IBM TotalStorage™ SAN File System
(based on IBM Storage Tank™ technology)

# System Management API Guide and Reference

*Version 1 Release 1*

GA27-4315-00

IBM TotalStorage™ SAN File System
(based on IBM Storage Tank™ technology)

# System Management API Guide and Reference

*Version 1 Release 1*

GA27-4315-00

**NOTE**
Before using this information and the product it supports, read the general information in Appendix C, "Notices", on page 263.

**First Edition (November 2003)**

This edition applies to the IBM TotalStorage SAN File System and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office servicing your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for reader's comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

International Business Machines Corporation
Design & Information Development
Department CGFA
PO Box 12195
Research Triangle Park, NC 27709–9990
U.S.A.

You can also submit comments by selecting **Feedback** at www.ibm.com/storage/support.

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.

# Contents

# About this guide

This guide introduces the Administrative agent for SAN File System. It describes the Administrative agent object model, the classes and properties that make up the model, and methods that the classes provide to implement the model.

- Chapter 1, "Getting Started", on page 1 provides an overview of CIM and SAN File System concepts and introduces the Administrative agent object model.
- Chapter 2, "Managing SAN File System", on page 53 describes how to perform tasks for managing SAN File System using the object model.
- Chapter 3, "Administrative agent methods", on page 77 describes intrinsic and extrinsic methods.
- Chapter 4, "Administrative agent object classes", on page 91 describes the classes that make up the object model of the Administrative agent and their properties and methods.
- Chapter 5, "SAN File System MOF", on page 165 shows the actual MOF.
- The appendices provide the following additional information:
  - Accessibility features of the SAN File System console and help system
  - SNMP Trap MIB
  - Notices

## Who should use this guide

This guide should be used by application programmers writing third-party applications for SAN File System.

**Note:** This document contains information and procedures intended for only selected business partners. Contact your IBM representative before using this publication.

Application programmers should have experience in at least the following skills, or have access to personnel with experience in these skills:

- Object-oriented programming
- CIM-based application programming
- Networking and network management
- SAN management

**Related topics:**

- Chapter 1, "Getting Started", on page 1

## Notices in this guide

The following notices are contained with the this guide and convey these specific meanings:

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

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

## Publications

The following publications are available in the SAN File System library. They are provided in softcopy on the *IBM TotalStorage SAN File System Publications CD* that came with your storage engine and at www.ibm.com/storage/support. To use the CD, insert it in the CD-ROM drive. If the CD does not launch automatically, follow the instructions on the CD label.

**Note:** The softcopy version of these publications are accessibility-enabled for the IBM Home Page Reader.

- *IBM TotalStorage SAN File System Release Notes*

  This document provides any changes that were not available at the time the publications were produced. This document is available only from the technical support Web site: www.ibm.com/storage/support
- *IBM Safety Information — Read This First*, SD21-0030

  This document provides translated versions of general safety notices and should be read before using this product. This document is provided only in hardcopy.
- *IBM Statement of Limited Warranty*

  This publication describes the IBM statement of limited warranty as it applies to the SAN File System Model 1RX storage engine.
- *IBM TotalStorage SAN File System Software License Information*

  This publication provides multilingual information regarding the software license for IBM TotalStorage SAN File System Software.
- *IBM TotalStorage SAN File System Administrator's Guide and Reference*, GA27-4317

  This publication introduces the concept of SAN File System, and provides instructions for configuring, managing, and monitoring the system using the SAN File System console and Administrative command-line interfaces. This book also contains a commands reference for tasks that can be performed at the administrative and client command-line interfaces.
- *IBM TotalStorage SAN File System Maintenance and Problem Determination Guide*, GA27-4318

  This publication provides instructions for adding and replacing hardware components, monitoring and troubleshooting the system, and resolving hardware and software problems.

  **Note:** This document is intended only for trained personnel.
- *IBM TotalStorage SAN File System Messages Reference*, GC30-4076

  This publication contains message description and resolution information for errors that can occur in SAN File System software.
- *IBM TotalStorage SAN File System Planning, Installation and Configuration Guide*, GA27-4316

  This publication provides detailed procedures to plan the installation and configuration of SAN File System, set up and cable the hardware, perform the minimum required configuration, migrate existing data, and upgrading software.
- *Rack Installation Instructions*

  This publication provides instructions for installing the Model 1RX in a rack.
- *IBM TotalStorage SAN File System System Management API Guide and Reference*, GA27-4315

  This publication contains guide and reference information for using the CIM Proxy API, including common and SAN File System-specific information.

**Note:** This document contains information and procedures intended for only selected IBM Business Partners. Contact your IBM representative before using this publication.

- *Subsystem Device Driver User's Guide for the IBM TotalStorage Enterprise Storage Server and the IBM TotalStorage SAN Volume Controller*, SC26-7540

  The Subsystem Device Driver (SDD) provides the multipath configuration environment support for a host system that is attached to an IBM TotalStorage Enterprise Storage Server® (ESS), IBM TotalStorage SAN Volume Controller or IBM TotalStorage SAN File System. This book provides step-by-step procedures on how to install, configure, and use SDD for the host systems.

  **Note:** SAN File System supports the version of the Subsystem Device Driver that is shipped with the program product.

- *IBM TotalStorage Translated Safety Notices*, GA67-0043

  This publication contains translated versions of hardware caution and danger statements that appear in the publications in this library. Each caution and danger statement has an assigned number that you can use to locate the corresponding statement in your native language.

## Web sites

**Web sites:**

The following Web sites have additional and up-to-date information about SAN File System:

- www.ibm.com/storage/support
- www.ibm.com/storage/software/virtualization/sfs

The following Web site has additional and up-to-date information about CIM:

http://www.dmtf.org

The following Web site has additional and up-to-date information about SLP:

http://www.openslp.org/

# Chapter 1. Getting Started

This chapter introduces the Common Information Model (CIM) agent for SAN File System, known as the Administrative agent. The Administrative agent implements an object-oriented management interface over Hypertext Transfer Protocol (HTTP). It conforms to CIM 2.7 and plans to follow the Storage Management Initiative Specification (SMI-S) as it develops. The Administrative agent's managed object format (MOF) derives from CIM standard models where they apply to SAN File System.

This chapter describes key concepts related to CIM, SAN File System, and the Administrative agent. It also presents functional views of the Administrative agent object model and describes programming considerations such as accessing the object model and invoking dynamic and static methods.

**Related topics:**

- "CIM concepts"
- "SAN File System concepts" on page 4
- "Administrative agent for SAN File System" on page 29
- "Functional view of the Administrative agent" on page 29
- "Programming considerations" on page 51

## CIM concepts

This section provides an overview of the Common Information Model (CIM) and CIM-related concepts. It also describes a CIM agent, in general, and the Storage Management Initiative Specification (SMI-S).

**Related topics:**

- "CIM"
- "CIM-related concepts" on page 2
- "CIM agent" on page 2
- "Storage Management Initiative Specification" on page 3

### CIM

The Common Information Model (CIM) is a set of standards from the Distributed Management Task Force Inc. (DMTF). It provides a conceptual framework for storage management and an open approach to the design and implementation of storage systems, applications, databases, networks, and devices.

The CIM specifications provide the language and the methodology for describing management data. Specifically, the CIM defines common object classes, associations, and methods. Member vendors can use those objects and extend them to specify how data should be processed and organized in a specific managed environment.

**Related topics:**

- "CIM-related concepts" on page 2
- "CIM agent" on page 2

- "Storage Management Initiative Specification" on page 3

## CIM-related concepts

The CIM specifications use the following concepts and terminology to describe the various object models:

**Schema**
A group of object classes defined for and applicable to a single namespace. Within the CIM agent, the supported schemas are the ones that are loaded through the managed object format (MOF) compiler.

**Namespace**
The scope within which a CIM schema applies.

**Object name**
An object that consists of a namespace path and a model path. The namespace path provides access to the CIM implementation managed by the CIM agent, and the model path provides navigation within the implementation.

**Class** The definition of an object within a specific hierarchy. An object class can have properties and methods and serve as the target of an association.

**Property**
An attribute that is used to characterize instances of a class.

**Key** A property that is used to provide a unique identifier for an instance of a class. Key properties are marked with the Key qualifier. (DMTF CIM Tutorial Glossary).

**Method**
A way to implement a function on a class.

**Qualifier**
A value that provides additional information about a class, association, indication, method, method parameter, instance, property, or reference.

**Managed Object Format (MOF)**
A compiled language for defining classes and instances. A MOF compiler offers a textual means of adding data to the CIM Object Manager repository. MOF eliminates the need to write code, thus providing a simple and fast technique for modifying the CIM Object Manager repository (DMTF CIM Tutorial Glossary).

**Related topics:**

- "CIM" on page 1

## CIM agent

A CIM agent is a specific piece of software that handles CIM requests in an embedded model.

**Components:**

A CIM agent typically contains (or interacts) with the following components:

**The agent**
An open-system standard that interprets CIM requests and responses as they are transferred between the client application and the device.

**Client application**
A storage management program that initiates CIM requests to the CIM agent for the device.

**CIM object manager (CIMOM)**
The common conceptual framework for data management that receives, validates, and authenticates the CIM requests from the client application and then directs the requests to the appropriate component or device provider.

**Service location protocol (SLP)**
A directory service that the client application calls to locate the CIMOM.

**Device provider**
A device-specific handler and serves as a plug-in for the CIM. That is, the CIMOM uses the handler to interface with the device.

**Device**
The storage server that processes and hosts the client application requests.

In the case of the CIM agent for SAN File System, known as the Administrative agent, its components include the agent, the CIMOM, and the device provider. The Administrative agent is separate from and interacts with the SLP daemon, client application and device.

**CIM agent at work:**

The client application locates the CIMOM by calling an SLP directory service. When an application first invokes the CIMOM, it registers itself to the SLP and supplies its location information, including IP address, port number, and the type of service that it provides. With this information, the client application starts to directly communicate with the CIMOM.

The client application then sends CIM requests to the CIMOM. As requests arrive, the CIMOM validates and authenticates each request. It then directs the requests to the appropriate functional component of the CIMOM or a device provider. The provider makes calls to a device-unique programming interface on behalf of the CIMOM to satisfy client application requests.

**Related topics:**


- "Administrative agent for SAN File System" on page 29


## Storage Management Initiative Specification

The Storage Management Initiative Specification (SMI-S) is a design specification of the Storage Management Initiative (SMI) launched by the Storage Networking Industry Association (SNIA). It specifies a secure and reliable interface that allows storage management systems to identify, classify, monitor, and control physical and logical resources in a storage area network (SAN). The interface is intended as a solution that integrates the various devices to be managed in a SAN and the tools used to manage them. SMI-S is based on a number of existing technologies or industry standards that include the following:

**Common Information Model (CIM)**
An object model for data storage and management developed by the Distributed Management Task Force (DMTF). CIM makes it possible to organize devices and components of devices in an object-oriented pattern.

**Web-Based Enterprise Management (WBEM)**
A tiered enterprise management architecture also developed by the DMTF. This architecture provides the management design framework that consists of devices, device providers, the object manager, and the messaging protocol for the communication between client applications and the object manager. In the case of the CIM, the object manager is the CIMOM and the messaging protocol is the CIM over HTTP technology. The CIM over HTTP approach specifies that the CIM data is encoded in XML and sent in specific messages between the client applications and the CIMOM over the TCP/IP network in a SAN.

**Service Location Protocol (SLP)**
A directory service that the client application calls to locate the CIMOM.

Intended to be an industry standard, SMI-S extends the generic capabilities of the CIM, the WBEM, and the SLP to implement storage networking interoperability. For example, the WBEM is expanded to provide provisions for security, resource-locking management, event notification, and service discovery.

**Related topics:**
- "CIM" on page 1
- "CIM-related concepts" on page 2

## SAN File System concepts

This section discusses concepts that will help you understand how SAN File System works. Becoming familiar with the SAN File System components and understanding the concepts in this section enables you to use SAN File System most effectively.

**Related topics:**
- "Administrative server" on page 5
- "Alerts" on page 5
- "Cluster" on page 6
- "Components" on page 9
- "Engines" on page 11
- "File placement" on page 23
- "Filesets" on page 11
- "FlashCopy images" on page 13
- "Locks and leases" on page 16
- "Logs" on page 16
- "Metadata server" on page 18
- "Policies and rules" on page 24
- "SNMP" on page 20
- "Storage management" on page 22
- "Storage pools" on page 20
- "User interfaces" on page 25
- "User roles" on page 26
- "Volumes" on page 27

## Administrative server

The SAN File System *Administrative server*, which is based on a Web server software platform, is made up of the following parts:

- The GUI Web server, which renders the Web pages that make up the SAN File System console. The console is a Web-based user interface. It can be accessed using any standard Web browser, such as Mozilla or Microsoft® Internet Explorer, that has network access to the engines that host the Metadata servers in a server cluster.
- The Administrative agent, which implements all of the management logic for the GUI, CLI, and CIM interfaces, as well as performing administrative authorization and authentication against LDAP. The Administrative agent processes all management requests initiated by an administrator from the SAN File System console, as well as requests initiated from the administrative command line interface, which is called *tanktool*. The Agent communicates with the SAN File System Metadata servers, the OS, the RSA II card, the LDAP, and Administrative agents on other nodes in the cluster when processing requests.

An Administrative server interacts with a Metadata server through the Administrative agent. When you issue a request, the Administrative agent checks with an LDAP server, which must be installed in your environment, to authenticate the user ID and password and to verify whether the user has the authority (is assigned the appropriate role) to issue a particular request. After authenticating a user, the Administrative agent interacts with the Metadata server on behalf of that user to process the request. This same system of authentication and interaction is also available to third-party CIM clients to manage SAN File System.

To ensure high availability, an Administrative server resides on each engine of a cluster. All requests are received by the Administrative server that runs on the same engine as the master Metadata server. This is the primary Administrative server. However, requests can also be processed by Administrative servers running on other engines, and all requests are redirected to an Administrative server running on another engine if the primary Administrative server is not available.

**Related topics:**


- "Engines" on page 11
- "Metadata server" on page 18
- "Cluster" on page 6
- "User interfaces" on page 25
- "User roles" on page 26


## Alerts

An event is a happening in the Metadata server or cluster, such as a change in state from online to offline. An *alert* is a message that can be generated for an event. It informs an administrator about certain conditions, such as a fileset or a storage pool reaching or exceeding its threshold.

Events are recorded as messages in the cluster log. These messages can be viewed by an administrator using the SAN File System console or by using administrative commands.

A trap is a notification mechanism to convey the occurrence of an event. An administrator can choose to set configuration parameters that determine whether Simple Network Management Protocol (SNMP) trap messages are generated for

events. SNMP trap messages notify administrators of events asynchronously, and eliminate the need for an administrator to frequently view messages in the cluster log to determine the state of the SAN File System cluster.

*Table 1. Alerts, events, and traps*

| | **Alerts** | **Traps** |
|---|---|---|
| Description | Warns of a significant event on a Metadata server or cluster. Also informs about condition changes such as a change in state to offline, or approaching storage capacity | Optional notification method that notifies the administrator of events asynchronously |
| Delivery | Sends a generated message to the terminal | Sends the administrator a generated message directly, either locally or remotely |

The first configuration parameter determines where SNMP trap messages are sent. An administrator specifies a list of SNMP Managers that are the recipients of any SNMP trap messages. The list includes the IP address, port number, version of SNMP, and community string for one or two managers. If no SNMP Managers are specified, no SNMP trap messages are sent.

The second parameter specifies which types of event messages also generate SNMP trap messages. An administrator can specify any combination of messages classified as informational, warning, error, or severe. If no severity types are specified, no SNMP trap messages are sent.

In addition, if you have the call home feature activated, that feature generates a specific type of SNMP trap whenever a Metadata server encounters an event that requires notification to IBM support personnel. When a Metadata server encounters such an event, it sends an SNMP trap Protocol Data Unit (PDU) to the Master console, which then parses and converts the trap into a Simple Mail Transfer Protocol (SMTP) e-mail message. The e-mail message is then sent to a known SMTP mail server. Finally, the e-mail message is forwarded to the IBM RETAIN system for processing by support personnel. Note that all of the SNMP-related configuration parameters must be set properly for the call-home feature to work.

**Related topics:**

• "SNMP" on page 20

## Cluster

A SAN File System *cluster* is a set of Metadata servers and engines with one Metadata server running on each engine. The servers in a cluster communicate with each other and with SAN File System clients over your existing IP network. A cluster provides a single point of control for administrative and service operations.

Each cluster has one master Metadata server, which is designated by an administrator, and one or more subordinate Metadata servers. The master server maintains cluster state and is the focal point for most administrative services. The maximum number of Metadata servers and engines that you can have in a cluster is eight.

**Note:** Although you cannot purchase SAN File System with only one Metadata server and engine, you can run a single engine system if all of the other

engines in your cluster have failed (for example, if you have only two engines, and one of them fails), or if you want to bring down all of the engines except one before performing scheduled maintenance tasks. To keep the SAN File System global namespace available, you must set the single remaining active server to be the master server. If the last remaining active server in a cluster is a subordinate server, it eventually goes into a wait state because it cannot contact the master, and the entire global namespace becomes inaccessible.

**Server interactions:**

The Metadata servers in a cluster interact with each other for a variety of reasons. For example, they exchange *heartbeats*, which are messages sent periodically from one server to another so that each knows that the other is still active. If a server stops sending heartbeats for a specific period of time (which is set with heartbeat parameters), the other servers reform the server cluster without it.

The master server needs to communicate with its subordinate servers to perform many administrative tasks, such as supplying the servers with the current workload map and querying server status. It also needs to contact them to process requests from administrators to perform tasks such as draining volumes (when an administrator removes a volume from a storage pool) and creating FlashCopy images of filesets.

Subordinate servers need to initiate contact with the master server for specific tasks, such as acquiring more space or obtaining file placement policy information.

**Cluster workload:**

Each server in a cluster, including the master server, is assigned a *workload*; that is, each server is responsible for providing metadata and locks to clients whenever they request access to data that resides in one or more filesets assigned to that particular server. A *fileset* is a subset of the SAN File System global namespace and serves as the unit of workload for Metadata servers.

During client setup, a client is given the address of one of the Metadata servers in your server cluster for initial contact and server discovery. Then, when the client issues a request to access data, it is automatically directed to the appropriate server to obtain the metadata and locks required to access the data.

**Creating and assigning filesets:** An administrator can create any number of filesets within the global namespace and assign those filesets to specific servers. Each server receives a workload map from the master server and knows which filesets are assigned to it and to all other servers in the cluster. Fileset assignments are fixed, which means that after an administrator assigns a fileset to a specific Metadata server, it remains assigned to that server until the administrator chooses to reassign it to another Metadata server.

**Balancing the workload:** An administrator has complete control over how the global namespace workload is distributed. For example, an administrator can choose to assign an equal number of filesets to each server. Assuming that the metadata activity level for each fileset is relatively the same, this balances the workload evenly across all of the servers in the cluster.

To ensure that each server handles a share of the entire workload, the administrator should create at least one fileset for each server. Creating additional

filesets provides the administrator with greater flexibility in assigning and reassigning filesets to achieve optimal results.

Files are typically placed in a particular fileset based on some association that the files share, such as all of the files are used by a specific application or are used by a specific group. When distributing filesets among Metadata servers, an administrator can consider whether there are some filesets in which many files are constantly being created, deleted, and updated, thus requiring numerous and frequent metadata updates. If several of these filesets are identified, the administrator can choose to distribute them appropriately among the servers to help further balance the workload.

**Handling server and engine failures:**

How SAN File System handles a failure within a server cluster depends on whether it is a soft or a hard failure.

**Soft failures:** A soft failure requires no administrator intervention and is recoverable by simply restarting the server software.

Soft failures are detected in different ways depending on their cause. The Administrative server, which runs on the same engine as the master Metadata server and in standby mode on each subordinate server, provides an optional Metadata server restart service that monitors the Metadata server software and restarts it if necessary. If a Metadata server hangs, an internal server thread detects that condition and enables the rebooting of the software. If the operating system on an engine crashes or hangs, SAN File System can reboot the operating system. Then, the Metadata server restart service is automatically started, and in turn, restarts the Metadata server on the engine. When a Metadata server is restarted after a soft failure, it resumes serving the same workload that it was serving before the failure.

If a subordinate Metadata server fails, only the filesets that are assigned to that server are temporarily unavailable. If the master Metadata server fails, all of the subordinate servers eventually pause, either because they require a service from the master server or because they are no longer receiving a heartbeat from the master server, and the entire global namespace is unavailable until the master server is restarted.

When a client has an active session with a Metadata server and that server fails, the client stops receiving responses for transactions and lease renewal attempts. However, the client remains active. The client loses its lease (which is the amount of time a client can hold its locks on data before contacting the server again to renew its lease) and any locks it had obtained from that server. After the server is restarted, the client can contact the server and renew its lease. Then, it can reassert its locks (get back all the locks it had before the server failure) and refresh its metadata cache as needed. A Metadata server provides a grace period for lock reassertion to allow clients to reassert their locks before allowing other clients to obtain new ones. Applications running on the client experience a pause in service during the restart and recovery period.

**Hard failures:** A hard failure is one that requires intervention by an administrator, as in the case of a hardware failure. A hard failure is detected through a heartbeat timeout. When a Metadata server stops sending heartbeats for the amount of time specified with heartbeat parameters, an SNMP trap message is generated (if an

administrator has set all SNMP-related parameters appropriately and the failure is graceful). An administrator must verify that the server is down and take actions to restart it.

