created or modified. If specified, it is limited to 63 characters and can contain letters, digits, blank spaces, -, \_, . and ~ characters. Blank spaces cannot be the beginning and ending characters. If not specified during pool creation, a random pool name is generated in the format pool<random integer>.

---

**Note:** The name of the pool must be unique in the system. It cannot be a name that is already assigned to one of the other pools in the system.

---

**Size (required for creation)**

As an input parameter, Size specifies the requested size of the pool. Null is not allowed for pool creation. As an output parameter, Size specifies the achieved pool's size.

**Creating a thin provisioning pool**

You can use the CreateOrModifyStoragePool method to create a thin provisioning pool.

1. Call IBMTSDS\_PrimordialStoragePool.GetSupportedSizeRange with ElementType set to StoragePool1. If the return code is 0 (success), specify a size.
2. Call invokeMethod on IBMTSDS\_PrimordialStoragePoolCapabilities.CreateSetting and specify the SettingType value.
3. Call enumerateInstances on IBMTSDS\_VirtualPoolSetting, and then save the CIMObjectPath value of this newly created instance.
4. Call invokeMethod on IBMTSDS\_VirtualPoolSetting.modify and specify the IBMTSDS\_VirtualPoolSetting value obtained in step 3 to Cop and pool snapshot size from ThinProvisionedInitialReserve.
5. Call invokeMethod on IBMTSDS\_StorageConfigurationService.CreateOrModifyStoragePool and specify the IBMTSDS\_VirtualPoolSetting value obtained in step 3 to Goal and the ElementName value to Size.
6. Save the pool output parameter. It is the reference to the thin provisioning VirtualPool that was created.

**Thin Provisioning indications**

The IBM FlashSystem A9000 and A9000R Common Information Model (CIM) agent supports three Thin Provisioning profile indications.

Table 37: Thin Provisioning profile indications

CIM Indication	IBM FlashSystem A9000 and A9000R Event	Description
CAPACITY_WARNING	STORAGE_POOL_VOLUME_USAGE_TOO_HIGH	Thin provisioned volume or pool with the identifier <i>Volume or Pool ID</i> capacity in use near available limit.
CAPACITY_CRITICAL	STORAGE_POOL_EXHAUSTED	Thin provisioned volume or pool with the identifier <i>Volume or Pool ID</i> capacity in use exceeded available limit.
CAPACITY_OKAY	STORAGE_POOL_VOLUME_USAGE_BACK_TO_NORMAL	Thin provisioned volume or pool with the identifier <i>Volume or Pool ID</i> capacity condition cleared.

# Chapter 5. Conformance tests

All Storage Networking Industry Association (SNIA) official Certification Test Programs (CTP) and Microsoft System Center Virtual Machine Manager (SCVMM) Storage Automation tests have been completed.

## SNIA official CTP tests

Table 38: SNIA CTP results		
IBM FlashSystem A9000 and A9000R CIM agent release	Official CTP Test Suite	Official CTP Test Results
12.0.x	<ul style="list-style-type: none"><li>Storage Management Initiative Specification (SMI-S) version 1.6</li><li>Test version 1.6.0.1081</li></ul>	<a href="http://www.snia.org/tech_activities/standards/curr_standards/smi">http://www.snia.org/tech_activities/standards/curr_standards/smi</a>

## SCVMM Storage Automation tests

All Microsoft System Center Virtual Machine Manager (SCVMM) 2012 Storage Automation tests were completed on IBM FlashSystem A9000 and A9000R with IBM FlashSystem A9000 and A9000R Common Information Model (CIM) agent version 12.0.





## Notices

---

These legal notices pertain to the information in this IBM Storage product documentation.

This information was developed for products and services offered in the US. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
USA*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119*

Armonk, NY 10504-1785  
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

## Trademarks

---

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

# Glossary

---

This glossary includes terms and definitions for IBM FlashSystem A9000 and A9000R.