The server must fail gracefully in order to generate an SNMP trap to announce its failure. If the server does not fail gracefully, an operational server will detect that the server failed and generate an SNMP trap reporting the problem. An SNMP trap will not be generated if the failing server is the last server and does not fail gracefully.

An administrator can repair the failed engine and bring it back into service with recovery similar to that of a soft failure. Or, if the problem is more complex, the administrator can reassign the filesets that belong to the server on the failed engine to other Metadata servers in the cluster and continue operations while the failed engine is repaired. Note that if the failed engine is the one on which the master server resides, the administrator must also set a new master server.

**Note:** An administrator can reassign the filesets that belong to a Metadata server only when that server is down, and can designate a new master server only if the current master server is down (the SAN File System cluster state is Offline). To ensure that an engine and its Metadata server are down, an administrator can issue a **stopengine** command. This command forcefully shuts down a designated engine.

**Related topics:**

- "Administrative server" on page 5
- "Components"
- "Engines" on page 11
- "Filesets" on page 11
- "Locks and leases" on page 16
- "Metadata server" on page 18

## Components

A *component* is one of the elements that make up the entire SAN File System. The following figure illustrates the major SAN File System software components.

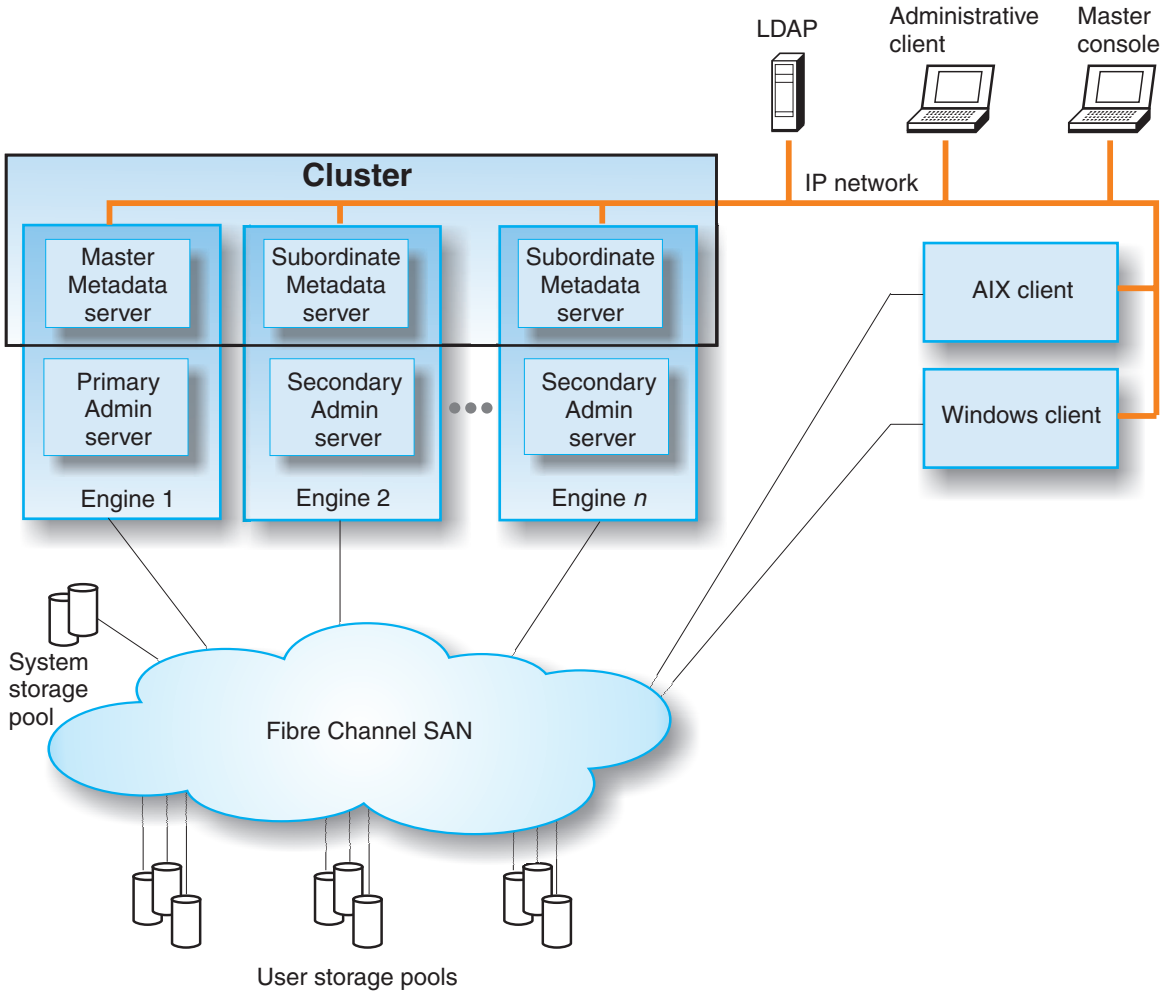



*Figure 1. SAN File System components*

The Metadata servers and clients communicate over an internal IP network and access data over the SAN.

**Note:** SAN File System relies on preexisting networking hardware, including an IP network, SAN, network switches, and routers.

A *Metadata server* is a server that runs on a SAN File System engine and performs metadata, administrative, and storage management services. Multiple Metadata servers form a server *cluster*. In a SAN File System server cluster, there is one master Metadata server and one or more subordinate Metadata servers, each running on a separate engine in the cluster. Together these Metadata servers provide clients with shared, coherent access to the SAN File System *global namespace*.

The *metadata* resides on private storage that is shared among all the Metadata servers in the cluster.

The *Administrative server* allows SAN File System to be remotely monitored and controlled through a Web-based user interface, called the *SAN File System console*, using either Netscape 6.2 and later or Internet Explorer 6.0 and later. The Administrative server also processes requests issued from the administrative command line interface, which is called *tanktool*. The Administrative server uses an

*LDAP server*, also connected to the internal IP network, to look up authentication and authorization information about the administrative users. The primary Administrative server runs on the same engine as the master Metadata server. It receives all requests issued by administrators and also communicates with the Administrative servers that run on each additional server in the cluster to perform routine requests.

The *clients* access the global namespace through a virtual or installable file system, which is installed on the client machine. These clients can act as servers to a broader clientele, providing Network File System (NFS) or Common Internet File System (CIFS) access to the global namespace or hosting applications (such as database servers or Web-hosting services that use multiple servers).

**Related topics:**

- "Administrative server" on page 5
- "Cluster" on page 6
- "Engines"
- "Metadata server" on page 18

## Engines

Within SAN File System, an *engine* is the hardware (which is based on the IBM xSeries® platform) on which a Metadata server and an Administrative server run. Each engine comes preloaded with SAN File System server software and must be configured by onsite IBM personnel. SAN File System supports from two to eight Model 1RX engines.

**Note:** Although you cannot purchase SAN File System with only one engine, you can run a single-engine system if all of the other engines fail (for example, if you have only two engines, and one of them fails), or if you want to bring down all of the engines except one before performing scheduled maintenance tasks.

The administrative infrastructure on each engine allows an administrator to monitor and control SAN File System from a standard Web browser or an administrative command line interface. The two major components of the infrastructure are an Administrative agent, which provides access to administrative operations, and a Web server that is bundled with the console services and servlets that render HTML for the administrative browsers. The infrastructure also includes a Service Location Protocol (SLP) daemon, which is used for administrative discovery of SAN File System resources by third-party Common Information Model (CIM) agents, and an IBM Director agent, which detects when the operating system "hangs" or stops unexpectedly and can reboot the operating system.

An administrator can use the SAN File System console, which is the Web-based user interface, or administrative commands to monitor and control an engine.

**Related topics:**

- "Metadata server" on page 18

## Filesets

A *fileset* (except for the global fileset) is a subset of the entire SAN File System global namespace. It serves as the unit of workload for Metadata servers and is also the unit that an administrator specifies to create FlashCopy images that are used in backup procedures.

**Creating and attaching filesets:**

A fileset has the following properties:

- A fileset name
- A directory path leading to the directory within which the fileset is attached
- A directory name that the fileset is given at the end of the directory path

The fileset attach point is the path formed by combining the directory path and the directory name.

The root of the global namespace is the global fileset. The name of the global fileset is always ROOT. The directory name of the global fileset is specified at SAN File System setup as sanfs.

The global fileset is attached to the root level in the namespace hierarchy. An administrator creates filesets and attaches them at specific locations below the global fileset. An administrator can attach a fileset to the global fileset (by specifying a directory path of sanfs). An administrator can also attach a fileset to another fileset (for example, by specifying a directory path of sanfs/fileset1, where sanfs/fileset1 is the attach point of the parent fileset). When a fileset is attached to another fileset, it is called a *nested fileset* or a child of a parent fileset.

When creating a fileset, an administrator can also specify a maximum size for the fileset (called a *quota*) and specify whether SAN File System should generate an alert if the size of the fileset reaches or exceeds a specified percentage of the maximum size (called a threshold).

**Note:** The space used by a fileset includes the space used by FlashCopy images. It does not include the space used by any filesets nested within it.

An administrator can detach a fileset and reattach it at the same location or at a different location. If a fileset is reattached at a different location, all the files contained in the fileset are rooted to the new location without any further operations. Note that before a fileset can be detached, any nested filesets must be detached first.

**Assigning filesets to Metadata servers:**

When creating a fileset, an administrator assigns it to a specific Metadata server in your SAN File System server cluster. That Metadata server is then responsible for providing metadata and locks to clients when they request access to files that reside in that fileset. The fileset-to-server assignment is automatically communicated between SAN File System clients and servers. The client transparently discovers which server to deal with when accessing files on a fileset.

An administrator should create and attach at least one fileset for each Metadata server in your server cluster. However, creating more filesets gives the administrator greater flexibility in distributing filesets among servers to maintain availability and to balance the workload.

**Note:** An administrator can assign a nested fileset to a different Metadata server than the one to which its parent fileset is assigned. However, if the server to which the parent fileset is assigned becomes inactive, the parent fileset and all of its nested filesets become unavailable until the server comes back up or an administrator reassigns the parent fileset to an active server.

**Creating objects in filesets:**

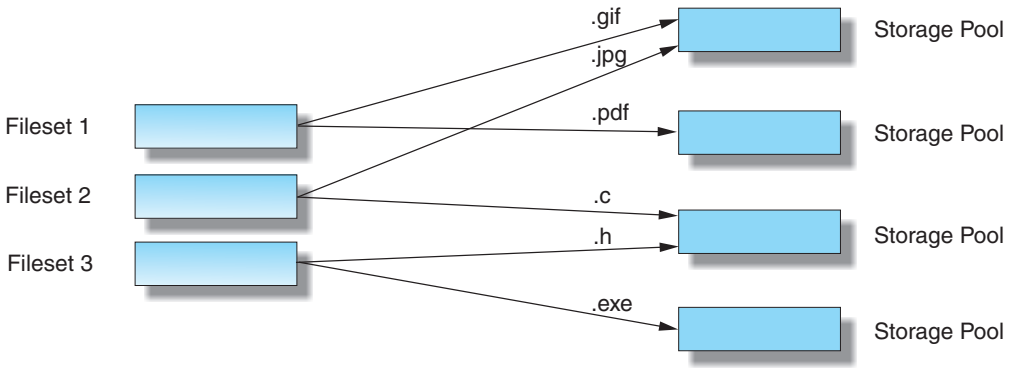
From a client perspective, a fileset appears to be a regular directory. Users and applications on SAN File System clients can create objects, such as directories and files, within the fileset.

A fileset must be attached to the global namespace and assigned to an active Metadata server before it is available for use by clients. If an administrator detaches an existing fileset, it becomes unavailable to clients until it is attached again. If the Metadata server to which the fileset is assigned becomes inactive, the fileset is unavailable to clients until the server comes back up or an administrator reassigns the fileset to another Metadata server.

Note that users cannot create hard links across fileset boundaries. In addition, a user cannot rename, move, or delete a directory that is the root of a fileset. If a user attempts to perform any of these operations, SAN File System issues an error message.

**Placing files in storage pools:**

A *storage pool* is a collection of SAN File System volumes that can be used to store either metadata or file data. A fileset can contain files that reside in more than one storage pool. A file is placed in a specific storage pool based on rules in a policy that are used for automatic file placement. There can also be files from more than one fileset in a storage pool. The following illustration shows an example of the relationship between filesets and storage pools.




Note that when creating file placement policies, an administrator can specify that all files created in a particular fileset are to be stored in a specific storage pool.

**Related topics:**

- "Alerts" on page 5
- "FlashCopy images"
- "Policies and rules" on page 24
- "Storage pools" on page 20

## FlashCopy images

A *FlashCopy® image* is a space-efficient, read-only copy of the contents of a fileset in a SAN File System global namespace at a particular point in time. A FlashCopy image can be used with standard backup tools available in your environment to create backup copies of files on tape.

**Creating FlashCopy images:**

When creating FlashCopy images, an administrator specifies each fileset to be included; the FlashCopy image feature does not automatically include nested filesets. The FlashCopy image operation is performed individually for each fileset. Also note that an administrator cannot create ″incremental″ FlashCopy images to be used as the basis for incremental backups. A FlashCopy image is simply an image of an entire fileset as it exists at a specific point in time. Note that while a FlashCopy image is being created, all data remains online and available to users and applications.

FlashCopy images are available to clients in a special subdirectory named .flashcopy under the root directory of a fileset as shown below.

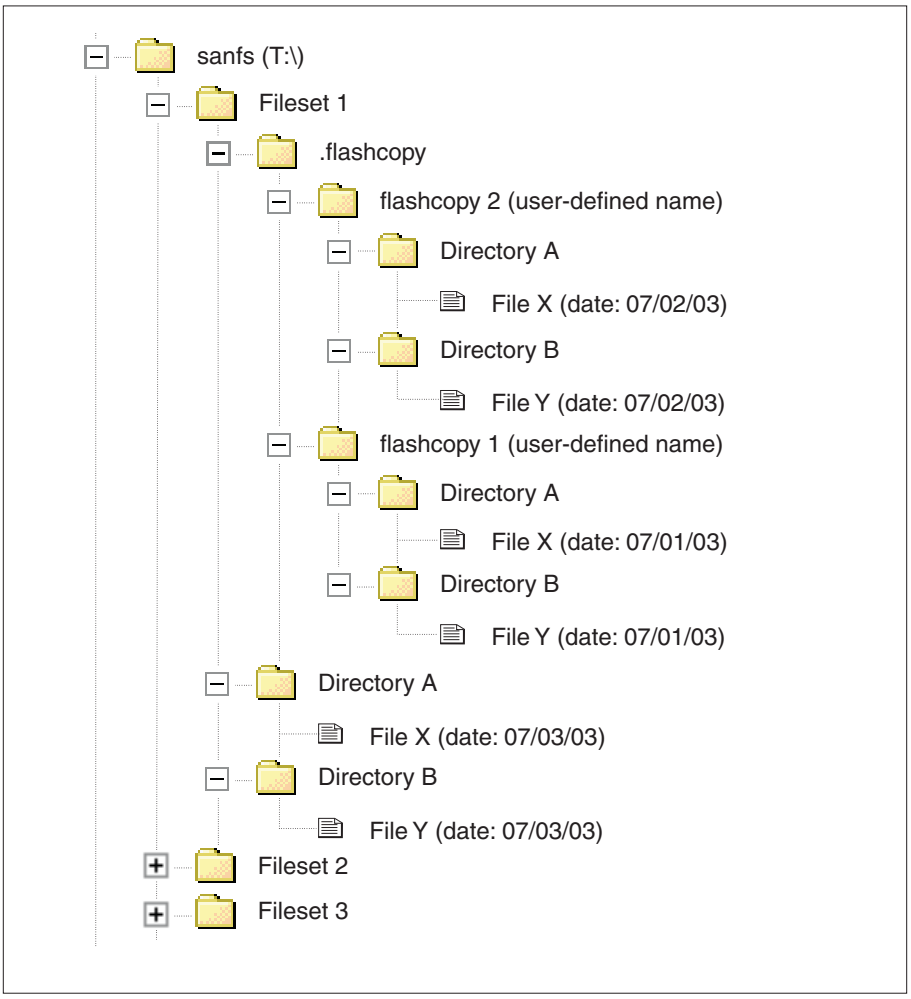



*Figure 2. FlashCopy images*

In this case, two FlashCopy images, flashcopy 2 and flashcopy 1, have been created under the special .flashcopy directory in Fileset 1. All directories (A, B) and files (X, Y) beneath flashcopy 1 and flashcopy 2 are read-only versions with all other attributes preserved from the original directory or file. Note that flashcopy 1 was created on 07/01/03, and flashcopy 2 was created on 07/02/03. Each reflects the contents of the files on those dates, while the original files have since been updated on 07/03/03.

A fileset can have as many as 32 read-only FlashCopy images. When creating a FlashCopy image for a fileset, the administrator can indicate whether the oldest image should be deleted if creating a new one causes the maximum number of images to be exceeded. Note that once a FlashCopy image is created, its name cannot be changed.

An administrator can list detailed information, such as the following, about each FlashCopy image:

- Its unique name
- Associated fileset name
- Associated fileset state (attached or detached)
- Directory name under which the FlashCopy image is stored
- Directory path for the FlashCopy image
- Description of the FlashCopy image
- Recency (range from 1 to 32)
- Size
- Description

**Note:** The actual files in a fileset and the FlashCopy images of the files in the fileset share the same file data blocks until a client makes changes to the files. When a client makes a change to a file, such as adding or deleting data, the client performs an operation called *copy on write*. It writes the changed blocks to a new location on disk. At this point, the FlashCopy image points to the old blocks, and the actual file points to the blocks with the new data. Therefore, any access to the FlashCopy image accesses the data blocks as they existed when the FlashCopy image was created, and any access to the actual file accesses the new data blocks.

**Backing up files using FlashCopy images:**

When performing a backup using standard backup tools available in your environment, a client can specify the path to the FlashCopy image instead of the path to the actual files and continue working with the files while the backup occurs. This procedure produces a consistent backup of the files in the fileset.

**Reverting files to a previous FlashCopy image:**

Although creating FlashCopy images is *not* a replacement for creating backups of your files to protect your data, in some cases, a user can choose to use a FlashCopy image to revert a file or a set of files to a specific point in time. For example, if a user accidentally deletes a file, restoring it by copying it from a FlashCopy image to another directory instead of restoring it from a backup copy can be faster.

When you restore files from a backup taken from a FlashCopy image, you cannot restore the files to the same location – all FlashCopy image directories are read-only directories. You must restore the files to the directory where the original files resided or to another directory.

**Note:** For ease of management, it is recommended that you create FlashCopy Images of all filesets at the same point in time, and use a common naming convention to indicate that they represent a set.

**Related topics:**

- "Filesets" on page 11

## Locks and leases

SAN File System uses locks and leases to ensure the consistency and integrity of data in the SAN File System global namespace. The internal locks discussed are in addition to the locks provided with the native file systems, such as flock().

A *lock* is a mechanism that restricts access to data and metadata. The SAN File System protocol provides locks that enable file sharing among SAN File System clients, and, when necessary, provides locks that allow clients to have exclusive access to files. It uses distributed data locks for cache consistency and file access locks to synchronize multiple, concurrent open instances of the same file. SAN File System supports locking semantics that correspond to open modes that are native to Windows and UNIX operating systems. It also supports byte-range locks.

When a client or server fails, SAN File System uses a lease-based safety protocol to ensure data consistency and to protect the structural integrity of the global namespace.

A client obtains a *lease* from a Metadata server as soon as it makes contact with that server. A lease is valid for the period of time set by an administrator using a Metadata server configuration parameter. When a client obtains a lock from a server, that lock is guaranteed to be valid by the server only as long as the client has a valid lease with the server. The server renews a client's lease each time the client contacts the server.

If a client does not contact the server within the specified lease period (for example, because of a temporary network partition), the server can revoke the client's locks. If other clients request locks on the same data, the server revokes the first client's locks and grants new locks to the other clients. If no such requests are made, when the client contacts the server again, it can renew its lease and reassert any locks (get its old locks back) that protect modified but uncommitted data in the client's cache, thus preventing data loss.

A client can also lose its lease because of a server failure. However, when the Metadata server is restarted, the client can renew its lease and reassert its locks. A Metadata server provides a grace period for lock reassertion to allow clients to reassert their locks before allowing other clients to obtain new ones. Clients cannot access any new data until the grace period has ended.

**Related topics:**

- "Metadata server" on page 18

## Logs

A *log* is file that contains a record of activity. SAN File System provides the following logs:

- Cluster log
- Event log
- Administrative log
- Audit log
- Security log
- Administrative agent message log

An administrator can view these logs from the SAN File System console or by using the **catlog** command, and can clear the cluster and audit logs using the **clearlog** command.

**Cluster log:**

The cluster log contains entries from all of the Metadata servers in a cluster. A separate log is maintained on each Metadata server. However, when an administrator issues a request to view the log, entries from all of the separate logs are merged and ordered by date and time to provide the administrator with a cluster-wide view of activities and events.

**Event log:**

The event log contains a subset of the entries in the cluster log. The entries that appear in the event log are those events, such as changes in server state, that have been configured as alerts. The event log is not stored in a separate physical file, but is generated from entries in the cluster log when a SAN File System administrator issues a request to view the log.