This glossary includes selected terms and definitions from:

- The *American National Standard Dictionary* for Information Systems, ANSI X3.172–1990, copyright 1990 by the American National Standards Institute (ANSI), 11 West 42nd Street, New York, New York 10036. Definitions derived from this book have the symbol (A) after the definition.
- IBM Terminology, which is available online at the [IBM Terminology website](http://www.ibm.com/software/globalization/terminology/index.jsp) ([www.ibm.com/software/globalization/terminology/index.jsp](http://www.ibm.com/software/globalization/terminology/index.jsp)). Definitions derived from this source have the symbol (GC) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions derived from this book have the symbol (I) after the definition. Definitions taken from draft international standards, committee drafts, and working papers that the ISO/IEC JTC1/SC1 is developing have the symbol (T) after the definition, indicating that final agreement has not been reached among the participating National Bodies of SC1.

This glossary uses the following cross-reference forms:

## See

Refers the reader to one of two kinds of related information:

- A term that is the expanded form of an abbreviation or acronym. This expanded form of the term contains the full definition.
- A synonym or more preferred term

## See also

Refers the reader to one or more related terms.

## Contrast with

Refers the reader to a term that has an opposite or substantively different meaning.

## A

### access

To obtain computing services or data.

In computer security, a specific type of interaction between a subject and an object that results in flow of information from one to the other.

### Active Directory

Microsoft Active Directory (AD) provides directory (lookup), DNS and authentication services.

### alerting event

An event that triggers recurring event notifications until it is cleared.

### allocated storage

The space that is allocated to volumes but not yet assigned. Contrast with *assigned storage*.

### API

See *application programming interface (API)*.

### application programming interface (API)

An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

### assigned storage

The space that is allocated to a volume and that is assigned to a port.

### Asynchronous interval

Denotes, per given coupling, how often the master runs a new sync job.

**authorization level**

The authorization level determines the permitted access level to the various functions of IBM Hyper-Scale Manager:

**Read only**

Only viewing is allowed.

**Full**

Access to all the configuration and control functions is allowed, including shutdown of the system. This level requires a password.

**auto-delete priority**

As the storage capacity reaches its limits, snapshots are automatically deleted to make more space. The deletion takes place according to the value set for each snapshot, as follows:

**1**

last to be deleted

**4**

first to be deleted

Each snapshot is given a default auto delete priority of *1* at creation.

**B****basic mode**

A means of entering CLI commands on the CLI client that requires specifying IP address and login information for each command. Additional output formatting options are available in basic mode.

**best effort mode**

A mode of remote mirroring in which I/O operation is not suspended when communication between a primary and secondary volume is broken.

**C****call home**

A communication link established between the storage system and a service provider. The storage product can use this link to call IBM or to another service provider when it requires service. With access to the storage system, service personnel can perform service tasks, such as viewing error logs and problem logs or initiating trace and dump retrievals.

**clearing events**

The process of stopping the recurring event notification of alerting events.

**CLI**

The command-line interface (CLI). See *command-line interface (CLI)*

**CLI client**

The system on which the CLI command is entered.

**CLI identification parameters**

Parameters that identify the user issuing the command and the IBM FlashSystem A9000 or IBM FlashSystem A9000R system (if any) on which the command is to run. CLI identification parameters can be specified:

- By entering them at the beginning of an interactive mode session
- In a configuration file
- When running a command in basic mode
- When running a list of commands as a batch

**CLI system command**

An CLI command that is sent to the IBM FlashSystem A9000 or IBM FlashSystem A9000R system for processing.

**CLI utility command**

A CLI command that is issued on the CLI client. A CLI utility command is not sent to an IBM FlashSystem A9000 or IBM FlashSystem A9000R system for processing. CLI utility commands are used for setting up configurations on the CLI client and for queries (for example, of software version) that can be processed on the client.

**command-line interface (CLI)**

The non-graphical user interface used to interact with the system through set commands and functions. The CLI for IBM FlashSystem A9000 and A9000R.

**completion code**

The returned message sent as a result of running CLI commands.

**consistency group**

A cluster of specific volumes for which a snapshot can be taken simultaneously as a group, thus creating a synchronized snapshot. The volumes in a consistency group are grouped into a single volume set. Snapshots can be taken for the volume set in multiple snapshot sets under the specific consistency group. See also *snapshot set*, *volume set*.

**coupling**

A primary volume and a secondary volume connected together through mirroring definitions.