**Administrative log:**

The administrative log maintains a history of messages generated by the Administrative servers in a cluster. A separate log is maintained on each Administrative server. However, when an administrator issues a request to view the log, entries from all of the separate logs are merged into a single log.

**Audit log:**

The audit log is located on the master Metadata server engine, and merges with the existing file when the master Metadata server engine is switched. It contains entries for commands issued by administrators (from either the command line interface or the SAN File System console), ordered by date and time. The user ID of the administrator who issued a particular command is included in each entry.

**Note:** Generally, only commands sent to the Metadata server that change the metadata, cluster configuration, or are significant operations are recorded in the audit log. These commands can be sent using the GUI, CLI, or CIM. For example, list and stat functions are not logged because they do not change the system. Engine functions are not logged because they are not implemented in the metadata server, but making filesets, pools, and metadata checks are logged.

The exception is FlashCopy commands. These commands such as mk, rm, and revert, are logged here even though they do not change the SAN File System configuration. Queries made by a SAN File System administrator are not recorded.

SAN File System administrators can use information in the audit log to help them convert requests made from the SAN File System console into equivalent command line interface instructions or to perform troubleshooting in the case of a failure within the SAN File System system.

**Security log:**

The security log maintains a history of administrator login activity. A separate log is maintained on each Administrative server. However, when an administrator issues a request to view the log, entries from all of the separate logs are merged into a single log.

## Metadata server

A *Metadata server* is a server that runs on a SAN File System engine and performs metadata, administrative, and storage management services. In a SAN File System server cluster, there is one master Metadata server and one or more subordinate Metadata servers, each running on a separate engine in the cluster. Together these Metadata servers provide clients with shared, coherent access to the SAN File System global namespace.

All of the servers, including the master Metadata server, share the workload of the SAN File System global namespace. Each is responsible for providing metadata and locks to clients for specific filesets assigned to them by an administrator. They know which filesets belong to which server, and when contacted by a client can direct the client to the appropriate server. They manage distributed locks to ensure the integrity of all of the data within the global namespace.

In addition to providing metadata to clients and managing locks, Metadata servers perform a wide variety of other tasks. They process requests issued by administrators to create and manage filesets, storage pools, volumes, and policy sets, and they enforce the policies defined by administrators to place files in appropriate storage pools and ensure that capacity quotas established for filesets and storage pools are not exceeded.

**Performing Metadata service:**

There are two types of metadata:

- *File metadata*, which is information that clients need to access files directly from storage devices on your storage area network. File metadata includes permissions, owner and group, access time, creation time, and other file characteristics.
- *System metadata*, which is metadata used by the system itself. System metadata includes information about filesets, storage pools, volumes, and policies. It is stored and managed in a separate system storage pool that is only accessible by the Metadata servers in your server cluster. The Metadata servers perform the reads and writes required to create, distribute, and manage this information.

Distributing locks to clients involves the following:

- Issuing leases that determine the length of time that a server guarantees the locks that it grants to clients.
- Granting locks to clients that allow them shared or exclusive access to files or parts of files. These locks are semi-preemptible, which means that if a client does not contact the server within the lease period, the server can “steal” the client’s locks and grant them to other clients if requested; otherwise, the client can reassert its locks (get its locks back) when it can contact the server again.
- Providing a grace period during which a client can reassert its locks before other clients can obtain new locks if the server itself goes down and then comes back online.

**Performing administrative services:**

A Metadata server processes requests from administrators (issued from the SAN File System console or by using administrative commands) to perform the following types of tasks:

- Create and manage filesets, which are subsets of the entire global namespace and serve as the units of workload assigned to specific Metadata servers.
- Create and manage volumes, which are LUNs labeled for SAN File System's use in storage pools.
- Create and maintain storage pools (for example, an administrator can create a storage pool that consists of RAID or striped storage devices to meet reliability requirements, and can create a storage pool that consists of random-access or low-latency storage devices to meet high performance requirements).
- Create FlashCopy images of filesets in the global namespace that can be used to make file-based backups easier to perform.
- Define policy sets that contain rules that determine in which storage pools specific files are stored.

**Performing storage management services:**

A Metadata server performs these storage management services:

- Manages allocation of blocks of space for files on LUNs
- Maintains pointers to the bits of a file that constitute the file
- Evaluates the rules in the active policy set and manages the placement of files in specific storage pools based on those rules
- Issues alerts when filesets and storage pools reach or exceed their administrator-specified thresholds, or returns out-of-space messages if they run out of space

**Using configuration parameters:**

Each Metadata server has a configuration file that is stored on local storage for each engine in a Metadata server cluster. This configuration file contains settings, which are specified by an administrator, for the following:

**Space reclamation interval**
Controls how often the master Metadata server reclaims free storage pool partitions allocated by filesets.

**Lease period**
Specifies the maximum length of time that a server guarantees the validity of the locks it has granted to a client. The client must contact the server before the lease period ends to renew its lease and retain its locks.

**List of privileged clients**
Lists the clients on which users specified as root or administrator users can have those same privileges on the SAN File System global namespace.

**List of SNMP managers**
Specifies the IP address, port, version (SNMPv1 or SNMPv2c) and community strings for SNMP trap recipients.

**SNMP trap setting**
Specifies the filter used on the cluster message log to determine which messages (informational, warning, error, and severe) are also issued as SNMP trap messages.

**Related topics:**

- "Alerts" on page 5
- "Cluster" on page 6
- "Engines" on page 11
- "FlashCopy images" on page 13
- "Locks and leases" on page 16
- "Policies and rules" on page 24
- "Storage pools"
- "SNMP"
- "Storage management" on page 22
- "Components" on page 9
- "Volumes" on page 27

## SNMP

*Simple Network Management Protocol (SNMP)* is typically used to monitor the health and performance of software, hardware, and networks. SNMP consists of two main components:

- SNMP agents, which are software components that reside on managed devices and collect management information (using Management Information Bases or MIBs). SNMP agents issue traps when SNMP events occur. These traps are sent through User Datagram Protocol (UDP) to an SNMP Manager.
- An SNMP manager, which is a management application (client) that monitors and controls devices using SNMP protocol.

In SAN File System, the Metadata server generates SNMP traps in response to certain events. Note that no SNMP traps are issued from the operating system, hardware, or the Administrative agent.

**Note:** The RSA II cards can be set up to generate hardware traps as well.

SAN File System administrators can choose to configure which severity levels of events (informational, warning, error, or severe) generate SNMP traps and can specify the SNMP Managers that receive the traps. When an event occurs with a severity level that causes an SNMP trap, SAN File System sends the trap and logs the event in the cluster log.

**Note:** SAN File System supports asynchronous monitoring through traps but does not support SNMP GETs or PUTs for active management. The SNMP Manager cannot manage SAN File System.

Not all events in SAN File System generate traps. Examples of events that might generate SNMP trap messages include the following:

- When a server executes a change in state
- When a server detects that another server is not active
- When the size of a fileset reaches a specified percentage of its capacity

**Related topics:**

- "Alerts" on page 5

## Storage pools

A *storage pool* is a collection of SAN File System volumes that can be used to store either metadata or file data. A storage pool typically contains a set of volumes that

provide a desired quality of service for a specific use, such as to store all files for a particular application or a specific business division. Note that an administrator must assign one or more volumes to a storage pool before it can be used.

**Understanding storage pool types:**

SAN File System includes the following three types of storage pools:

**System storage pool**
The system storage pool contains the system metadata (system and file attributes, configuration information, and Metadata server state) that is accessible to all Metadata servers in your server cluster. There is only *one* system storage pool.

The system storage pool is created when SAN File System is installed. However, an administrator must assign one or more volumes to this storage pool before it can be used.

**Note:** The first volume assigned to the system storage pool is called the *master volume*. This volume contains the master dbspace within SAN File System, which contains the most critical pages of metadata that SAN File System manages. Internal to the server, the dbspace is organized into various structures that are used by the subsystems within the Metadata servers. It is important to use highly reliable and available LUNs as volumes within the system storage pool so that the server always has a robust copy of this and other system metadata available at all times.

Because the amount of metadata grows as the global namespace grows, you must monitor the system storage pool to ensure that there is always enough volumes assigned to it to accommodate the growth. The system storage pool typically requires approximately 2% to 5% of the total storage capacity that SAN File System manages, but this amount varies depending on your environment. Use the alert features on the system storage pool to ensure that you do not run out of space.

**Note:** The minimum size of a system volume is 2 GB; therefore, the minimum size of the system storage pool is also 2 GB.

For security and reliability, the volumes assigned to the system storage pool should be accessible only to the server cluster by using a private SAN or a shared SAN with a combination of zoning, LUN masking, or special configuration. For reliability, the volumes should be virtualized RAID arrays (also known as ranks within IBM Enterprise Storage Server®).

**User storage pools**
User storage pools contain the blocks of data that make up user files. SAN File System stores the data that describes the files (which is called file metadata) separately from the actual file data.

An administrator can create one or more user storage pools, and then create policy sets that contain rules that cause Metadata servers to store data for specific files in the appropriate storage pools.

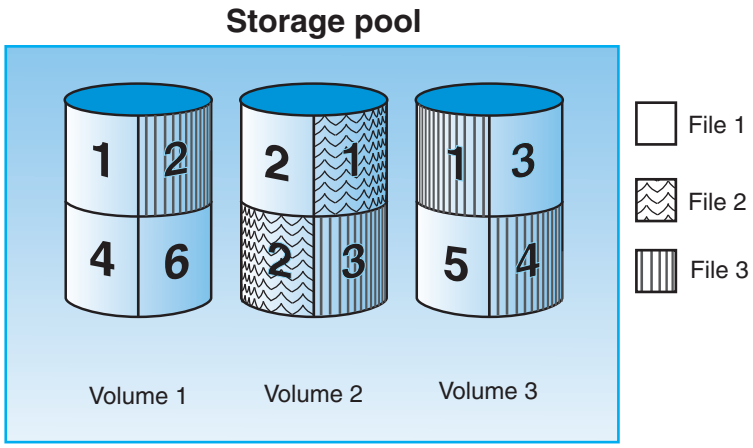
**Default storage pool**
The default storage pool is the user storage pool where a Metadata server stores the data for a file if the file is not assigned to a specific storage pool by a rule in the active policy set.

A default storage pool is created when SAN File System is installed. However, an administrator must assign one or more volumes to it before it can be used. Note that while there can be only one default storage pool at a time, an administrator can designate any user storage pool that has volumes assigned to it to be the default storage pool.

**Assigning volumes to storage pools:**

Typically, an administrator assigns volumes to storage pools based on their common characteristics, such as device capabilities (availability or performance level) and usage (business division, project, application, location, or customer).

Each storage pool manages its own volumes. Space is allocated to the volumes in each storage pool in a round-robin algorithm (see the following figure), in logical partitions, or in blocks. Logical partitions are allocated to the system storage pool in 16-MB blocks. For user storage pools, including the default storage pool, an administrator can choose to allocate logical partitions in 16, 64, or 256-MB blocks. All logical partitions in the same storage pool must be the same size.




**Note:** An administrator can choose to have an alert generated whenever a storage pool reaches or exceeds a certain percentage of its maximum capacity. By default, an alert is generated when a storage pool becomes 80% full. An alert is logged every five minutes until one or more volumes are assigned to the storage pool. Note that an administrator can set configuration parameters to cause an SNMP trap message to be generated as well. An SNMP trap notifies an administrator of this condition asynchronously.

**Related topics:**

- "Alerts" on page 5
- "File placement" on page 23
- "Filesets" on page 11
- "Policies and rules" on page 24
- "Storage management"
- "Volumes" on page 27

## Storage management

SAN File System provides automatic file placement through the use of *policies* and *storage pools*. An administrator can create quality of service storage pools that are available to all users, and define *rules* in policies that cause newly created files to

be placed in the appropriate storage pools automatically. For more information about policies, rules, and storage pools, see the related topics below.

**Related topics:**

- "File placement"
- "FlashCopy images" on page 13
- "Policies and rules" on page 24
- "Storage pools" on page 20

## File placement

SAN File System provides automatic *file placement* through the use of policies and storage pools.

A *policy* is a list of rules that determines where the data for specific files is stored.

A *rule* is an SQL-like statement that tells a SAN File System Metadata server to place the data for a file in a specific storage pool if the file attribute that the rule specifies meets a particular condition. A rule can apply to any file being created or to only files being created within a specific list of filesets and any filesets nested within the specified filesets.

A *storage pool* is a named set of storage volumes that can be specified as the destination for files in rules. If a storage pool is the destination for user files, it is called a *user storage pool* (in contrast to a *system storage pool*, which is used to store SAN File System system metadata).

The rules in a policy are processed in order until the condition in one of the rules is met. The data for the file is then stored in the specified storage pool. If none of the conditions specified in the rules of the policy is met, the data for the file is stored in the default storage pool.

**Notes:**

1. Rules in a policy are evaluated only when a file is being created. If an administrator switches from one policy to another, the rules in the new policy apply only to newly created files. Activating a new policy does not change the storage pool assignments for existing files. And moving a file does not cause a policy to be applied. Note that while an administrator can create multiple policies, only one policy can be active at a time.
2. After a file has been created, you can check its storage pool assignment. However, the only way to do so is to list files on volumes in the target storage pool from the Administrative command-line interface using the **reportvolfiles** command.
3. If you base your policies on userids, be aware of how the UNIX **untar** command restores files from backup. During the restore, a file is first created from backup with the userid of the performer of the backup and is then changed to the userid of the original creator of the file. With SAN File System, since the policy is applied to the file at creation, the policy applies to the userid of the performer of the backup rather than the userid of the original file creator.

**Related topics:**

- "Policies and rules" on page 24
- "Storage pools" on page 20

## Policies and rules

Policies and the rules that they contain are used to assign files to specific storage pools. A storage pool typically contains a set of volumes that provide a specific quality of service for a specific use, such as to store all files for a particular application or a specific business division.

**Policies:**

A *policy* contains a list of rules that determines where specific files are placed. An administrator can define any number of policies, but only one policy can be active at a time. If an administrator switches from one policy to another (defines a different policy to be the active policy), that action has no effect on files that are already assigned to specific storage pools. Rules in the newly activated policy are evaluated only for files created after the policy is activated.

A policy can contain any number of rules; however, the entire policy cannot exceed a length of 32–KB (which includes any spaces used to delimit the rules).

SAN File System performs error checking for policies in the following phases:

- When an administrator creates a new policy, the master Metadata server checks the basic syntax of all the rules in the policy.
- When an administrator activates the policy, the master Metadata server checks all references to filesets and storage pools. If a rule in the policy refers to a fileset that does not exist or is not attached or to a storage pool that does not exist, an error is returned, and the policy is not activated.
- When a new file is created, the rules in the policy are evaluated in order. If an error is detected, the Metadata server responsible for creating the file logs an error, skips all subsequent rules, and assigns the file to the default storage pool.

**Note:** When SAN File System is first installed, a null policy is created and remains active until an administrator creates and activates a new one. The null policy assigns all files to the default storage pool. Note that a default storage pool is created when SAN File System is first started; however, an administrator must assign volumes to it before it can be used. If users attempt to create new files that would be assigned to the default storage pool, and there are no volumes assigned to it, users will receive No Space errors.

**Rules:**

A *rule* is an SQL-like statement that tells a SAN File System Metadata server to place the data for a file in a specific storage pool if the file meets a particular condition. A rule can apply to any file being created or only to files being created within a specific list of filesets and any filesets nested within them.

SAN File System evaluates rules in the order that they appear in the active policy. When a client creates a file, SAN File System scans the list of rules in the active policy to determine which rule applies to the file. When a rule applies to the file, SAN File System stops processing the rules and assigns the file to the appropriate storage pool. If no rule applies, the file is assigned to the default storage pool.

**Notes:**

1. Rules in a policy are evaluated only when a file is being created. If an administrator switches from one policy to another, the rules in the new policy apply only to newly created files. Activating a new policy does not change the

storage pool assignments for existing files. And moving a file does not cause a policy to be applied. Note that while an administrator can create multiple policies, only one policy can be active at a time.

2. After a file has been created, you can check its storage pool assignment. However, the only way to do so is to list files on volumes in the target storage pool from the Administrative command-line interface using the **reportvolfiles** command.
3. If you base your policies on userids, be aware of how the UNIX **untar** command restores files from backup. During the restore, a file is first created from backup with the userid of the performer of the backup and is then changed to the userid of the original creator of the file. With SAN File System, since the policy is applied to the file at creation, the policy applies to the userid of the performer of the backup rather than the userid of the original file creator.

**Note:**

For detailed information about creating policies and rules, see the related topics below.

**Related topics:**

- "File placement" on page 23
- "Filesets" on page 11
- "Storage pools" on page 20

## User interfaces

SAN File System provides the following user interfaces:

- A Web-based administrative user interface called the SAN File System console
- An administrative command line interface called tanktool
- A client command line interface

**SAN File System console:**

The SAN File System console allows an administrator to control and monitor SAN File System from a Web-based graphical user interface. For ease of monitoring, it provides a system overview that illustrates the status of the various SAN File System components. In addition, the SAN File System console provides in-line messaging that assists with system configuration, performance tuning and troubleshooting tasks.

The SAN File System console also contains the Help Assistant, which provides panel-level help information as well as links to related topics in the SAN File System Information Center. The Information Center serves as an online, searchable repository for all of the product documentation.

The SAN File System console includes the following main features:

**Task bar**

The task bar has both a tool segment and a task segment. The tool segment provides buttons for global user actions, such as closing the banner, accessing user assistance, and signing off from the system. The task segment allows you to toggle the navigation frame, titled **My Work**, on and off. The task bar also keeps track of all opened primary (or "main")

tasks, and allows you to quickly jump back and forth between them. Primary tasks are those tasks listed in the **My Work** frame, and serve as starting points for related sub tasks.

**My Work frame**
Contains primary task-based links that open panels in the work area.

**Work area**
Displays all user input and product content through panels. The panels contain static information and user interface elements, such as text fields and tables, that you can manipulate to affect SAN File System settings and behavior.

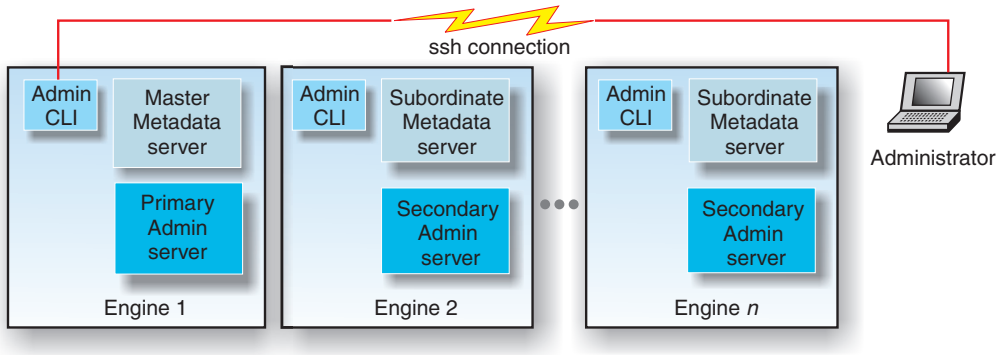
**Tanktool:**

Tanktool is an administrative command line interface (ACLI). An administrator can use this interface to administer all aspects of SAN File System, including setting up and managing storage pools, volumes, and filesets. For security reasons, tanktool runs only on the engines in your SAN File System server cluster. An administrator can use tanktool interactively and can embed tanktool commands in scripts.

To access tanktool, an administrator must log in to any engine that hosts a Metadata server.

*Figure 3. Accessing tanktool*

The following figure illustrates how the administrator accesses tanktool.




**Client command line interface:**

The client command line interface provides a set of commands used to set up a SAN File System client and to perform planning, migration, and verification tasks for data. This interface runs on all machines on which the SAN File System client code is installed.

To access client commands, a user logs in to the client machine.

**Related topics:**

- "Administrative server" on page 5

## User roles

SAN File System provides different levels of user access that you can assign to administrative tasks in your environment. These access levels, or *user roles*, are one

way to provide added security. The SAN File System user roles are described in the following table:

*Table 2. SAN File System user roles*

| Role | Level | Description |
|---|---|---|
| Monitor | Basic level of access | Allows the user to obtain basic status information about the cluster, display the cluster message log, display the rules in a policy set, and list information regarding SAN File System elements such as storage pools, volumes, and filesets. |
| Backup | Monitor + backup access | Allows the user to use all commands available to the Monitor role. In addition, it provides the user with the ability to perform backup and recovery tasks for metadata. |
| Operator | Backup + additional access | Provides use of all commands available to the Backup and Monitor roles. Also allows the user to perform day-to-day operations and tasks requiring frequent modifications. |
| Administrator | Full access | Provides access to all administrative commands except for the IBM Support-only commands for initial cluster setup and configuration tasks. |

**Note:** Each less-restrictive role includes the privileges and abilities of the more-restrictive roles above it.

## Volumes

A *volume* is a logical unit number (LUN) labeled by SAN File System for its use. A LUN is the logical unit of storage that a SAN or other disk subsystem can assign to SAN File System Metadata servers and clients.

A LUN becomes a SAN File System volume when an administrator adds it to a storage pool. It is automatically assigned a system-generated label that identifies it as a SAN File System volume. An administrator must also give the volume a name that is unique within all the volumes used by a SAN File System cluster.

At every startup, a Metadata server scans all of the LUNs that it can access, searching for labels that indicate which LUNs are valid SAN File System volumes. Clients perform this same search whenever they are started.