**D****data availability**

The degree to which data is available when needed. Availability is typically measured as a percentage of time in which the system is able to respond to data requests (for example, 99.999% available).

**data module**

A module dedicated to data storage. A fully populated rack contains nine dedicated data modules, each with 12 disks.

**default storage pool**

The default storage pool when a volume is created.

**destination**

See *event destination*.

**E****escalation**

A process in which event notifications are sent to a wider list of event destinations because the event was not cleared within a certain time.

**event destination**

An address for sending event notifications.

**event notification rule**

A rule that determines which users are to be notified, for which events and by what means.

**event notification**

The process of notifying a user about an event.

**event**

A user or system activity that is logged (with an appropriate message).

**F****fabric**

The hardware that connects workstations and servers to storage devices in a SAN. The SAN fabric enables any-server-to-any-storage device connectivity through the use of Fibre Channel switching technology.

**FC-AL**

Also known as arbitrated loop. A Fibre Channel topology that requires no Fibre Channel switches. Devices are connected in a one-way loop fashion.

**FC-HBA**

Fibre Channel host bus adapter.

**FC**

See *Fibre Channel*.

**Fibre Channel**

Serial data transfer architecture developed by a consortium of computer and mass storage device manufacturers and now being standardized by ANSI.

**functional area**

One of the high-level groupings of icons (functional modules) of the left pane in IBM Hyper-Scale Manager screen (for example, Monitor, Configuration, or Volume management). See *functional module*.

**functional module**

One of the icons of a functional area, on the left pane in IBM Hyper-Scale Manager screen. For example, System (under Monitor) or Hosts and LUNs (under Configuration). See *functional area*.

**G****Graphical user interface (GUI)**

On-screen user interface supported by a mouse and a keyboard.

**GUI**

See *graphical user interface (GUI)*.

**H****H/W**

Hardware.

**HBA**

Host bus adapter.

**host interface module**

The interface data module serves external host requests with the ability to store data. A fully populated rack has six interface data modules.

**host**

A port name of a host that can connect to the system. The system supports Fibre Channel and iSCSI hosts.

**I****I/O**

input/output.

**image snapshot**

A snapshot that has never been unlocked. It is the exact image of the master volume it was copied from, at the time of its creation. See also *snapshot*.

**interactive mode**

A means of entering CLI commands on the CLI client in which the IP address, user, and password information does not need to be specified for each command.

**Internet Protocol**

Specifies the format of packets (also called datagrams), and their addressing schemes. See also *Transmission Control Protocol (TCP)*.

**IOPs**

input/output (I/O) per second.

**IP**

See *Internet Protocol*.

## **iSCSI**

Internet SCSI. An IP-based standard for linking data storage devices over a network and transferring data by carrying SCSI commands over IP networks.

## **L**

### **latency**

Amount of time delay between the moment an operation is issued, and the moment it is committed.

### **LDAP**

Lightweight Directory Access Protocol.

### **LDAP attribute**

A property of an LDAP object, with a single or multiple values. A special object attribute is designated by an LDAP administrator to hold user group memberships values corresponding to IBM FlashSystem A9000 and A9000R roles.

### **LDAP authentication**

A method for authenticating users by validating the submitted credentials against data stored on an LDAP directory.

### **LDAP directory**

A hierarchical database stored on an LDAP server and accessed through LDAP calls.

### **LDAP mapping**

An association of data on the LDAP server (a specific LDAP attribute) and data on the IBM FlashSystem A9000 or IBM FlashSystem A9000R system. The mapping is used to determine which access rights to grant to an authenticated LDAP user.

### **LDAP server**

A server that provides directory services through LDAP.

### **LDAP status**

The status of an LDAP server.

### **load balancing**

Even distribution of load across all components of the system.

### **locking**

Setting a volume (or snapshot) as unwritable (read-only).

### **LUN map**

A table showing the mappings of the volumes to the LUNs.

## **LUN**

Logical unit number. Exports a systems volume into a registered host.

## **M**

### **main power cable**

The electrical connection between the ac power source and the automatic transfer switch (ATS).

### **mandatory mode**

A mode of remote mirroring in which I/O operation stops whenever there is no communication to the secondary volume.