**Adding volumes to storage pools:**

When you install SAN File System, there is a system storage pool, which is used by Metadata servers to store system and file metadata, and a default storage pool, which can be used to store file data. You can create additional user storage pools for file data. However, no data can be stored in a storage pool until an administrator assigns one or more volumes to it.

The volumes added to the system storage pool are called *system volumes*.

As the amount of metadata that is generated for the server cluster and client files grows, an administrator must ensure that the system storage pool always has enough volumes assigned to it so that it does not run out of space.

An administrator must also ensure that the default storage pool and any other user storage pools the administrator creates are assigned a sufficient number of volumes. Each storage pool must have at least one volume assigned to it before any files can be stored in it.

To assist an administrator in monitoring storage pool capacity, SAN File System provides a threshold option that an administrator can specify when adding a volume to a storage pool or changing settings for a storage pool. A threshold is a specified percentage of the estimated maximum capacity of the storage pool. When a storage pool reaches or exceeds the percentage specified as its threshold, SAN File System generates an alert. Note that this alert can also generate an SNMP trap message to notify an administrator of the condition asynchronously if the administrator sets the appropriate parameters for SNMP traps.

**Activating volumes:**

When an administrator adds a volume to a storage pool, the volume is activated by default. This means that a Metadata server can allocate data to the newly added volume. Note that an administrator can also choose to add a volume to a storage pool in a suspended state. However, no data can be allocated to the volume until the administrator activates it.

**Suspending volumes:**

A volume can be in a suspended state if an administrator chooses to add it to a storage pool without activating it. An administrator can also change the state of a volume from activated to suspended. When a volume is in a suspended state, a Metadata server cannot allocate any data to it.

**Removing volumes:**

An administrator can also remove a volume from a storage pool. During this process, any files stored in the volume are automatically redistributed among the remaining volumes in the same storage pool. When an administrator removes a volume, data is moved and committed one logical partition at a time. If a failure occurs while moving the contents of a volume, the administrator can reissue the command, and the move process starts where it left off.

If I/O errors occur for some files or if the redistribution of the files to other volumes fails, an administrator can specify a force option that causes the volume to be removed anyway. If there is a bad file on the volume, specifying this option causes the system to delete the entire file, even if parts of the file reside on other volumes.

If the bad file is part of a FlashCopy image, SAN File System removes it from the FlashCopy image. This removal might render the FlashCopy image non-revertable. You can still backup and restore the remaining files on the FlashCopy image.

**Note:** There is no automatic recovery process when an administrator specifies the force option. All files are removed immediately without being copied. Before removing a volume with the force option, use the **reportvolfiles** command

to display a list of files on the volume. The files for which failures occur are also listed in the cluster message log, and an administrator can restore those files manually.

When an administrator removes a volume from a storage pool, its label is removed, and the volume becomes a LUN again.

**Related topics:**

- "Alerts" on page 5
- "Storage pools" on page 20

## Administrative agent for SAN File System

The CIM Agent for SAN File System, known as the Administrative agent, provides an application programming interface for the operations that an administrator performs to manage a cluster. It offers CIM-compatible objects for managing SAN File System.

**Note:** The SAN File System CIM model is following the direction of the Distributed Management Task Force Inc. (DMTF) industry standard as it develops. As the standard develops, the SAN File System model should improve in its consistency and its inheritance from CIM base classes. Also, the SAN File System model does not currently include associations and indications.

Service Location Protocol (SLP) is a mechanism for publishing and locating the Administrative agent. It enables third-party clients to discover and connect to the Administrative agent. SAN File System provides a default SLP configuration. If you want to modify the default configuration, see the www.openslp.org web site for information about SLP settings.

A standard CIM client can use the Administrative agent to access and control the Metadata server. The Administrative agent restricts access to administrative operations through user roles that are stored in the Lightweight Directory Access Protocol (LDAP) server. Before using a CIM client, make sure the LDAP server contains your username and password. See the *SAN File System Planning, Installation and Configuration Guide* for information about configuring LDAP.

**Related topics:**

- "STC_MasterDisruptiveSetting" on page 106
- "STC_TankDisruptiveSetting" on page 144
- "User roles" on page 26

## Functional view of the Administrative agent

This section provides functional views of the Administrative agent object model. Diagrams show specific functionality that is provided by the Administrative agent and illustrate the architecture of the Administrative agent.

**Related topics:**

- "CIM base classes" on page 30
- "SAN File System component classes" on page 32
- "SAN File System configuration classes" on page 40
- "SAN File System status classes" on page 42

## CIM base classes

The following diagram shows the CIM base classes which are the superclasses of the SAN File System classes.

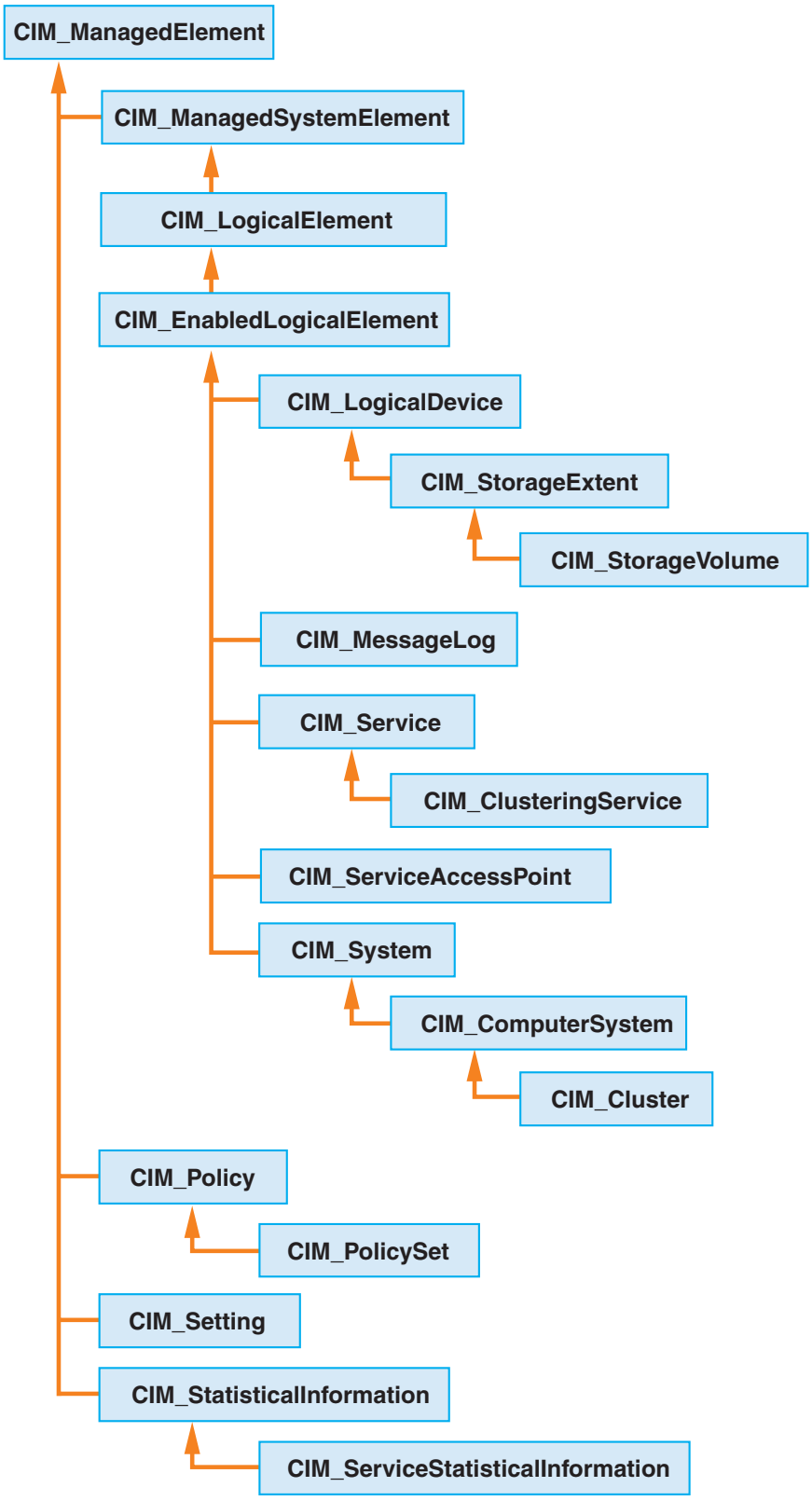



*Figure 4. CIM base classes*

**Related topics:**

- "CIM concepts" on page 1

## SAN File System component classes

Table 3 provides an overview of the classes that represent the major elements of SAN File System.

*Table 3. SAN File System element classes*

| Name | Description |
|---|---|
| STC_AvailableLUNs | This class represents an available Fibre Channel (FC) LUN. (When you assign a LUN to a storage pool, SAN File System labels it as a volume.) This class provides information about the channel and the LUN size and state. |
| STC_Cluster | This class, along with the STC_MasterService class, represents a cluster. It provides the identifier of the cluster and the number of engines. |
| STC_ComputerSystem | This class represents each engine in the cluster. It provides the identifier and state of the engine. Its method enables you to set the power state of the engine. |
| STC_Container | This class represents a fileset (also known as a container). It provides the identifier, its location, size, FlashCopy images and server of the fileset. Its methods enable you to define a new fileset, attach, detach, delete or move an existing fileset or change its hosting server. |
| STC_MasterSAP | This class represents the master Metadata server service access point. It extends the STC_TankSAP class and indicates whether the local server is the master Metadata server. |
| STC_MasterService | This class, along with the STC_Cluster class, represents a cluster and provides cluster services. It identifies the service and the state of the cluster. Its methods enable you to bring up, bring down, quiesce or resume all servers in a cluster, start or stop a check of metadata, or commit the cluster to start using the latest software. |
| STC_PolicySet | This class represents a policy, which is a list of file-placement and service-class rules that define characteristics and placement of files. It provides information about the policy and its policy rules. Its methods enable you to create, delete, activate, and get the rules for a policy. |
| STC_RegisteredFSClients | This class represents a registered client of a Metadata server. It provides identifier, location, and lease information about the client. |
| "STC_RemoteServiceAccessPoint" on page 136 | This class represents a remote service access point. It provides information that you can use to access the SAN File System console. |
| STC_StoragePool | This class represents a storage pool. It provides identifier, type, and size information about the storage pool. Its methods enable you to create, delete, move and set the type of the storage pool. |
| STC_TankSAP | This class represents a Metadata server service access point. It provides the addresses of the Ethernet server and the configuration of the ports. |

*Table 3. SAN File System element classes (continued)*

| Name | Description |
|---|---|
| STC_TankService | This class represents a Metadata server and provides server services. It provides the identifier and state of the server, whether it is a master Metadata server, and the number of filesets that it serves. Its methods enable you to start, stop and make a server the master Metadata server. |
| STC_Volume | This class represents a volume. It provides identifier, location, state and size information about the volume. Its methods enable you to create, delete, move a volume and suspend and resume partition allocation of a volume, reset the iterator that locates each file entry on the volume, and return the next file entry. |

**Available LUNs class:**

The following diagram shows the hierarchy and definition of the STC_AvailableLUNs class.

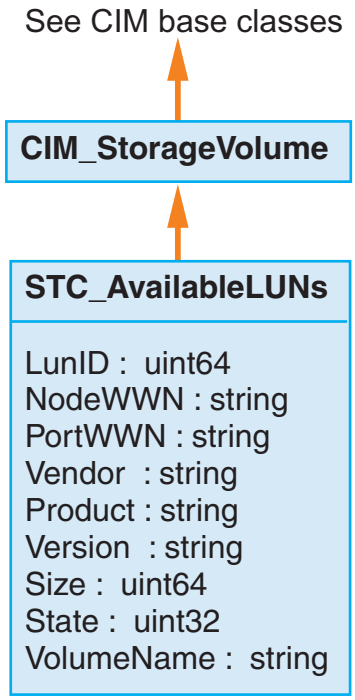



*Figure 5. Available LUNs class*

**Computer system classes:**

The following diagram shows the hierarchy and definitions of the STC_Cluster and STC_Computer classes.

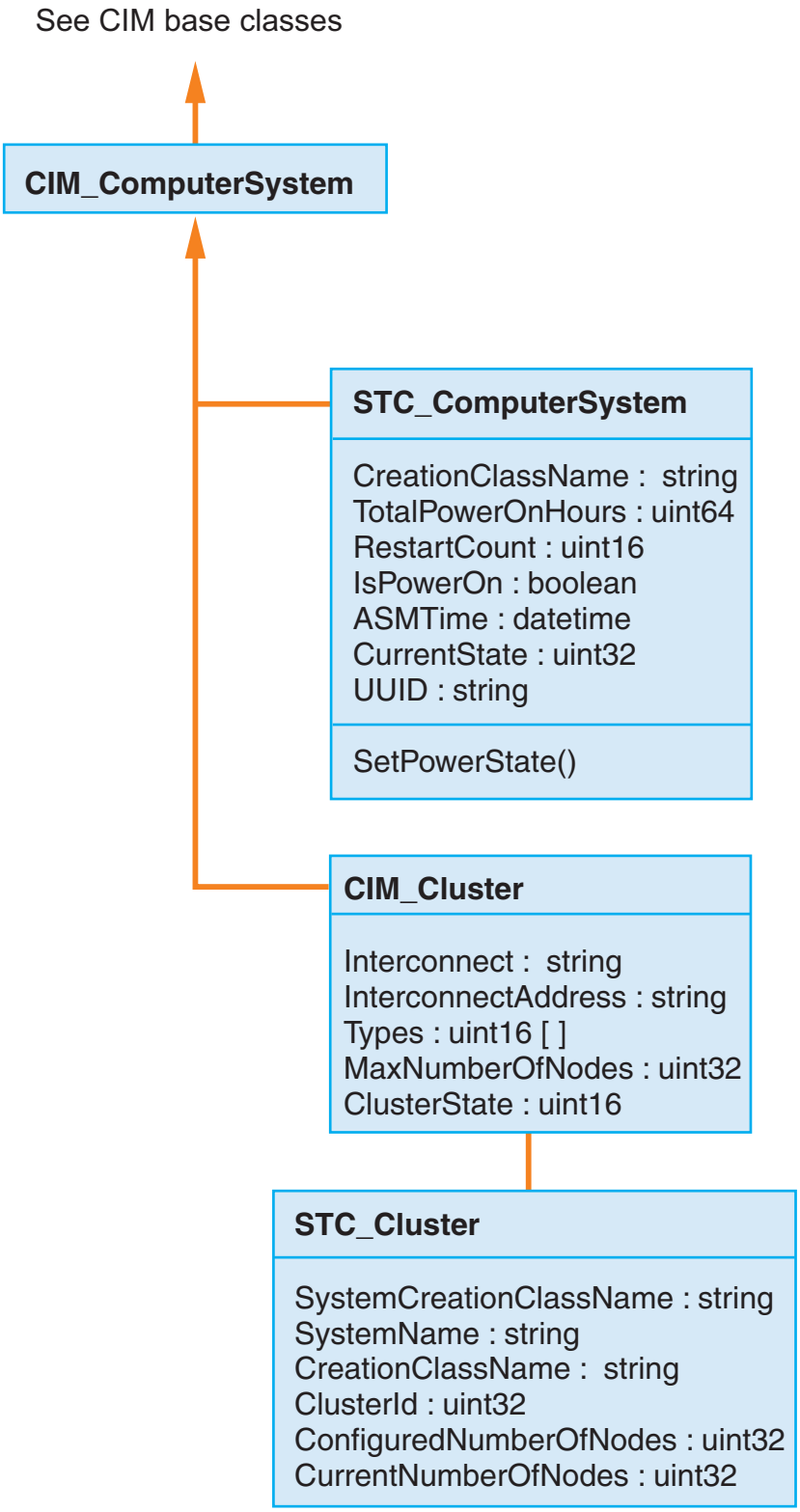



*Figure 6. Computer system classes*

**System element classes:**

The following diagram shows the hierarchy and definitions of the STC_Container, STC_StoragePool, and STC_Volume classes.

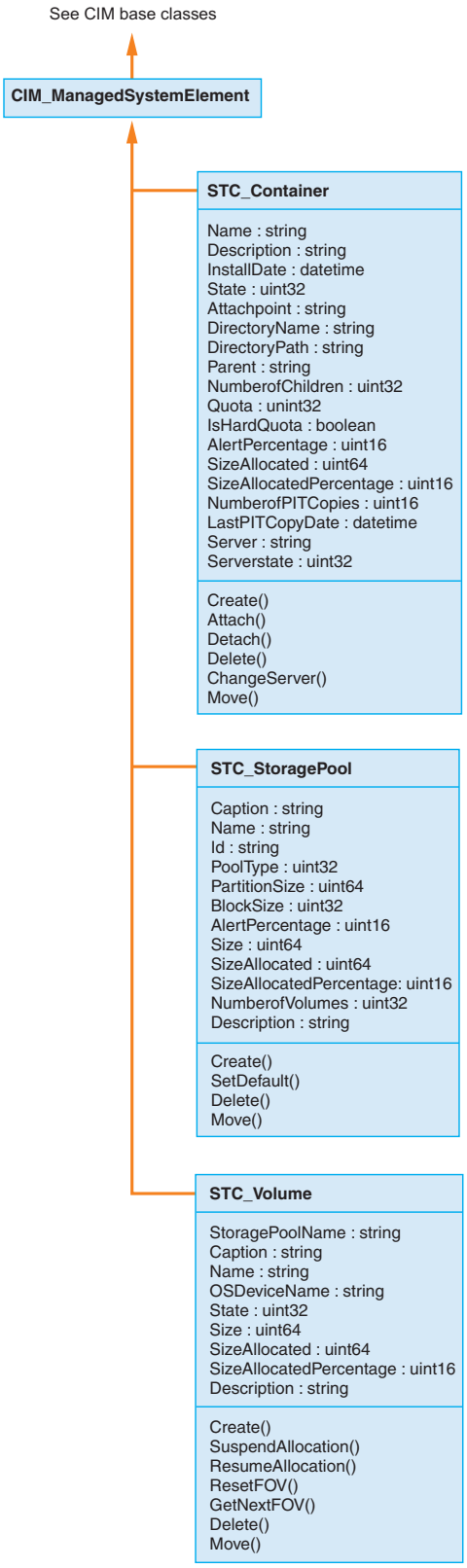



*Figure 7. System element classes*

**Service access point classes:**

The following diagram shows the hierarchy and definitions of the STC_RemoteServiceAccessPoint, STC_MasterSAP, and STC_TankSAP classes.

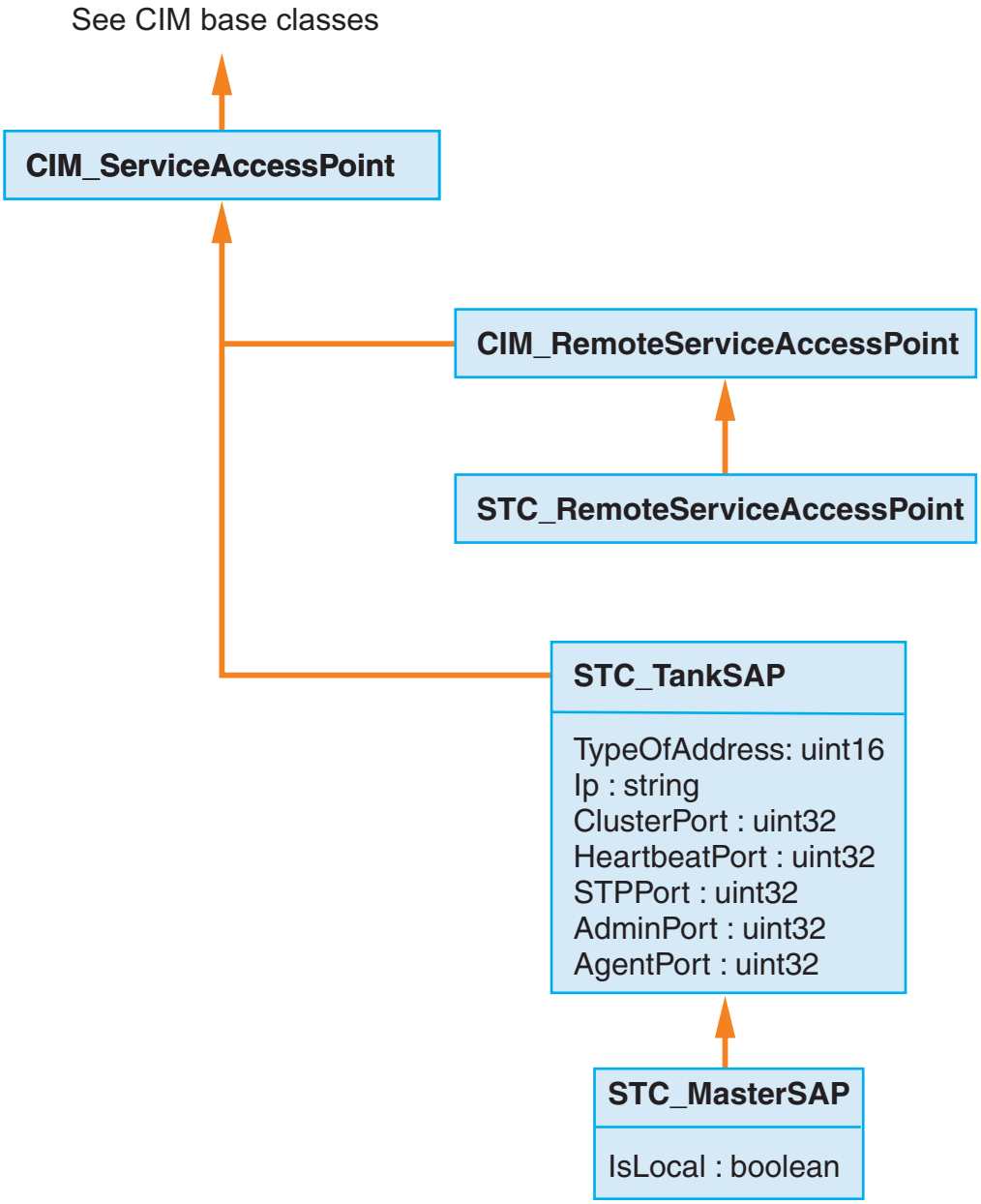



*Figure 8. Service access point classes*

**Service classes:**

The following diagram shows the hierarchy and definitions of the STC_MasterService and STC_TankService classes.