### **master volume**

A volume that has snapshots is called the master volume of its snapshots.

## **MIB**

Management Information Base. A database of objects that can be monitored by a network management system. SNMP managers use standardized MIB formats to monitor SNMP agents.

### **Microsoft Active Directory**

See *Active Directory*

### **mirror volume**

A volume that contains a backup copy of the original volume.

**mirroring**

See *remote mirroring*.

**modified State**

A snapshot state. A snapshot in modified state can never be used for restoring its master volume.

**multipathing**

Used for direct access from host-interface modules to any volume.

**P****peer**

Denotes a constituent side of a coupling. Whenever a coupling is defined, a designation is specified for each peer - one peer is designated primary and the other is designated secondary.

**pool**

See *storage pool*.

**primary volume**

A volume that is mirrored for backup on a remote storage system.

**R****rack**

The cabinet that stores all of the hardware components of the system.

**remote mirroring**

The process of replicating a volume on a remote system.

**remote target connectivity**

A definition of connectivity between a port set of a remote target and a module on the local storage system.

**remote target**

An additional storage system used for mirroring, data migration, and so on.

**role**

The actual role that the peer is fulfilling as a result of a specific condition, either a master or a subordinate.

**rule**

See *event notification rule*.

**S****SAN**

Storage area network.

**SCSI**

Small computer system interface.

**secondary volume**

A volume that serves as a backup of a primary volume.

**Simple Network Monitor Protocol**

A protocol for monitoring network devices. See also *MIB*, *SNMP agent*, *SNMP manager*, *SNMP trap*.

**SMS gateway**

An external server that is used to send SMSs.

**SMTP gateway**

An external host that is used to relay email messages through the SMTP protocol.

**snapshot set**

The resulting set of synchronized snapshots of a volume set in a consistency group. See also *consistency group*, *Volume set*.

**snapshot**

A point-in-time snapshot or copy of a volume. See also *image snapshot*.



**SNMP agent**

A device that reports information through the SNMP protocol to SNMP managers.

**SNMP manager**

A host that collects information from SNMP agents through the SNMP protocol.

**SNMP trap**

An SNMP message sent from the SNMP agent to the SNMP manager, where the sending is initiated by the SNMP agent and not as a response to a message sent from the SNMP manager.

**SNMP**

See *Simple Network Monitor Protocol*.

**snooze**

The process of sending recurring event notifications until the events are cleared.

**storage pool**

A reserved area of virtual disk space serving the storage requirements of the volumes.

**Sync Job**

A synchronization procedure run by the master at specified user-defined intervals, entailing synchronization between the master and the subordinate.

**synchronization**

The process of making the primary volume and secondary volume identical after a communication downtime or upon the initialization of the mirroring.

**T****target**

See *remote target*.

**TCP/IP**

See *Transmission Control Protocol, Internet Protocol*.

**thin provisioning**

The ability to define logical volume sizes that are much larger than the physical capacity installed on the system.

**Transmission Control Protocol**

Transmission Control Protocol (TCP) on top of the Internet Protocol (IP) establishes a virtual connection between a destination and a source over which streams of data can be exchanged. See also *IP*.

**trap**

See *SNMP trap*.

**U****unassociated volume**

A volume that is not associated with a consistency group. See *Consistency group*.

**uninterruptible power supply**

Provides battery backup power for a determined time, so that the system can power down in a controlled manner, on the occurrence of a lengthy power outage.

**V****volume cloning**

Creating a snapshot from a volume.

**volume set**

A cluster of specific volumes in a consistency group, for which snapshots are taken simultaneously, thus, creating a synchronized snapshot of all of them. Snapshots of the volume set can be taken into multiple snapshot sets of the specific consistency group. See also *Snapshot set, Volume set*.

**volume**

A logical address space, having its data content stored on the systems disk drives. A volume can be virtually any size as long as the total allocated storage space of all volumes does not exceed the net capacity of the system. A volume can be exported to an attached host through a LUN. A volume can be exported to multiple hosts simultaneously. See also *Storage pool*, *Unassociated volume*.

**W****WWPN**

Worldwide port name

**X****XDRP**

The disaster recovery program for the IBM FlashSystem A9000 and A9000R – The remote mirror feature of the IBM FlashSystem A9000 and A9000R.