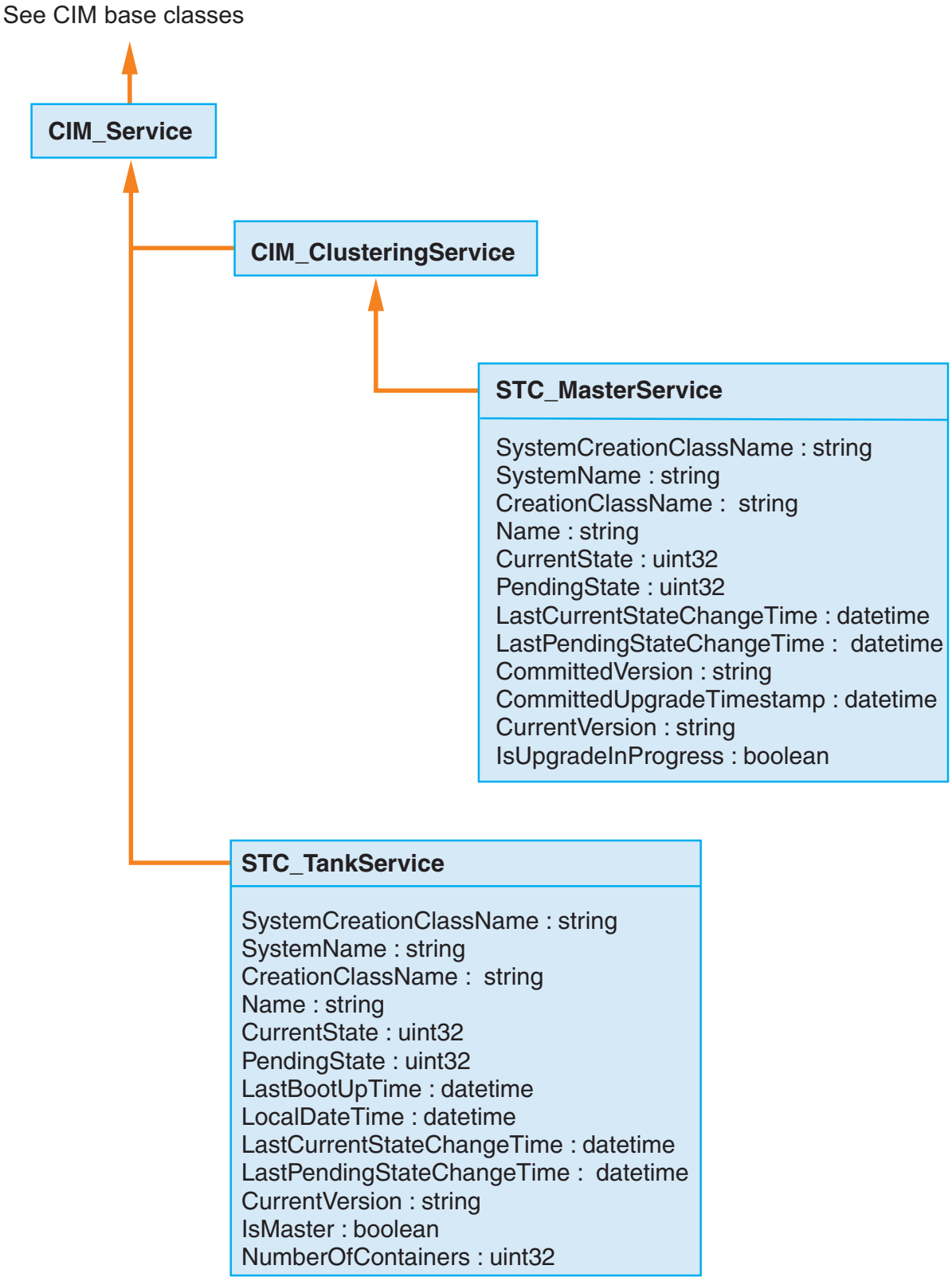



*Figure 9. Service classes*

**Policy class:**

The following diagram shows the hierarchy and definition of the STC_PolicySet class.

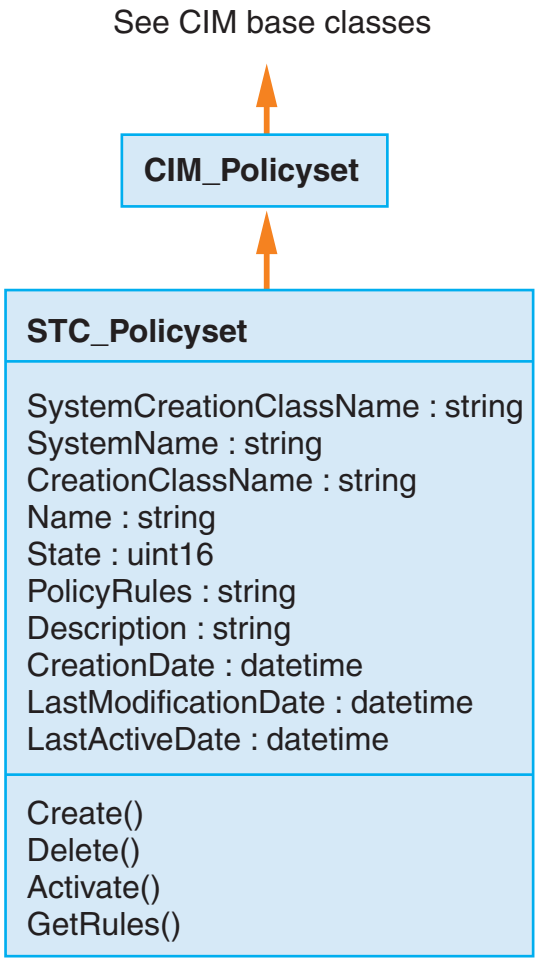



*Figure 10. Policy class*

**Metadata server clients class:**

The following diagram shows the hierarchy and definition of the STC_RegisteredFSClients class.

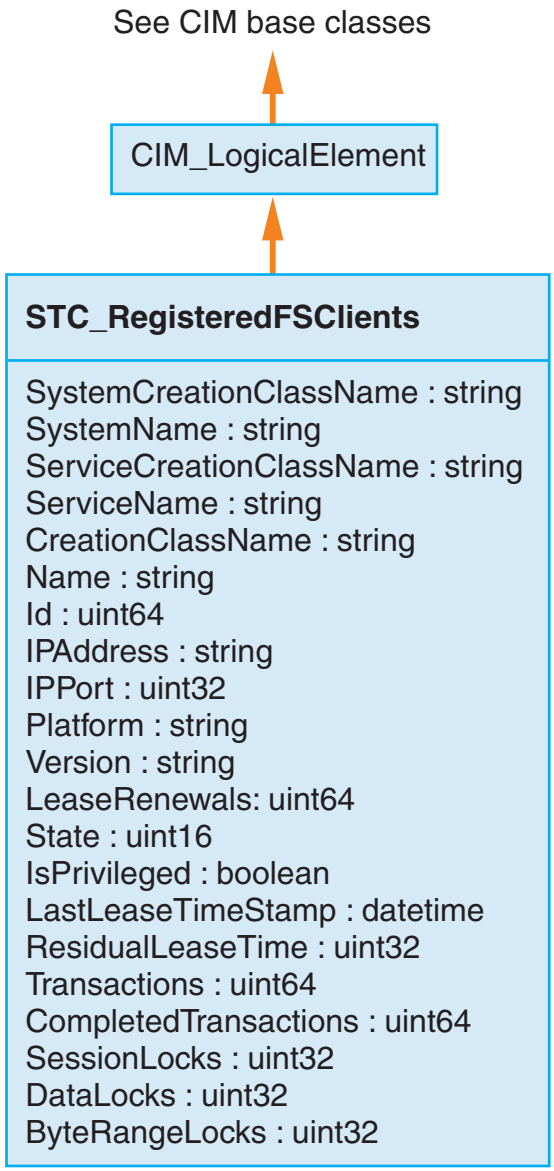



*Figure 11. Metadata server clients class*

**Related topics:**


- "STC_AvailableLUNs" on page 95
- "STC_Cluster" on page 96
- "STC_ComputerSystem" on page 96
- "STC_Container" on page 98
- "STC_MasterSAP" on page 109
- "STC_MasterService" on page 110
- "STC_PolicySet" on page 131
- "STC_RegisteredFSClients" on page 135
- "STC_StoragePool" on page 137
- "STC_TankSAP" on page 148
- "STC_TankService" on page 149
- "STC_Volume" on page 156

## SAN File System configuration classes

Table 4 provides an overview of the classes that represent configuration parameters.

*Table 4. SAN File System configuration parameter classes*

| Name | Description |
|---|---|
| STC_MasterDisruptiveSetting | This class represents cluster configuration parameters that require a cluster restart for an update to take effect. |
| STC_MasterDynamicSetting | This class represents cluster configuration parameters that you can dynamically update, without a cluster restart. |
| STC_TankDisruptiveSetting | This class represents settings for server-specific, configuration parameters that need a Metadata server restart for an update to take effect. |
| STC_TankTransientSetting | This class represents server-specific configuration parameters that are effective only until the next restart. |

The following diagram shows the hierarchy and definitions of the configuration classes.

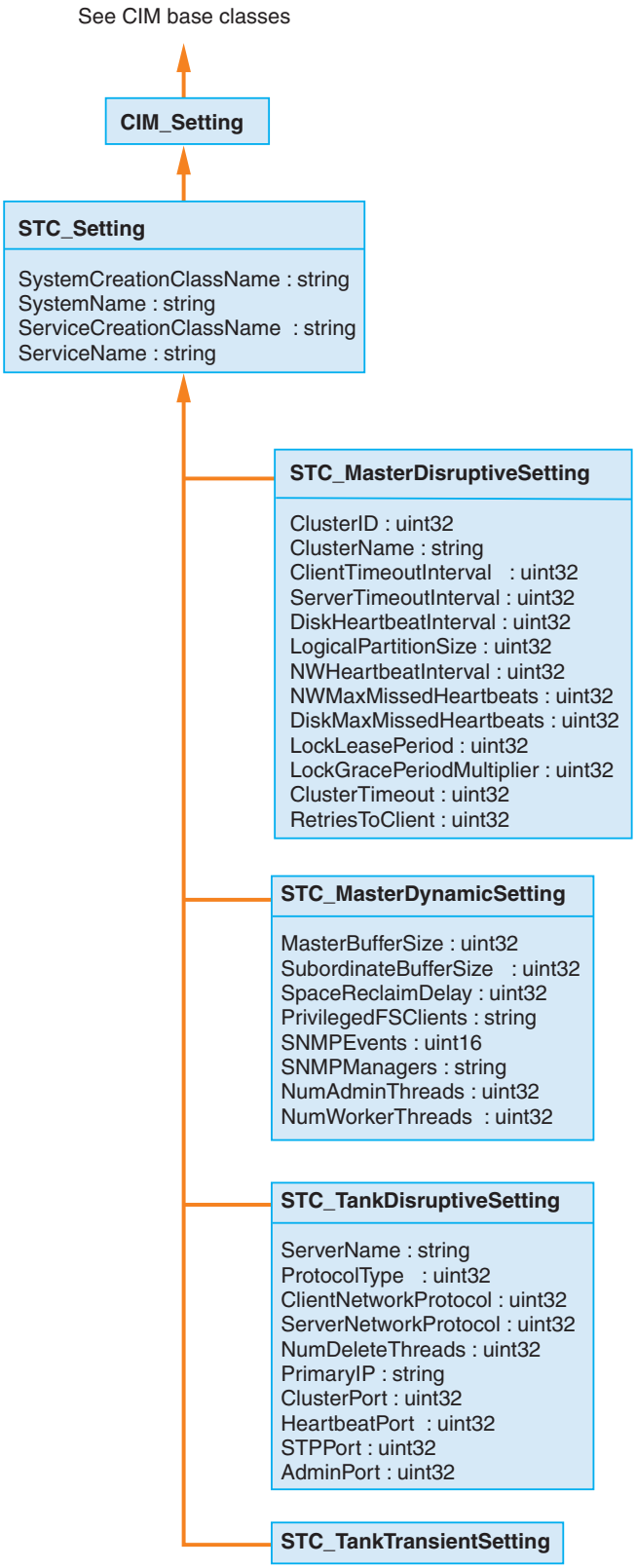



*Figure 12. Configuration classes*

**Related topics:**

- "STC_MasterDisruptiveSetting" on page 106

- "STC_MasterDynamicSetting" on page 107
- "STC_TankDisruptiveSetting" on page 144
- "STC_TankTransientSetting" on page 152

## SAN File System status classes

Table 5 provides an overview of the classes that represent the status of SAN File System components.

*Table 5. SAN File System status classes*

| Name | Description |
|---|---|
| STC_AdminProcess | This class represents a long-running administrative process in the cluster of servers. It provides the identifier and start time for the process, and the command that initiated the process. |
| STC_AdminUser | This class represents an authorized user of the SAN File System. It provides a user's identifier and role as defined in the Lightweight Directory Access Protocol (LDAP). It also indicates whether the LDAP needs to reauthorize a user for a new request or if the user is still authorized from the last request, and the time remaining in that authorization window. Its methods enable you to clear this authorization window for an individual or all users. |
| STC_MasterMetrics | This class represents the metrics for a cluster. It provides metrics for metadata activity and buffers within the cluster. |
| STC_NodeFan | This class represents the status of a engine's fan. It provides an identifier for the specific fan and the speed of the fan. |
| STC_NodeTemperature | This class represents the temperature state of hardware components of a engine. An instance exists for each temperature sensor on every engine in the cluster. It provides current temperature and threshold values. |
| STC_NodeVitalProductData | This class represents vital product data about the components of a engine. It provides the model and serial number of the host machine and firmware information. |
| STC_NodeVoltage | This class represents the state of the voltage sources of a engine. It provides engine voltage information and warning thresholds. |
| STC_NodeWatchdog | This class represents the watchdog for each engine in a cluster. |
| STC_TankEvents | This class represents a possible event that a server might generate. It provides information about the message that would be logged and the trap that might be generated by the event. |
| STC_TankMetrics | This class represents the metrics for each subordinate server. It provides metrics for metadata activity, locks, and buffers within the server. |
| STC_TankWatchdog | This class represents the Metadata server restart service operations. It provides the state of the Metadata server restart service, probe intervals, total number of retries, total number of absence tests. |

**Current user and process status classes:**

The following diagram shows the hierarchy and definitions of the STC_AdminProcess and STC_AdminUser classes.

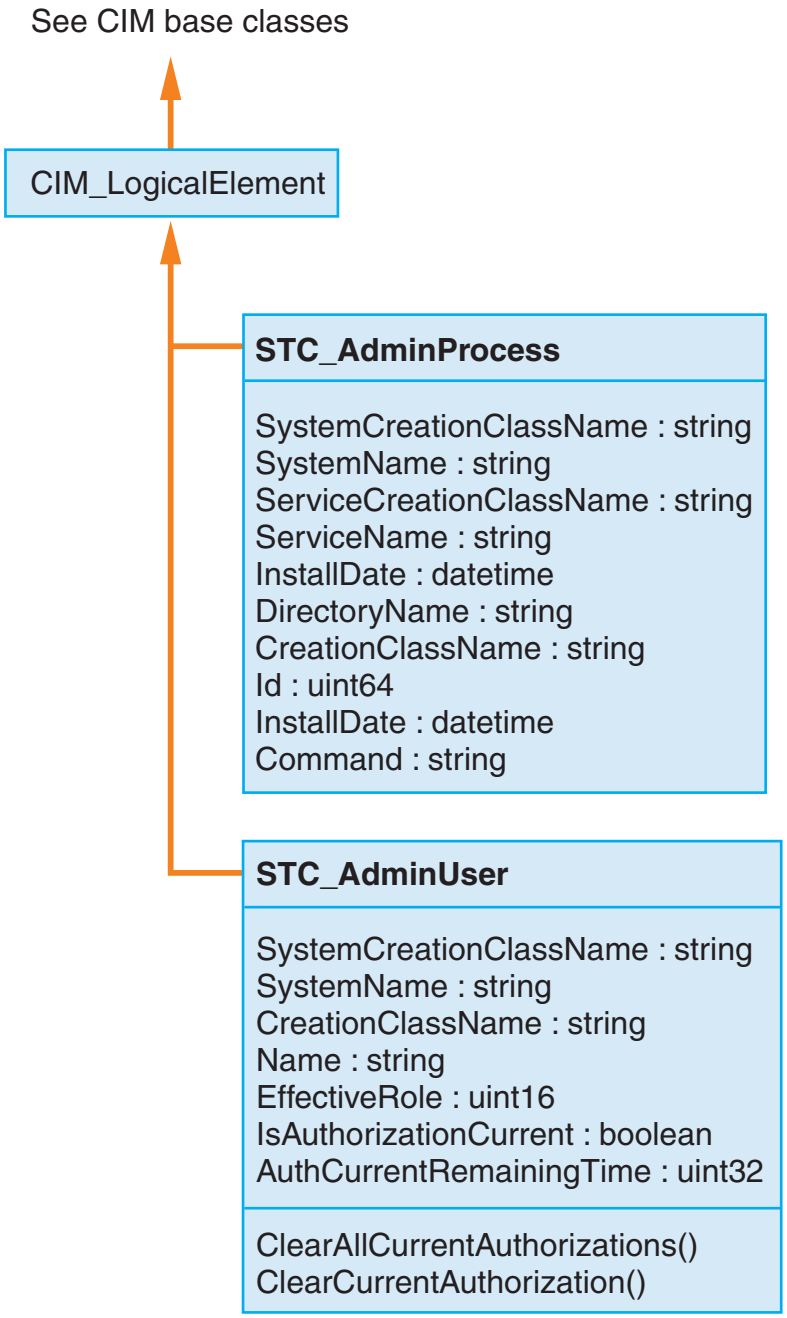



*Figure 13. Current user and process status classes*

**Metrics classes:**

The following diagram shows the hierarchy and definitions of the STC_MasterMetrics and STC_TankMetrics classes.

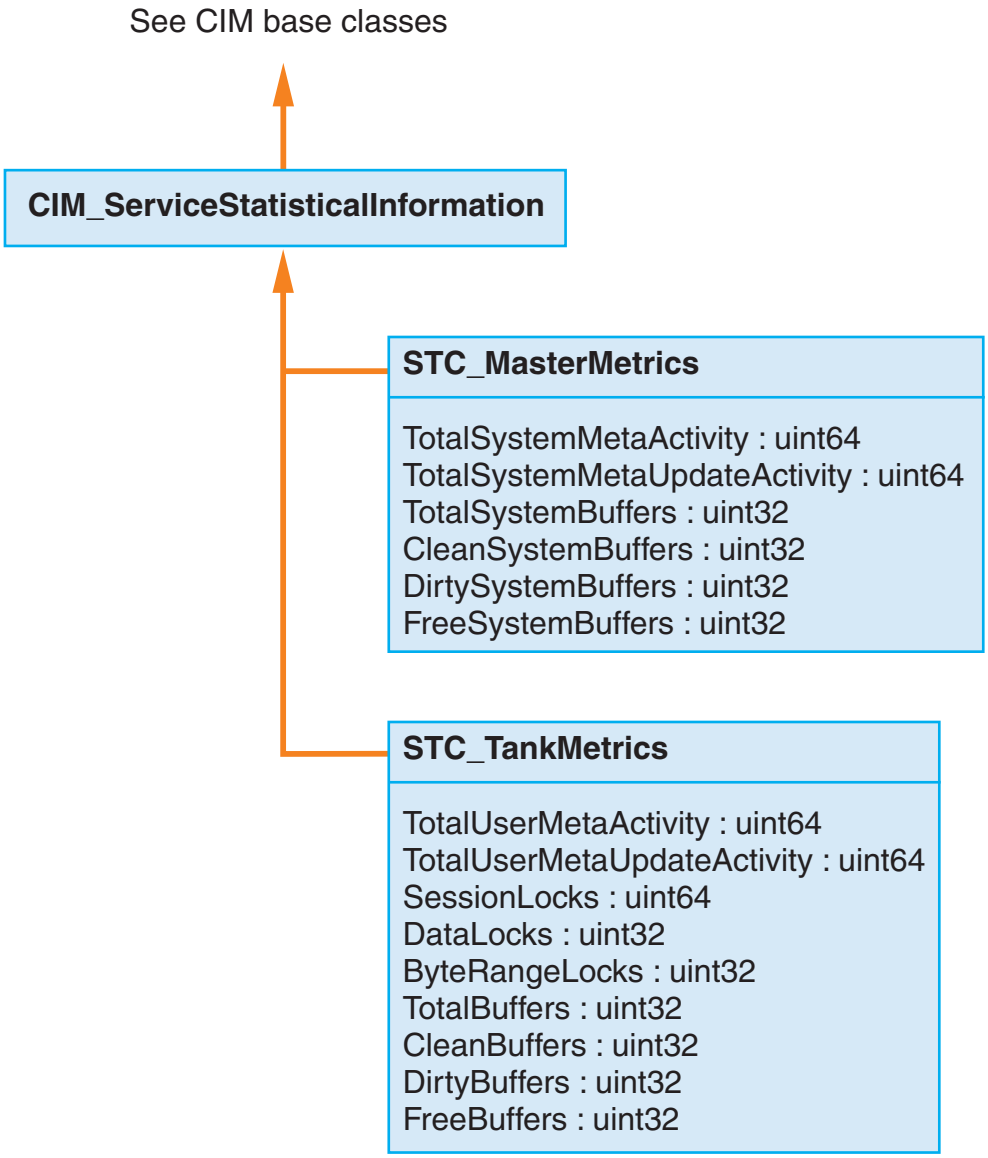



*Figure 14. Metrics classes*

**Engine status classes:**

The following diagram shows the hierarchy and definitions of the STC_NodeFan, STC_NodeTemperature, STC_NodeVitalProductData, STC_NodeVoltage and STC_NodeWatchdog classes.