---

# Index

## A

AssociatorNames [17](#)  
Associators [16](#)

## B

Block Services  
    object model [32](#)

## C

CIM agent  
    components [3](#)  
    concepts [2](#)  
    IBM FlashSystem A9000 and A9000R [7](#)  
    limitations [7](#)  
    overview [1](#)  
    port number [7](#)  
    security [5](#)  
CIM classes  
    CIM\_AuthorizedPrivilege [46](#)  
    CIM\_FCPort [46](#)  
    CIM\_SCSIProtocolController [46](#)  
    CIM\_SCSIProtocolEndPoint [46](#)  
    CIM\_StorageHardwareID [46](#)  
    CIM\_StorageVolume [46](#)  
    CIM\_SystemSpecificCollection [46](#)  
CIM\_AuthorizedPrivilege [46](#)  
CIM\_ERR\_ACCESS\_DENIED [20](#)  
CIM\_ERR\_ACCESS\_DENIED [10–22](#)  
CIM\_ERR\_ALREADY\_EXISTS [12](#)  
CIM\_ERR\_FAILED [10–22](#)  
CIM\_ERR\_INVALID\_CLASS [20](#)  
CIM\_ERR\_INVALID\_NAMESPACE [10–22](#)  
CIM\_ERR\_INVALID\_PARAMETER [10–22](#)  
CIM\_ERR\_INVALID\_CLASS [11–16, 18–20](#)  
CIM\_ERR\_INVALID\_QUERY [16](#)  
CIM\_ERR\_NO\_SUCH\_PROPERTY [19, 20](#)  
CIM\_ERR\_NOT\_FOUND [11, 12, 19–21](#)  
CIM\_ERR\_NOT\_SUPPORTED [16, 22](#)  
CIM\_ERR\_QUERY\_FEATURE\_NOT\_SUPPORTED [16](#)  
CIM\_ERR\_QUERY\_LANGUAGE\_NOT\_SUPPORTED [16](#)  
CIM\_ERR\_TYPE\_MISMATCH [20](#)  
CIM\_FCPort [46](#)  
CIM\_SCSIProtocolController [46](#)  
CIM\_SCSIProtocolEndPoint [46](#)  
CIM\_StorageHardwareID [46](#)  
CIM\_StorageVolume [46](#)  
CIM\_SystemSpecificCollection [46](#)  
CIMOM  
    return codes [22](#)  
client application  
    receiving CIMOM error codes [22](#)  
clusters [44](#)  
component  
    definitions [4](#)

conformance testing  
    SCVMM [73](#)  
    SNIA [73](#)  
CreateClass [22](#)  
CreateInstance [12](#)

## D

DeleteClass [22](#)  
DeleteInstance [11](#)  
DeleteQualifier [22](#)

## E

EnumerateClasses [13](#)  
EnumerateInstanceNames [15](#)  
EnumerateInstances [14](#)  
EnumerateQualifiers [22](#)  
error codes [22](#)  
ExecuteQuery [16](#)

## F

forums [ix](#)

## G

GetClass [10](#)  
GetInstance [11](#)  
GetProperty [20](#)  
GetQualifier [21](#)  
group management  
    Replication Services [58](#)

## H

hosts [44](#)  
HTTP  
    status messages [22](#)

## I

IBM FlashSystem A9000 and A9000R CIM classes  
    IBMTSDS\_SystemSpecificCollection [46](#)  
    IBMTSDS\_FCPort [46](#)  
    IBMTSDS\_Privilege [46](#)  
    IBMTSDS\_SCSIProtocolController [46](#)  
    IBMTSDS\_SEVolume [46](#)  
    IBMTSDS\_StorageHardwareID [46](#)  
IBM FlashSystem A9000 and A9000R Open API  
    components [4](#)  
    overview [1](#)  
IBMTSDS\_FCPort [46](#)  
IBMTSDS\_Privilege [46](#)  
IBMTSDS\_SCSIProtocolController [46](#)  
IBMTSDS\_SEVolume [46](#)

IBMTSDS\_StorageHardwareID [46](#)  
IBMTSDS\_SystemSpecificCollection [46](#)