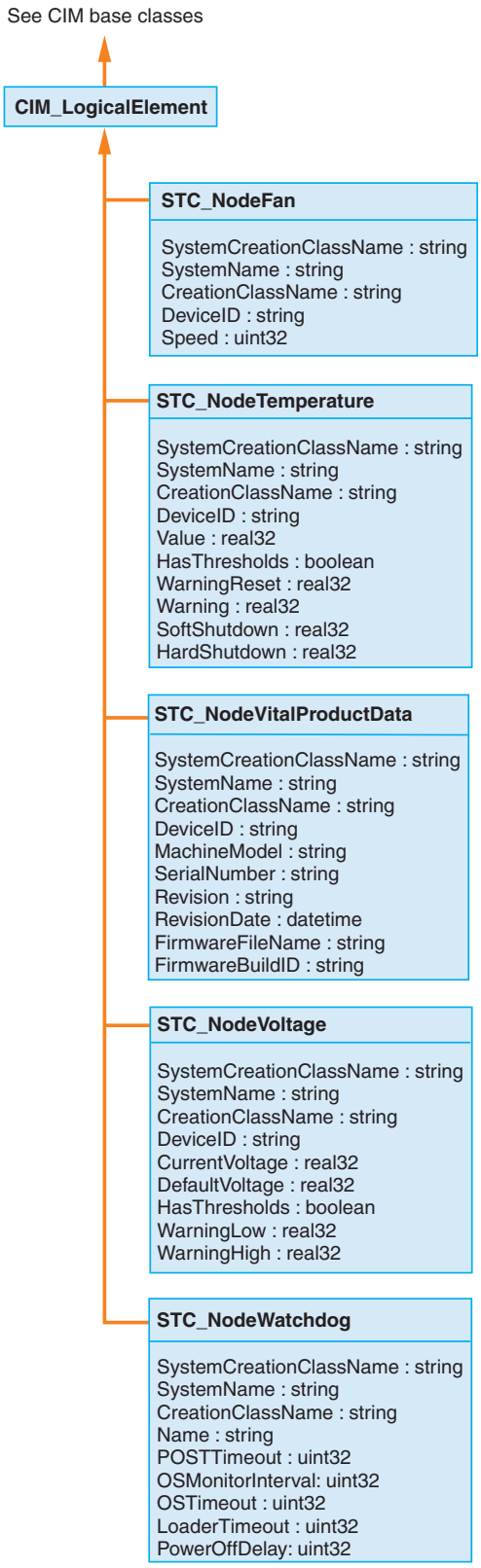



*Figure 15. Engine status classes*

### Server status classes:

The following diagram shows the hierarchy and definitions of the STC_TankEvents and STC_TankWatchdog classes.

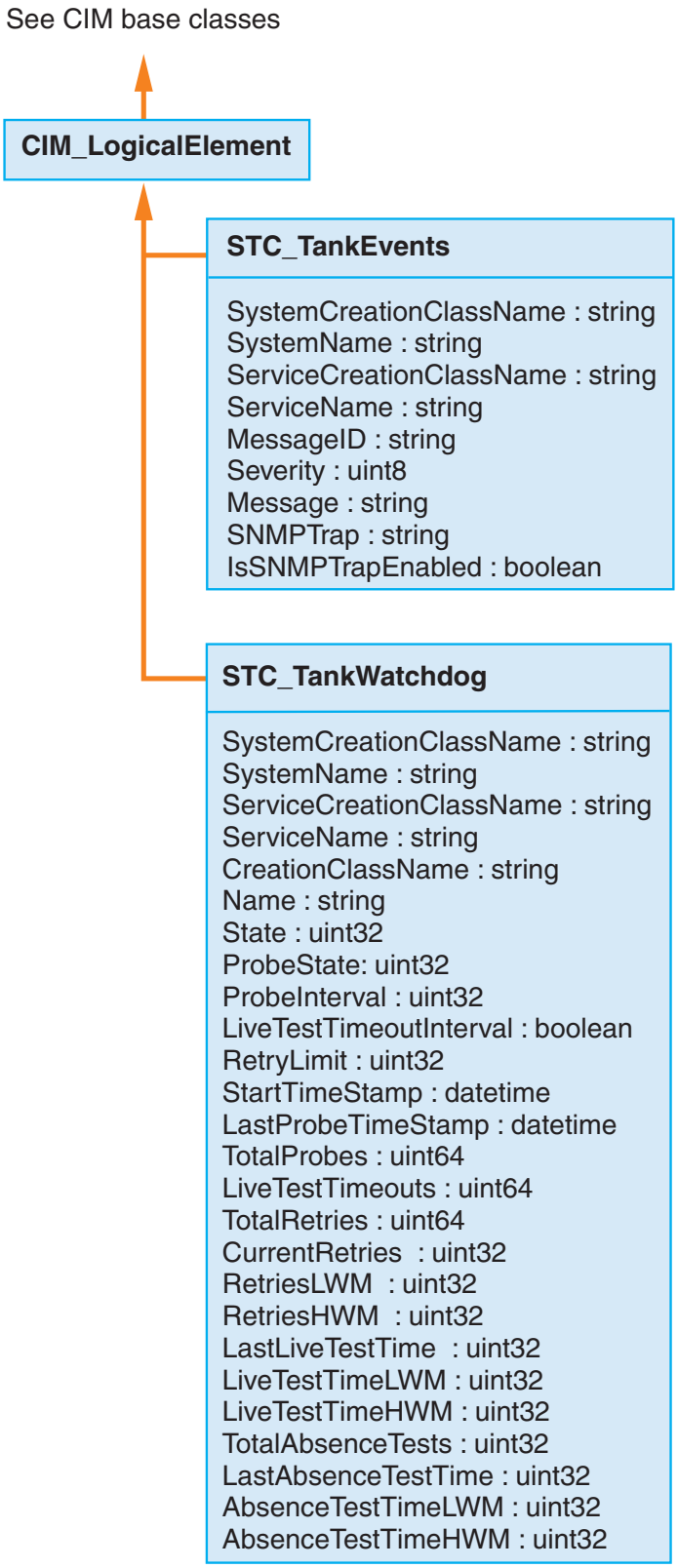



*Figure 16. Service status classes*

**Related topics:**

- "STC_AdminProcess" on page 92
- "STC_AdminUser" on page 93
- "STC_MasterMetrics" on page 109
- "STC_NodeFan" on page 123
- "STC_NodeTemperature" on page 123
- "STC_NodeVitalProductData" on page 124
- "STC_NodeVoltage" on page 125
- "STC_NodeWatchdog" on page 126
- "STC_TankEvents" on page 145
- "STC_TankMetrics" on page 147
- "STC_TankWatchdog" on page 153
-

## SAN File System log classes

Table 6 provides an overview of the classes that represent logs.

*Table 6. SAN File System log classes*

| Name | Description |
|---|---|
| STC_AdminMessageLog | This class represents the message log file for the Administrative server. It extends the STC_MessageLog class. |
| STC_AdminSecurityLog | This class represents the security log file for the Administrative server. It extends the STC_MessageLog class. |
| STC_MDSAuditLog | This class represents the audit log file for a Metadata server. It extends the STC_MessageLog class. |
| STC_MDSEventLog | This class represents the event log file for a Metadata server. It extends the STC_MessageLog class. |
| STC_MDSMessageLog | This class represents the message log file for a Metadata server. It extends the STC_MessageLog class. |
| STC_MessageLog | This class represents log files that are present in the SAN File System. It provides identifier and location information about the log. Its methods enable you to traverse a log forwards and backwards, retrieve a specified number of log records, and clear a log. |

The following diagram shows the hierarchy and definitions of the log classes.

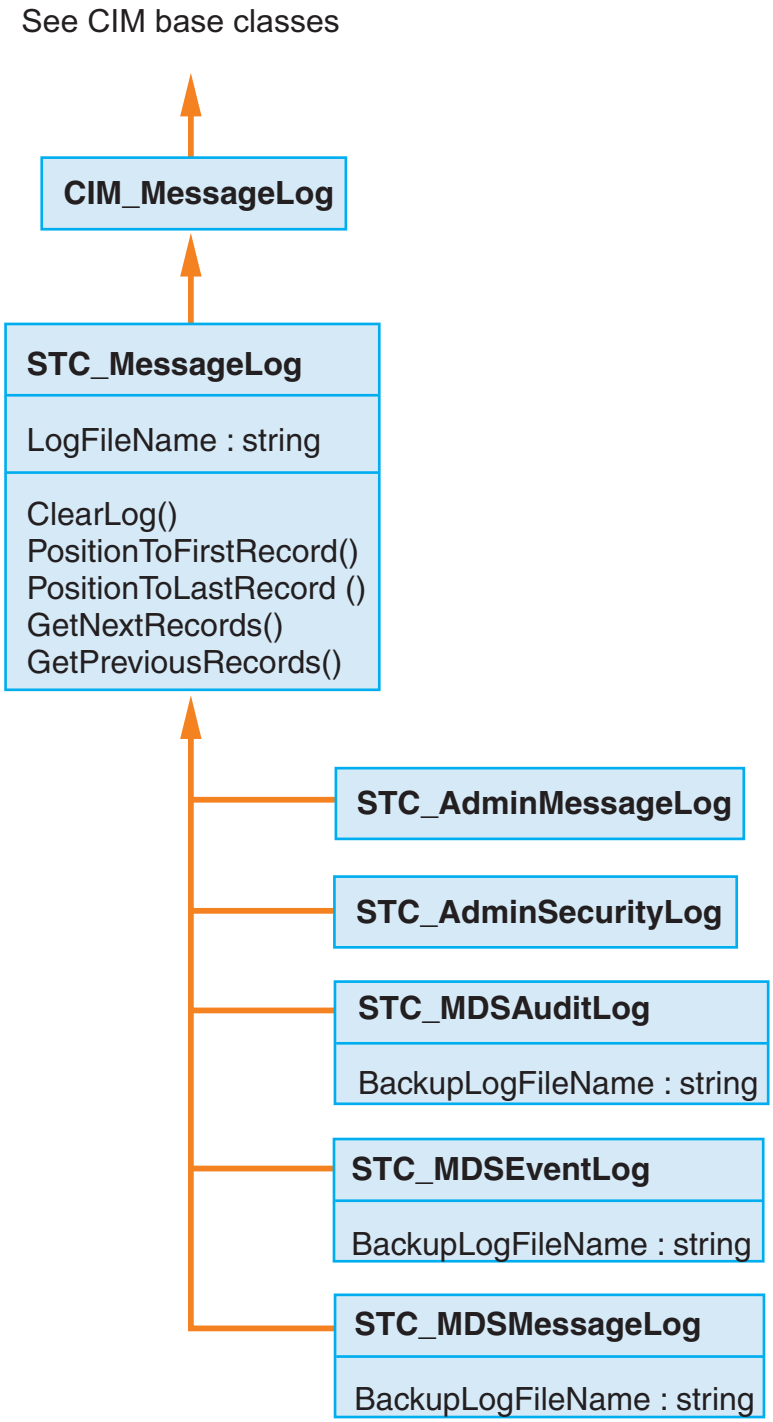



*Figure 17. Log classes*

**Related topics:**

- "STC_AdminMessageLog" on page 92
- "STC_AdminSecurityLog" on page 92
- "STC_MDSAuditLog" on page 116
- "STC_MDSEventLog" on page 117
- "STC_MDSMessageLog" on page 117
- "STC_MessageLog" on page 117

## SAN File System backup classes

Table 7 provides an overview of the classes for backup.

*Table 7. SAN File System backup classes*

| Name | Description |
|---|---|
| STC_PitImage | This class represents a FlashCopy image (also known as the point-in-time image) of a fileset. It describes the identifier of the fileset and the FlashCopy image and the FlashCopy image location. Its methods enable you to create a new FlashCopy image, revert the fileset to a FlashCopy image, or delete a FlashCopy image. |
| STC_SystemMDRAid | This class provides a mechanism to extract system metadata information into a recovery file on a local disk. It provides recovery file identifier, location, and size information, as well as a script to generate commands from the file. Its methods enable you to create and delete a recovery file and to generate commands for recreating metadata from the file. |

The following diagram shows the hierarchy and definitions of the backup classes.

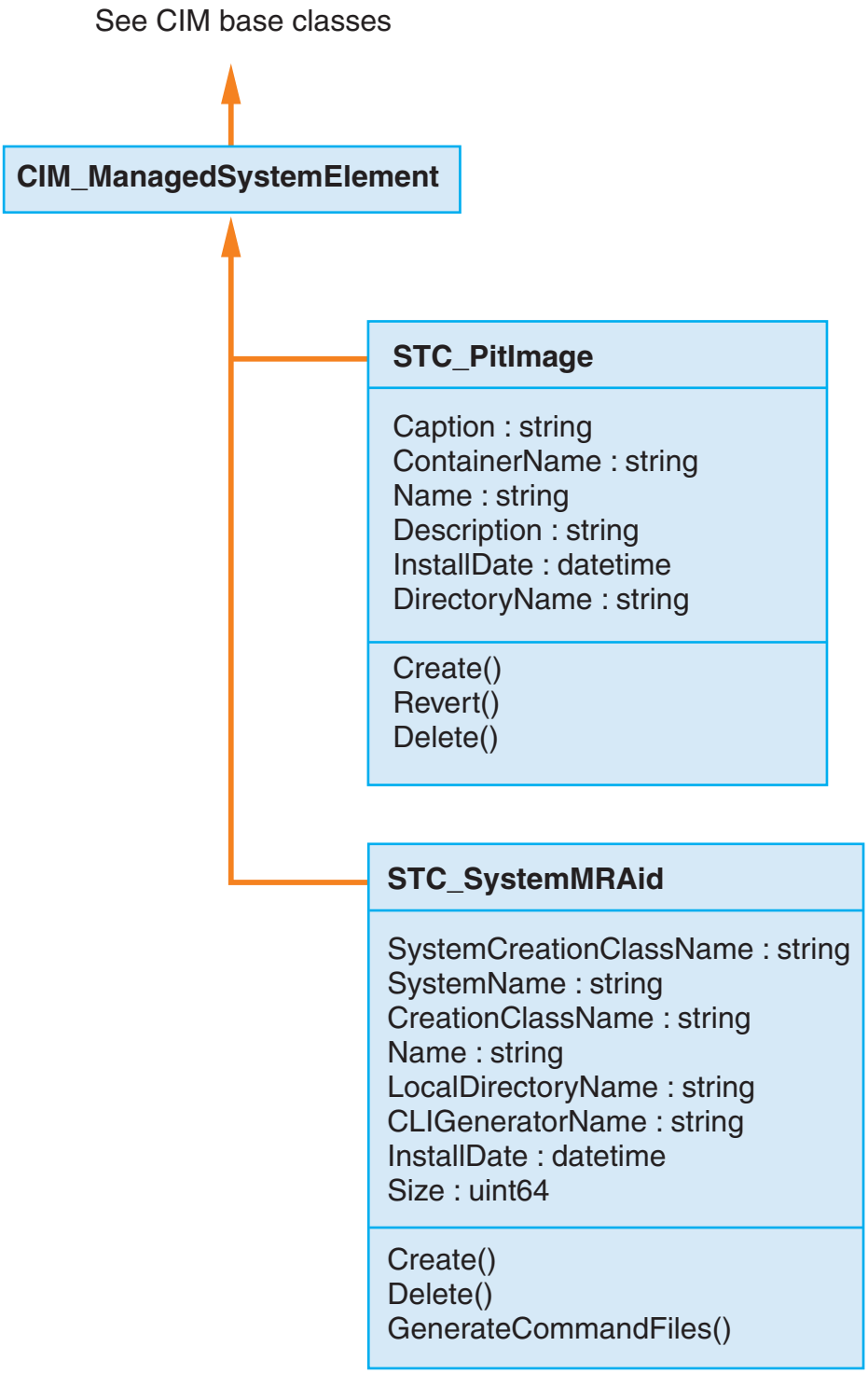



*Figure 18. Backup classes*

**Related topics:**

- "STC_PitImage" on page 127
- "STC_SystemMDRAid" on page 142

## Programming considerations

Third-party CIM clients can manage SAN File System by calling methods in Administrative agent classes. In general, CIM clients should interface with the Administrative agent on the master Metadata server.

Consider the following conventions when programming the classes:

- A CIM client cannot change read-only properties. You can set only those properties that are writeable.
- Although method parameters are independent from properties, often a parameter of a method correlates to a property of its class.

**Related topics:**

- "Role-based access"
- "Dynamic and static methods"

## Role-based access

For security, the object model implements role-based access. It restricts access to administrative operations through user roles that are stored in the Lightweight Directory Access Protocol (LDAP). As described in "User roles" on page 26, the Administrator role has access to the Monitor, Backup, and Operator tasks, plus access to Administrator tasks. To define the minimum role needed for a certain operation, the object model uses the following CIM qualifiers:

- ReadRole –The role required to read a property. A CIM client must have at least the Monitor role to read a class property.
- WriteRole – The role required to write a writeable property. A CIM client must have the Administrator role to change a writeable property.
- ExecuteRole – The role required to invoke a method. A CIM client must have the Administrator role to invoke a method with the following exceptions:
  - Create() (FlashCopy image) method requires the Backup role.
  - Delete() (FlashCopy image) method requires the Backup role.
  - GetRules() method requires the Monitor role.
  - GetNextFOV() method requires the Backup role.
  - GetNextRecords() method requires the Monitor role.
  - GetPreviousRecords() method requires the Monitor role.
  - PositionToFirstRecord() method requires the Monitor role.
  - PositionToLastRecord() method requires the Monitor role.
  - ResetFOV() method requires the Backup role.

**Related topics:**

- "User roles" on page 26

## Dynamic and static methods

CIM supports dynamic and static methods. A static method operates on a class while a dynamic method operates on a specific instance of a class. Therefore, a dynamic method must reference a specific instance of the class. You can call a static method by constructing a CIM object path that just has the class name. For a dynamic method, the CIM object path must be the fully-qualified name of the instance.

The fully-qualified name of the instance might include the class name and the following set of keys that uniquely identify the instance:

- CreationClassName - The class name of this instance
- SystemCreationClassName - The class name of the system to which this instance belongs. The system is usually either a cluster (represented by the STC_Cluster class) or an engine (represented by the STC_ComputerSystem class).
- System Name - The instance name of the system to which this instance belongs
- ServiceCreationClassName - The service class is usually either STC_MasterService or STC_TankService.
- Service Name - The instance name of the service

Most extrinsic methods are dynamic.

With a static method, you only need to specify the class name to indicate the CIM object path.

The following extrinsic methods are static methods:

- Create() (fileset)
- Create() (FlashCopy image)
- Create() (policy)
- Create() (recovery file)
- Create() (storage pool)
- Create() (volume)
- Test()

**Related topics:**

- Chapter 3, “Administrative agent methods”, on page 77

# Chapter 2. Managing SAN File System

This chapter describes how to perform tasks for managing SAN File System using the object model.

**Related topics:**



## Managing the cluster

This section describes the methods you can invoke to perform tasks for managing the cluster.

**Related topics:**



### Changing configuration parameters

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

STC_MasterDisruptiveSetting and STC_MasterDynamicSetting classes contain writeable properties that represent configuration parameters.

**Steps:**

To change configuration parameters, invoke the SetProperty() intrinsic method. Specify the instance name and the property that represents the configuration parameter.

**Related topics:**

- "STC_MasterDisruptiveSetting" on page 106
- "STC_MasterDynamicSetting" on page 107
- "SetProperty()" on page 84

## Changing active cluster states

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The quiescent states restrict activity on all Metadata servers in the cluster.

**Steps:**

The STC_MasterService class provides methods for putting the cluster in a quiescent state and for returning the cluster to an online state.

1. To place the cluster in a quiescent state, invoke the STC_MasterService.QuiesceService method while specifying the mode parameter. The Mode parameter can be one of the following values:

   0: Partly Quiescent - A limited quiescent mode that allows the client to continue file data activity but prevents client metadata activity and new client connections. In this state, a backup would preserve metadata integrity, but might not preserve file data integrity.

   1: Fully Quiescent - A full quiescent mode that suspends all client metadata activity and file data activity and terminates all client sessions. This state allows a backup with metadata and file data integrity.

   2: Administrative Quiescent - A quiescent mode that allows administrative operations that do not permit client activity.
2. To return the cluster to an active state, invoke the STC_MasterService.ResumeService method. This method returns the cluster to a fully online state from the quiescent state.

**Related topics:**

- "STC_MasterService" on page 110
- "QuiesceService() method" on page 113
- "ResumeService() method" on page 114

## Starting the cluster

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_MasterService class provides the method for starting the cluster.

**Steps:**

To start the cluster, invoke the STC_MasterService.StartService method, which brings up all pre-commissioned servers on all engines. This method starts the master Metadata server, checks that the master Metadata server is online and then starts all subordinate servers.

**Related topics:**

- "STC_MasterService" on page 110
- "StartService() method" on page 114

## Stopping the cluster

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_MasterService class provides the method for stopping a cluster.

**Steps:**

To stop a cluster, invoke the STC_MasterService.StopService method, which brings down all servers on all engines.

**Related topics:**

- "STC_MasterService" on page 110
- "StopService() method" on page 116

## Upgrading cluster software

**Prerequisites:**

You must have Administrator privileges to perform this task. A cluster upgrade can occur only after you have upgraded each individual Metadata server to the same new software version.

**Context:**

The STC_MasterService class provides the method for committing a cluster to start using an upgraded software version.

**Steps:**

1. Stop each Metadata server in the cluster, install the new version of software, and restart the server.
2. Invoke the STC_MasterService.CommitUpgrade method to commit the software version upgrade and begin the process of updating the metadata structures.

**Related topics:**

- "STC_MasterService" on page 110
- "CommitUpgrade() method" on page 111

# Managing disaster recovery files

This section describes the methods you can invoke to perform tasks for managing disaster recovery files.

**Related topics:**

- "Creating a recovery file"
- "Deleting a recovery file"
- "Generating recovery commands"

## Creating a recovery file

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_SystemMDRAid class provides the method to create a cluster-wide, metadata recovery file.

**Steps:**

To create a recovery file, invoke the STC_SystemMDRAid.Create method. If you set the IsForce parameter to True, the new recovery file overwrites any existing one.

**Related topics:**

- "STC_SystemMDRAid" on page 142
- "Create() method" on page 142

## Deleting a recovery file

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_SystemMDRAid class provides the method to delete a recovery file.

**Steps:**

To delete a recovery file, invoke the STC_SystemMDRAid.Delete method.

**Related topics:**

- "STC_SystemMDRAid" on page 142
- "Delete() method" on page 143

## Generating recovery commands

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_SystemMDRAid class provides the method to generate recovery commands. The class properties specify the location of the generated command files and the script used to generate command files.

**Steps:**

To generate commands for recreating metadata from a recovery dump file, invoke the STC_SystemMDRAid.GenerateCommandFiles() method. The method can generate the following command files:

- TankSysCLI.auto - This file contains commands to re-create storage pools, filesets, and policies. In case of disaster, this file can run without manual intervention.
- TankSysCLI.volume - This file contains commands to re-create volumes. This file requires manual verification and editing to run.
- TankSysCLI.attachpoint - This file contains commands to re-create fileset attach points. This file requires manual verification, editing, and intervention to run.

You only need these command files for recovery and can postpone their generation until needed. Also the Metadata server does not have to be up and running to generate these files.

**Related topics:**

- "STC_SystemMDRAid" on page 142
- "GenerateCommandFiles() method" on page 144

# Managing engines

This section describes the methods you can invoke to perform tasks for managing engines.

**Related topics:**

- "Powering off the engine"
- "Powering on the engine" on page 58
- "Restarting the engine" on page 58

## Powering off the engine

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_ComputerSystem class provides the method to power off the engine.

**Steps:**

To power off the engine, invoke the STC_ComputerSystem.SetPowerState() method while specifying the following parameters:

- PowerState - 6 for power off, 7 for hibernate, and 8 for soft off
- Time - The time when the power setting should occur. If value is zero, the setting occurs immediately.

**Related topics:**

- "STC_ComputerSystem" on page 96
- "SetPowerState() method" on page 98

## Powering on the engine

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_ComputerSystem class provides the method to power on the engine.

**Steps:**

To power on the engine, invoke the STC_ComputerSystem.SetPowerState() method while specifying the following parameters:

- PowerState - 1 for full power, 2 for low power mode, 3 for standby mode, and 4 for any other type of power save
- Time - The time when the power setting should occur. If value is zero, the setting occurs immediately.

**Related topics:**

- "STC_ComputerSystem" on page 96
- "SetPowerState() method" on page 98

## Restarting the engine

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_ComputerSystem class provides the method to restart the engine.

**Steps:**

To restart the engine, invoke the STC_ComputerSystem.SetPowerState() method while specifying the following parameters:

- PowerState - 5 for power cycle
- Time - The time when the power setting should occur. If value is zero, the setting occurs immediately.

**Related topics:**

- "STC_ComputerSystem" on page 96
- "SetPowerState() method" on page 98

## Managing filesets

This section describes the methods you can invoke to perform tasks for managing filesets.

**Related topics:**

- "Attaching a fileset"
- "Changing a fileset server"
- "Creating a fileset" on page 60
- "Deleting a fileset" on page 60
- "Detaching a fileset" on page 61
- "Moving a fileset" on page 61

### Attaching a fileset

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to attach a fileset.

**Steps:**

To attach a fileset, invoke the STC_Container.Attach() method while specifying the following parameters:

- ExistingDirPath - Current path where the fileset directory resides.
- NewDirName - The directory name of the fileset.

**Related topics:**

- "STC_Container" on page 98
- "Attach() method" on page 101

### Changing a fileset server

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to change the Metadata server that is hosting a fileset.

**Steps:**

To change the Metadata server, invoke the STC_Container.ChangeServer() method while specifying the new server as a parameter.

**Related topics:**

- "Metadata server" on page 18

- "STC_Container" on page 98
- "ChangeServer() method" on page 101

## Creating a fileset

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to create a fileset.

**Steps:**

To create a fileset, invoke the STC_Container.Create() method while specifying the following parameters:

- Name - Your label for the fileset
- Description - Your description of the fileset
- Quota - The maximum size limit, in megabytes
- IsHardQuota - An indicator that a quota limit cannot be extended
- AlertPercentage - The percent of the fileset size that, when reached, causes the server to generate an alert message
- ExistingDirPath - Current path where the fileset directory resides
- NewDirName - The directory name of the fileset
- Server - The name of the server to host this fileset

**Related topics:**

- "STC_Container" on page 98
- "Create() method" on page 102

## Deleting a fileset

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to delete a fileset. You can delete a fileset under the following conditions:

- Fileset is detached
- Fileset is not the global fileset
- Fileset does not have files on it unless the IsForce option is True

**Steps:**

To delete a fileset, invoke the STC_Container.Delete() method. If you set the IsForce parameter to True, the Delete() method deletes the fileset even if it has files in it.

**Related topics:**

- "STC_Container" on page 98

- "Delete() method" on page 104

## Detaching a fileset

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to detach a fileset.

**Steps:**

To detach a fileset, invoke the STC_Container.Detach() method. If you set the IsForce parameter to True, the Detach() method detaches the fileset even if clients are using files in it.

**Related topics:**

- "STC_Container" on page 98
- "Detach() method" on page 104

## Moving a fileset

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Container class provides the method to move a fileset.

**Steps:**

To move a fileset, invoke the STC_Container.Move() method while specifying your new label for the fileset as a parameter.

**Related topics:**

- "STC_Container" on page 98
- "Move() method" on page 105

# Managing FlashCopy images

This section describes the methods you can invoke to perform tasks for managing FlashCopy images.

**Related topics:**

- "Creating a FlashCopy image" on page 62
- "Deleting a FlashCopy image" on page 62
- "Reverting to a previous FlashCopy image" on page 62

## Creating a FlashCopy image

**Prerequisites:**

You must have Backup privileges to perform this task.

**Context:**

The STC_PitImage class provides the method to create a FlashCopy image.

**Steps:**

To create a FlashCopy image, invoke the STC_PitImage.Create() method while specifying the following parameters:

- ContainerName - Your label for the fileset to which this FlashCopy image belongs.
- Name - Your administrative name for the FlashCopy image.
- Description - Your description of the fileset.
- DirectoryName - The new directory name to be given to the FlashCopy image.
- IsForce - Indicator of whether to delete the oldest FlashCopy image to create this one when the number of FlashCopy images exceeds the maximum.

**Related topics:**

- "STC_PitImage" on page 127
- "Create() method" on page 128

## Deleting a FlashCopy image

**Prerequisites:**

You must have Backup privileges to perform this task.

**Context:**

The STC_PitImage class provides the method to delete a FlashCopy image.

**Steps:**

To delete a FlashCopy image, invoke the STC_PitImage.Delete() method. If you set the IsForce parameter to True, the Delete() method deletes the FlashCopy image even if client activity exists.

**Related topics:**

- "STC_PitImage" on page 127
- "Delete() method" on page 129

## Reverting to a previous FlashCopy image

**Attention:** When you revert to a FlashCopy image, all FlashCopy images created after the specified FlashCopy image are deleted. The specified FlashCopy image becomes the primary image for the fileset and no longer appears as an image listed in the .flashcopy directory.

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_PitImage class provides the method to revert to a previous FlashCopy image.

**Steps:**

To revert to a previous a FlashCopy image, invoke the STC_PitImage.Revert() method. If you set the IsForce parameter to True, the Revert() method reverts to the previous image even if client activity exists.

**Related topics:**

- "STC_PitImage" on page 127
- "Revert() method" on page 130

# Managing logs

This section describes the methods you can invoke to perform tasks for managing logs.

**Related topics:**

- "Clearing logs"
- "Retrieving log records"

## Clearing logs

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_MessageLog class provides the method for clearing a message log.

**Steps:**

To clear a message log of all entries, invoke the STC_MessageLog.ClearLog() method.

**Related topics:**

- "STC_MessageLog" on page 117
- "ClearLog() method" on page 118

## Retrieving log records

**Prerequisites:**

You must have Monitor privileges to perform this task.

**Steps:**

The STC_MessageLog class provides methods for positioning an iterator in the log and retrieving a number of log records.

1. To create an iterator and position it at the beginning of the log, invoke the STC_MessageLog.PositionToFirstRecord() method. To create an iterator and position it at the end of the log, invoke the STC_MessageLog.PositionToLastRecord() method. Both methods return the identifier for the iterator as a parameter.
2. To retrieve a specified number of records from the message log starting from the record indicated by the IterationIdentifier parameter, invoke the STC_MessageLog.GetNextRecords() method. To retrieve a specified number of records from the message log ending from the record indicated by the IterationIdentifier parameter, invoke the STC_MessageLog.GetPreviousRecords() method. With these methods, specify the following parameters:
   - IterationIdentifier - Identifier for the iterator
   - NumberOfEntries - The number of records to be retrieved

   These methods return the following parameters:

   IterationIdentifier - Identifier for the iterator

   NumberOfEntries - The actual number of records that were retrieved

   MessageTimeStamp - The timestamp for each message

   MessageID - The identifier for each message

   MessageType - The type of each message: 1 for Normal, 2 for Event, 3 for Audit, and 4 for Trace

   SourceNode - The identifier of the engine that originated each message.

   Severity - The severity of each message: 0 for Information, 1 for Warning, 2 for Error, and 3 for Severe

   MessageString - The content of each message

**Related topics:**


- "GetNextRecords() method" on page 118
- "GetPreviousRecords() method" on page 120
- "PositionToFirstRecord() method" on page 121
- "PositionToLastRecord method" on page 122
- "STC_MessageLog" on page 117


## Managing Metadata servers

This section describes the methods you can invoke to perform tasks for managing Metadata servers.

**Related topics:**


- "Changing the master server" on page 65
- "Starting a Metadata server" on page 65
- "Starting the Metadata server restart service" on page 65
- "Stopping a Metadata server" on page 66
- "Stopping the Metadata server restart service" on page 66

## Changing the master server

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_TankService class provides the method for making a different server the master Metadata server, if the master Metadata server is irrecoverably lost. A master can be lost because of hardware failures, software failures, or partitioned networks. Before changing the master Metadata server, be aware of the following considerations:

- You cannot change the master when the cluster is down.
- When a master is lost, the subordinate servers are no longer in operational states.

**Steps:**

1. Ensure that the previous master is down by turning the power off to the engine hosting the down master Metadata server.
2. Invoke the STC_TankService.BecomeMaster() method.
3. Manually log in to the engine and edit the STC.TankService class property to set the IsMaster property to True.

**Related topics:**

- "STC_TankService" on page 149
- "BecomeMaster() method" on page 150

## Starting a Metadata server

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_TankService class provides the method for starting a Metadata server on an engine.

**Steps:**

To start a Metadata server, invoke the STC_TankService.StartService method.

**Related topics:**

- "STC_TankService" on page 149
- "StartService() method" on page 114

## Starting the Metadata server restart service

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_TankWatchdog class provides the method for enabling the Metadata server restart service.

**Steps:**

To enable the Metadata server restart service, invoke the STC_TankWatchdog.Enable() method.

**Related topics:**

- "STC_TankWatchdog" on page 153
- "Disable() method" on page 155

## Stopping a Metadata server

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_TankService class provides the method for stopping a Metadata server on an engine.

**Steps:**

To stop a Metadata server, invoke the STC_TankService.StopService method.

**Related topics:**

- "STC_TankService" on page 149
- "StopService() method" on page 116

## Stopping the Metadata server restart service

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_TankWatchdog class provides the method for disabling the Metadata server restart service.

**Steps:**

To disable the Metadata server restart service, invoke the STC_TankWatchdog.Disable() method.

**Related topics:**

- "STC_TankWatchdog" on page 153
- "Disable() method" on page 155

# Managing policies

This section describes the methods you can invoke to perform tasks for managing policies.

**Related topics:**

- "Activating a policy"
- "Creating a policy"
- "Deleting a policy" on page 68
- "Viewing a policy" on page 68

## Activating a policy

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_PolicySet class provides the method for activating a policy.

**Steps:**

To activate a policy, invoke the STC_PolicySet.Activate() method.

**Related topics:**

- "STC_PolicySet" on page 131
- "Activate() method" on page 132

## Creating a policy

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_PolicySet class provides the method for creating a new policy.

**Steps:**

To create a new STC_PolicySet instance, invoke the STC_PolicySet.Create() method while specifying the following parameters:

- Name - A label for this policy.
- Description - Your description of this policy.
- PolicyRules - The set of policy rules belonging to this policy
- IsForce - Indicator of whether to modify an existing policy

**Related topics:**

- "STC_PolicySet" on page 131
- "Create() method" on page 133

## Deleting a policy

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_PolicySet class provides the method for deleting a policy.

**Steps:**

To delete a policy, invoke the STC_PolicySet.Delete() method.

**Related topics:**

- "STC_PolicySet" on page 131
- "Delete() method" on page 134

## Viewing a policy

**Prerequisites:**

You must have Monitor privileges to perform this task.

**Context:**

The STC_PolicySet class provides the method for retrieving the rules associated with a policy.

**Steps:**

To retrieve the rules, invoke the STC_PolicySet.GetRules() method. This method returns the rules as a string parameter.

**Related topics:**

- "STC_PolicySet" on page 131
- "GetRules() method" on page 134

# Managing storage pools

This section describes the methods you can invoke to perform tasks for managing storage pools.

**Related topics:**

- "Creating a storage pool" on page 69
- "Deleting a storage pool" on page 69
- "Moving a storage pool" on page 69
- "Setting the default storage pool" on page 70

## Creating a storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_StoragePool class provides the method for creating new storage pools.

**Steps:**

To create a new STC_StoragePool instance, invoke the STC_StoragePool.Create() method while specifying the following parameters:

- Name - Your label for the storage pool
- Description - A string that is your description of the storage pool.
- PartitionSize - The partition size, in megabytes, to use when a fileset allocates space.
- BlockSize - The allocation strategy to use for files on this storage pool.
- AlertPercentage - The percentage of the storage pool size that, when reached, causes the server to generate an alert message.

**Related topics:**

- "STC_StoragePool" on page 137
- "Create() method" on page 138

## Deleting a storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_StoragePool class provides the method for deleting a storage pool.

**Steps:**

To delete a storage pool, invoke the STC_StoragePool.Delete() method.

**Related topics:**

- "STC_StoragePool" on page 137
- "Delete() method" on page 140

## Moving a storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_StoragePool class provides the method to move or rename a storage pool. It creates a new storage pool with the specified name and migrates the data and capabilities to the new name.

**Steps:**

To move or rename a storage pool, invoke the STC_StoragePool.Move() method and specify the new label for the storage pool.

**Related topics:**

- "STC_StoragePool" on page 137
- "Move() method" on page 140

## Setting the default storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_StoragePool class provides the method for changing a user storage pool to the default storage pool. The PoolType property of the STC_StoragePool class changes from User to User Default.

**Steps:**

To change the storage pool type from User to User Default, invoke the STC_StoragePool.SetDefault() method.

**Related topics:**

- "STC_StoragePool" on page 137
- "SetDefault() method" on page 141

# Managing users

This section describes the methods you can invoke to perform tasks for managing users.

**Related topics:**

- "Timing out all user authorizations"
- "Timing out a user authorization" on page 71

## Timing out all user authorizations

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_AdminUser class provides the method for clearing all current validation windows.

**Steps:**

To clear all current validation windows, invoke the STC_AdminUser.ClearAllCurrentAuthorizations() method.

**Related topics:**

- "STC_AdminUser" on page 93
- "ClearAllCurrentAuthorizations() method" on page 94

## Timing out a user authorization

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_AdminUser class provides the method for clearing the validation window of a user.

**Steps:**

To clear a user's validation window, invoke the STC_AdminUser.ClearCurrentAuthorization() method.

**Related topics:**

- "STC_AdminUser" on page 93
- "ClearCurrentAuthorization() method" on page 94

# Managing volumes and data

This section describes the methods you can invoke to perform tasks for managing volumes and data.

**Related topics:**

- "Activating a suspended volume"
- "Adding a volume to a storage pool" on page 72
- "Checking metadata" on page 72
- "Removing volumes from a storage pool" on page 73
- "Retrieving file entries on a volume" on page 74
- "Suspending a volume" on page 74

## Activating a suspended volume

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Volume class provides the method for resuming suspended partition allocations on a volume.

**Steps:**

To activate a suspended volume, invoke the STC_Volume.ResumeAllocation() method.

**Related topics:**

- "STC_Volume" on page 156
- "ResumeAllocation() method" on page 162

## Adding a volume to a storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Volume class provides methods for adding volumes to a storage pool. You add a volume to a storage pool when you create a new volume or move or rename an existing volume.

**Steps:**

To create a new STC_Volume instance, invoke the STC_Volume.Create() method while specifying the following parameters:

- OSDeviceName - The file path to the storage device.
- StoragePoolName - Your label for the storage pool to which you are attaching the volume.
- VolumeName - Your label for the volume.
- Description - Your description of the volume.
- IsForce - Indicator of whether a volume can be deleted even if it has files on it.
- IsSuspendAllocations - Indicator of whether the volume state is set to Suspend Allocations.

To rename or move an STC_Volume instance, invoke the STC_Volume.Move() method while specifying your new label for the volume.

**Related topics:**

- "STC_Volume" on page 156
- "Create() method" on page 157
- "Move() method" on page 160

## Checking metadata

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_MasterService class provides a method for checking and repairing metadata. It also provides a method for stopping a metadata check that is in progress.

You can use the FileSystemCheck() method to check and repair metadata. It enables you to specify the following options:

- Check the integrity of the structure and the content of the metadata.
- Check the integrity of the system metadata and the file metadata.
- Limit the user metadata checking to a subset of filesets.

You can restrict this operation to check-only or check and repair. The message log contains a report generated by this method. If you did not limit the mode to check-only, the system automatically salvages and repairs the damaged data if possible. Some types of repair require manual intervention from the administrator. In those cases, the system places the cluster state into Administrative mode.

**Notes:**

1. This method is a long-running process. If there is a cluster reformation while the method is running, this method might stop.
2. Only one FileSystemCheck() operation can be in progress at time.

**Steps:**

To check only or check and repair metadata, invoke the STC_MasterService.FileSystemCheck() method while you specify the following parameters:

- IsCheckOnly - Indicator of whether to only check and not repair.
- CheckScope - A bitmap that indicates the scope of the check, which could be a check of the structure or content or both.
- Type - A bitmap that indicates the type of metadata to be checked. The possible types are system or user (file) or both.
- ContainerList - A list of filesets for the method to check or repair if the Type parameter indicates User and not System.

To stop a metadata check that is in progress, invoke the STC_MasterService.StopFileSystemCheck() method.

**Related topics:**

- "STC_MasterService" on page 110
- "FileSystemCheck() method" on page 111
- "StopFileSystemCheck() method" on page 115

## Removing volumes from a storage pool

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Volume class provides a method for deleting an existing volume.

**Steps:**

To delete an existing volume, invoke the STC_Volume.Delete() method. Specify with the IsForce parameter whether the method should delete a volume even if it has files on it.

If the IsForce parameter is True, this method deletes all the files that partly or fully exist on the volume before deleting the volume. If the IsForce parameter is False, this method first drains the volume. It moves the file data that resides on the volume to other volumes in the same storage pool. If the volume drain fails, the method places the volume into a Suspend Allocations state, which requires manual administrative action.