## J

Job Control  
    object model [68](#)

## L

LUN mapping [44](#)  
LUN masking [44](#)

## M

Masking and Mapping  
    object model [46](#)  
methods  
    AssociatorNames [17](#)  
    Associators [16](#)  
    CreateClass [22](#)  
    CreateInstance [12](#)  
    DeleteClass [22](#)  
    DeleteInstance [11](#)  
    DeleteQualifier [22](#)  
    EnumerateClasses [13](#)  
    EnumerateInstanceNames [15](#)  
    EnumerateInstances [14](#)  
    EnumerateQualifiers [22](#)  
    ExecuteQuery [16](#)  
    GetClass [10](#)  
    GetInstance [11](#)  
    GetProperty [20](#)  
    GetQualifier [21](#)  
    ModifyClass [22](#)  
    ModifyInstance [12](#)  
    ReferenceNames [19](#)  
    References [18](#)  
    SetProperty [20](#)  
    SetQualifier [21](#)  
models  
    Block Services [32](#)  
    Job Control [68](#)  
    Masking and Mapping [46](#)  
    Replication Services [55](#)  
    Thin Provisioning [69](#)  
ModifyClass [22](#)  
ModifyInstance [12](#)

## P

packages  
    Block Services [32](#)  
    Job Control [68](#)  
    Masking and Mapping [46](#)  
    Replication Services [55](#)  
    Thin Provisioning [69](#)  
parameters  
    AssocClass [16](#), [17](#)  
    ClassName [10](#), [13–15](#)  
    DeepInheritance [13](#), [14](#)  
    IncludeClassOrigin [10](#), [11](#), [13](#), [14](#), [16](#), [18](#)  
    IncludeQualifiers [10](#), [13](#)

parameters (*continued*)  
    Instance [12](#)  
    InstanceName [11](#), [20](#)  
    LocalOnly [10](#), [13](#)  
    ObjectName [16–19](#)  
    Property [20](#)  
    QualifierName [21](#)  
    Query [16](#)  
    QueryLanguage [16](#)  
    ResultClass [16–19](#)  
    ResultRole [16](#), [17](#)  
    Role [16–19](#)  
    SetName [21](#)  
PDFs [ix](#)  
port number [7](#)  
ports [44](#)  
profiles  
    Block Services [31](#)  
    Masking and Mapping [44](#)  
publications [ix](#)

## R

ReferenceNames [19](#)  
References [18](#)  
related information [ix](#)  
Replication management [60](#)  
Replication Services  
    group management [58](#)  
    object model [55](#)  
return code [22](#)  
return values  
    CIM\_ERR\_ACCESS\_DENIED [10–22](#)  
    CIM\_ERR\_ALREADY\_EXISTS [12](#)  
    CIM\_ERR\_FAILED [10–22](#)  
    CIM\_ERR\_INVALID\_NAMESPACE [10–22](#)  
    CIM\_ERR\_INVALID\_PARAMETER [10–22](#)  
    CIM\_ERR\_INVALID\_CLASS [11–16](#), [18–20](#)  
    CIM\_ERR\_INVALID\_QUERY [16](#)  
    CIM\_ERR\_NO\_SUCH\_PROPERTY [19](#), [20](#)  
    CIM\_ERR\_NOT\_FOUND [11](#), [12](#), [19–21](#)  
    CIM\_ERR\_NOT\_SUPPORTED [16](#), [22](#)  
    CIM\_ERR\_QUERY\_FEATURE\_NOT\_SUPPORTED [16](#)  
    CIM\_ERR\_QUERY\_LANGUAGE\_NOT\_SUPPORTED [16](#)  
    CIM\_ERR\_TYPE\_MISMATCH [20](#)  
descriptions [22](#)

## S

SCVMM testing [73](#)  
secure connection  
    port number [7](#)  
SetProperty [20](#)  
SetQualifier [21](#)  
SNAI testing [73](#)  
Storage Management Initiative Specifications (SMI-S)  
    overview [1](#)  
storage pools [31](#), [44](#)

## T

Thin Provisioning  
    object model [69](#)

trademarks [76](#)

## V

volumes [31](#), [44](#)







Printed in USA

SC27-8561-04