**Related topics:**

- "STC_Volume" on page 156
- "Delete() method" on page 158

## Retrieving file entries on a volume

**Prerequisites:**

You must have Backup privileges to perform this task.

**Steps:**

The STC_Volume class provides methods for positioning an iterator in the volume and retrieving a specific file entry on the volume (FOV).

1. To create an iterator that fetches each file entry that resides on the volume, invoke the STC_Volume.ResetFOV() method. This method returns the identifier for the iterator as a parameter.
2. To retrieve a file entry on the volume, invoke the STC_Volume.GetNextFOV() method. Specify the FOVHandle parameter, which is the FOV iteration identifier.
3. The method returns the following parameters:

   FOVHandle - The FOV iteration identifier

   FOVEntry - The file entry

**Related topics:**

- "STC_Volume" on page 156
- "ResetFOV() method" on page 161
- "GetNextFOV() method" on page 159

## Suspending a volume

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_Volume class provides the method for suspending partition allocations on a volume. A Metadata server cannot allocate new data on the volume.

**Steps:**

To suspend a volume, invoke the STC_Volume.SuspendAllocation() method.

**Related topics:**

- "STC_Volume" on page 156

- "SuspendAllocation() method" on page 162

## Collecting problem determination data

**Prerequisites:**

You must have Administrator privileges to perform this task.

**Context:**

The STC_ComputerSystem class provides a method to invoke the one-button data collector utility that collects server information and system information that is needed for problem determination.

**Steps:**

1. Determine the directory in which the data will collect. By default, the information collects in the /usr/tank/pmf directory. Use the TANKDIR environment variable, if you want to specify a different directory. If the utility generates any stdout output and any stderr output, the output collects in the /tmp/obdcout file on the local disk of the engine.
2. Invoke the STC_ComputerSystem.OneButtonDataCollector() method to invoke the one-button data collector utility.

See the *Maintenance and Problem Determination Guide* for details about the information the utility collects.

**Related topics:**

- "STC_ComputerSystem" on page 96
- "OneButtonDataCollector() method" on page 97

# Chapter 3. Administrative agent methods

This chapter describes the intrinsic and extrinsic methods that the Administrative agent classes provide. These methods are required for implementing the functionality of the Administrative agent.

**Related topics:**

- "Intrinsic methods"
- "Extrinsic methods" on page 85

## Intrinsic methods

Intrinsic methods are provided by the Distributed Management Task Force Inc. (DMTF) for the purpose of modeling a typical CIM operation. Intrinsic methods provide the basic means that enable you to work with an object model.

The Administrative agent uses the following intrinsic methods:

*Table 8. SAN File System Intrinsic Methods*

| Method name | Functional group |
|---|---|
| EnumerateClasses() | Basic read |
| EnumerateClassNames() | Basic read |
| EnumerateInstanceNames() | Basic read |
| EnumerateInstances() | Basic read |
| EnumerateQualifiers() | Qualifier declaration |
| ExecQuery() | Query execution |
| GetClass() | Basic read |
| GetInstance() | Basic read |
| GetProperty() | Basic read |
| GetQualifier() | Qualifier declaration |
| ModifyInstance() | Instance manipulation |
| SetProperty() | Basic write |

The CIM intrinsic methods are defined in the *Distributed Management Task Force Inc. (DMTF) Specification for CIM Operations over HTTP* available at www.dmtf.org/standards/standard_wbem.php.

**Related topics:**

- "EnumerateClasses()" on page 78
- "EnumerateClassNames()" on page 78
- "EnumerateInstanceNames()" on page 79
- "EnumerateInstances()" on page 80
- "EnumerateQualifiers()" on page 80
- "ExecQuery()" on page 81
- "GetClass()" on page 81

- "GetInstance()" on page 82
- "GetProperty()" on page 82
- "GetQualifier()" on page 83
- "ModifyInstance()" on page 83
- "SetProperty()" on page 84

## EnumerateClasses()

You can use the the EnumerateClasses() method to enlist all subclasses of a single object class or all classes of the same object type in the target namespace.

**Parameters:**

You can specify the following parameters of the EnumerateClasses() method:

*Table 9. EnumerateClasses() method parameters*

| Name | 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 | If set to True, returns all qualifiers for the class, its properties, methods, or method parameters; if set to False, returns no qualifiers. |
| IncludeClassOrigin | boolean | If set to True, returns the CLASSORIGIN attribute of the class. |

The EnumerateClasses() method enumerates the specified one or more classes or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## EnumerateClassNames()

You can use the the EnumerateClassNames() method to enlist the names of all subclasses of a single object class or the names of all classes of the same object type in the target namespace.

**Parameters:**

You can specify the following parameters of the EnumerateClassNames() method:

*Table 10. EnumerateClassNames() method parameters*

| Name | 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. |

The EnumerateClassNames() method enumerates the specified one or more classes or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## EnumerateInstanceNames()

You can use the the EnumerateInstanceNames() method to enlist all the names of the instances of the same object class in the target namespace.

**Parameters:**

You can specify the following parameters of the EnumerateInstanceNames() method:

*Table 11. EnumerateInstanceNames() method parameters*

| Name | Type | Description |
|---|---|---|
| ClassName | string | Defines the name of the class for which instances are to be returned. |

The EnumerateInstanceNames() method enumerates the specified names of the instances or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## EnumerateInstances()

You can use the the EnumerateInstances() method to enlist all instances of the same object class in the target namespace.

**Parameters:**

You can specify the following parameters of the EnumerateInstances() method:

*Table 12. EnumerateInstances() method parameters*

| Name | Type | Description |
|---|---|---|
| ClassName | string | Defines the name of the class for which instances are to be returned. |
| DeepInheritance | boolean | True returns all instances and all properties of the instance, including those added by creating subclasses. False returns only properties defined for the specified class. |
| 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 each instance, its properties, methods, or method parameters. False returns no qualifiers. |
| IncludeClassOrigin | boolean | If set to True, returns the CLASSORIGIN attribute of the class within the instance. |

The EnumerateInstances() method enumerates the specified instances or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## EnumerateQualifiers()

You can use the the EnumerateQualifiers() method to enumerate qualifier declarations in the target namespace.

The EnumerateQualifiers() method enumerates the specified qualifier declarations or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## ExecQuery()

You can use the the ExecQuery() method to execute a query against the target namespace.

**Parameters:**

You can specify the following parameters of the ExecQuery() method:

*Table 13. ExecQuery() method parameters*

| Name | Type | Description |
|---|---|---|
| QueryLanguage | string | Defines the query language in which the query parameter is expressed. SAN File System supports the WQL Level 1 query language, which is represented by the string WBEMSQL1. |
| Query | string | Defines the query to be executed. |

The ExecQuery() method retrieves one or more classes or instances or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## GetClass()

You can use the GetClass() method to retrieve a single object class from the target namespace.

**Parameters:**

You can specify the following parameters of the GetClass() method:

*Table 14. GetClass() method parameters*

| Name | Type | Description |
|---|---|---|
| ClassName | string | Defines the name of the class to retrieve. |
| LocalOnly | boolean | If set to True, returns all properties, methods, and qualifiers overridden within the definition of the class. |
| IncludeQualifiers | boolean | If set to True, returns all qualifiers for the class, its properties, methods, or method parameters; if set to False, returns no qualifiers. |
| IncludeClassOrigin | boolean | If set to True, returns the CLASSORIGIN attribute of the class. |

The GetClass() method returns the specified class or one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)

- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## GetInstance()

You can use the the GetInstance() method to retrieve a single instance of an object from the target namespace.

**Parameters:**

You can specify the following parameters of the GetInstance() method:

*Table 15. GetInstance() method parameters*

| Name | Type | Description |
|---|---|---|
| InstanceName | string | Defines the name of the instance to retrieve. |
| LocalOnly | boolean | If set to True, returns all properties, methods, and qualifiers overridden within the definition of the class. |
| IncludeQualifiers | boolean | If set to True, returns all qualifiers for the class, its properties, methods, or method parameters; if set to False, returns no qualifiers. |
| IncludeClassOrigin | boolean | If set to True, returns the CLASSORIGIN attribute of the class. |

The GetInstance() method returns the specified class or one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)
- 6 (CIM_ERR_NOT_FOUND)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## GetProperty()

You can use the the GetProperty() method to retrieve a single attribute value of an instance in the target namespace.

**Parameters:**

You can specify the following parameters of the GetProperty() method:

*Table 16. GetProperty() method parameters*

| Name | Type | Description |
|---|---|---|
| InstanceName | string | Defines the name of the instance. |

*Table 16. GetProperty() method parameters (continued)*

| Name | Type | Description |
|---|---|---|
| Property | string | The name of the property whose value is to be returned from the instance. |

The GetProperty() method returns the specified property of the target instance or one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)
- 6 (CIM_ERR_NOT_FOUND)
- 12 (CIM_ERR_NO_SUCH_PROPERTY)

**Related topics:**


- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85


## GetQualifier()

You can use the GetQualifier() method to retrieve a single qualifier declaration from the target namespace.

**Parameters:**

You can specify the following parameters of the GetQualifier() method:

*Table 17. GetQualifier() method parameters*

| Name | Type | Description |
|---|---|---|
| QualifierName | string | Defines the qualifier whose declaration is to be returned. |

The GetQualifier() method returns the specified qualifier or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 6 (CIM_ERR_NOT_FOUND)

**Related topics:**


- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85


## ModifyInstance()

You can use the ModifyInstance() method to modify an existing instance of an object in the target namespace.

**Parameters:**

You can specify the following parameters of the ModifyInstance() method:

*Table 18. ModifyInstance() method parameters*

| Name | Type | Description |
|---|---|---|
| InstanceName | string | Defines the name of the instance to modify |

The ModifyInstance() method returns the specified class or one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)
- 6 (CIM_ERR_NOT_FOUND)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## SetProperty()

You can use the SetProperty() method to define a single property value of an instance in the target namespace. You can use it to change a configuration parameter value by changing the writeable property in an instance of the STC_MasterDisruptiveSetting or STC_MasterDynamicSetting class.

**Parameters:**

You can specify the following parameters of the SetProperty() method:

*Table 19. SetProperty() method parameters*

| Name | Type | Description |
|---|---|---|
| InstanceName | string | Defines the name of the instance. |
| Property | string | The name of the property whose value is to be defined. |

The SetProperty() method defines the property name of the target instance or returns one of the following error codes:

- 1 (CIM_ERR_FAILED)
- 2 (CIM_ERR_ACCESS_DENIED)
- 3 (CIM_ERR_INVALID_NAMESPACE)
- 4 (CIM_ERR_INVALID_PARAMETER)
- 5 (CIM_ERR_INVALID_CLASS)
- 6 (CIM_ERR_NOT_FOUND)
- 12 (CIM_ERR_NO_SUCH_PROPERTY)
- 13 (CIM_ERR_TYPE_MISMATCH)

**Related topics:**

- "Intrinsic methods" on page 77
- "Intrinsic method return codes" on page 85

## Intrinsic method return codes

For easier diagnosis, return codes from method invocations have the same general meanings. Table 20 describes the meanings of intrinsic method return codes.

*Table 20. Intrinsic method return codes*

| Code | Symbolic Name | Definition |
|---|---|---|
| 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 passed to the method were invalid. |
| 5 | CIM_ERR_INVALID_CLASS | The specified class does not exist. |
| 6 | CIM_ERR_NOT_FOUND | The requested object could not be found. |
| 12 | CIM_ERR_NO_SUCH_PROPERTY | The specified property does not exist. |
| 13 | CIM_ERR_TYPE_MISMATCH | The value supplied is incompatible with the type. |

When you invoke an intrinsic method, the Administrative agent returns more specific information along with the intrinsic method return code. The Administrative agent includes a string in the form SSG:nn where nn is an extrinsic method return code. For example, if you invoke EnumerateInstances() method on the STC_Container class when the server is down, you receive the CIM_ERR_FAILED return code as well as the string SSG:65. The 65 represents the return code meaning: "Server not available". See "Extrinsic method return codes" on page 88 for a description of these return code meanings.

**Related topics:**

- "Intrinsic methods" on page 77

# Extrinsic methods

Extrinsic methods are specific to Administrative agent object classes. They add functionality to the object classes.

The Administrative agent supports the following extrinsic methods:

*Table 21. SAN File System Extrinsic Methods*

| Method origin (derived from) | Method name |
|---|---|
| STC_AdminUser | ClearAllCurrentAuthorizations() method |
| | ClearCurrentAuthorization() method |
| STC_ComputerSystem | OneButtonDataCollector() method |
| | SetPowerState() method |

*Table 21. SAN File System Extrinsic Methods (continued)*

| Method origin (derived from) | Method name |
|---|---|
| STC_Container | Attach() method |
| | ChangeServer() method |
| | Create() method |
| | Delete() method |
| | Detach() method |
| | Move() method |
| STC_MasterService | CommitUpgrade() method |
| | FileSystemCheck() method |
| | QuiesceService() method |
| | ResumeService() method |
| | StartService() method |
| | StopFileSystemCheck() method |
| | StopService() method |
| STC_MessageLog | ClearLog() method |
| | GetNextRecords() method |
| | GetPreviousRecords() method |
| | PositionToFirstRecord() method |
| | PositionToLastRecord method |
| STC_PitImage | Create() method |
| | Delete() method |
| | Revert() method |
| STC_PolicySet | Activate() method |
| | Create() method |
| | Delete() method |
| | GetRules() method |
| STC_StoragePool | Create() method |
| | Delete() method |
| | Move() method |
| | SetDefault() method |
| STC_SystemMDRAid | Create() method |
| | Delete() method |
| | GenerateCommandFiles() method |
| STC_TankEvents | Test() method |
| STC_TankService | BecomeMaster() method |
| | StartService() method |
| | StopService() method |
| STC_TankWatchdog | Enable() method |
| | Disable() method |

*Table 21. SAN File System Extrinsic Methods (continued)*

| Method origin (derived from) | Method name |
|---|---|
| STC_Volume | Create() method |
| | Delete() method |
| | GetNextFOV() method |
| | Move() method |
| | ResetFOV() method |
| | ResumeAllocation() method |
| | SuspendAllocation() method |

Each extrinsic method is described with its object class in the Chapter 4, "Administrative agent object classes", on page 91 chapter in this document.

**Related topics:**

- "Activate() method" on page 132
- "Attach() method" on page 101
- "BecomeMaster() method" on page 150
- "ChangeServer() method" on page 101
- "ClearAllCurrentAuthorizations() method" on page 94
- "ClearCurrentAuthorization() method" on page 94
- "ClearLog() method" on page 118
- "CommitUpgrade() method" on page 111
- "Create() method" on page 102
- "Create() method" on page 128
- "Create() method" on page 133
- "Create() method" on page 142
- "Create() method" on page 138
- "Create() method" on page 157
- "Delete() method" on page 104
- "Delete() method" on page 129
- "Delete() method" on page 134
- "Delete() method" on page 143
- "Delete() method" on page 140
- "Delete() method" on page 158
- "Detach() method" on page 104
- "Disable() method" on page 155
- "Enable() method" on page 155
- "FileSystemCheck() method" on page 111
- "GenerateCommandFiles() method" on page 144
- "GetNextFOV() method" on page 159
- "GetNextRecords() method" on page 118
- "GetPreviousRecords() method" on page 120
- "GetRules() method" on page 134
- "Move() method" on page 105
- "Move() method" on page 140

- "Move() method" on page 160
- "PositionToFirstRecord() method" on page 121
- "PositionToLastRecord method" on page 122
- "QuiesceService() method" on page 113
- "ResetFOV() method" on page 161
- "ResumeAllocation() method" on page 162
- "ResumeService() method" on page 114
- "Revert() method" on page 130
- "SetDefault() method" on page 141
- "SetPowerState() method" on page 98
- "StartService() method" on page 114
- "StartService() method" on page 151
- "StopService() method" on page 152
- "StopFileSystemCheck() method" on page 115
- "SuspendAllocation() method" on page 162
- "Test() method" on page 147

## Extrinsic method return codes

For easier diagnosis, return codes from method invocations have the same general meanings. Return codes from extrinsic methods have the following meanings:

- 0 - Method completed successfully
- 1 - Not supported
- 2 - Access failed
- 3 - Already defined
- 4 - Command failed
- 5 - In use
- 7 - Insufficient space
- 8 - Integrity lost
- 9 - Name not valid
- 10 - Invalid parameter
- 11 - Invalid size
- 12 - I/O failed
- 13 - Is Default
- 14 - Is referenced
- 15 - Is System
- 18 - Already exists
- 20 - Not attached
- 21 - Not found
- 22 - Not the primary Administrative server
- 23 - Not viable
- 24 - Server timed out
- 25 - Policy bind errors
- 26 - Policy syntax error
- 27 - Is global fileset
- 28 - Storage pool not found

- 30 - Transaction failed
- 32 - Volume in use
- 33 - Volume not found
- 34 - Allocations already suspended
- 35 - Allocations were not suspended
- 36 - Is attached
- 37 - End of iteration
- 38 - Invalid iteration identifier
- 39 - File not found
- 40 - Cannot read file
- 41 - Partial data
- 43 - Directory exists
- 44 - Incompatible operation
- 45 - Server not found
- 46 - Invalid cluster state
- 52 - Disk not viable
- 56 - Access denied
- 57 - No space
- 61 - Cannot connect to server
- 62 - Too many connections
- 63 - Metadata server restart service is already enabled
- 64 - Metadata server restart service is already disabled
- 65 - Server not available
- 66 - Metadata server restart service state cannot continue
- 67 - Cannot become the primary Administrative server
- 68 - Already in progress
- 69 - Up-to-date
- 70 - Servers not the same version
- 71 - RSA unavailable
- 75 - Invalid value
- 76 - Cancel pending
- 77 - Salvage failed

**Related topics:**

- "Extrinsic methods" on page 85

# Chapter 4. Administrative agent object classes

This chapter describes the classes that make up theAdministrative agent object model. The object classes are the building blocks of the Administrative agent and provide functionality to manage SAN File System.

**Related topics:**

## STC_AdminMessageLog

The STC_AdminMessageLog class represents the aggregated, message log file for the Administrative server. This class extends the STC_MessageLog class. It inherits methods from the STC_MessageLog class that enable you to traverse the log and retrieve a specified number of log records.

**Related topics:**

- "STC_MessageLog" on page 117

## STC_AdminProcess

The STC_AdminProcess class represents the long-running administrative commands in the cluster. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_AdminProcess class has the following properties:

*Table 22. STC_AdminProcess class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_Cluster. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| ServiceCreationClassName | string | The class name of the scoping service: STC_MasterService. This property is key. Maximum is 256 characters. |
| ServiceName | string | The instance name of the scoping service. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_AdminProcess. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Id | uint64 | The identifier of the process. This property is key. |
| InstallDate | datetime | The date-and-time timestamp of when the process was started. This property is read-only. |
| Command | string | The command that initiated this process. The Command string contains a command name and a list of parameters. This property is read-only. Maximum is 256 characters. |

## STC_AdminSecurityLog

The STC_AdminSecurityLog class represents the aggregated, security log file for the Administrative server. This class extends the STC_MessageLog class. It inherits methods from the STC_MessageLog class that enable you to traverse the log and retrieve a specified number of log records.

**Related topics:**

- "STC_MessageLog" on page 117

## STC_AdminUser

The STC_AdminUser class represents an authorized user of SAN File System. This class extends the CIM_LogicalElement class.

The Common Information Model Object Model (CIMOM) authenticates a user by comparing a username and password with information stored in the LDAP server. After the CIMOM authenticates a user, it authorizes the user depending on whether the user has the appropriate level of access to perform a requested action. Every property and method of a class has a minimum role needed to get or set a property or invoke a method.

When CIMOM receives an administrator request, it authenticates the user in the LDAP server and extracts the role of a successfully authenticated user. For performance purposes, an authenticated user's role remains validated for a small interval of time. CIMOM does not consult LDAP again within this time window. If the same user makes an administrator request again when the window is open, the CIMOM authenticates the user without consulting the LDAP server.

**Properties:**

The STC_AdminUser class has the following properties:

*Table 23. STC_AdminUser class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_Cluster. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_AdminUser. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | The user name. This property is key. Maximum is 256 characters. |
| EffectiveRole | uint16 | The effective role of the user, as determined by the authentication (CIMOM) service. The direct role of the user or the groups to which the user belongs determine the user's effective role. The strongest role is the effective role. Possible values are:<br>0: Administrator<br>1: Operator<br>2: Backup<br>3: Monitor<br>Any other value indicates that the role is unknown. |
| IsAuthorizationCurrent | boolean | Indicator of whether the role is currently validated. If a user's role was recently validated on the LDAP server, the effective role for the user will remain valid for a short interval of time. If a user makes a request again in this time interval, the LDAP server will not be contacted for authentication. |

*Table 23. STC_AdminUser class properties (continued)*

| Name | Type | Description |
|---|---|---|
| AuthCurrentRemainingTime | uint32 | The remaining time interval after which if the user makes a request, the LDAP server will be contacted again to determine the effective role of a user. The value of this property is valid only if validation is current. Otherwise, the value will be zero. |

**Related topics:**

- "User roles" on page 26
- "Role-based access" on page 51
- "ClearAllCurrentAuthorizations() method"
- "ClearCurrentAuthorization() method"

## ClearAllCurrentAuthorizations() method

You can use the ClearAllCurrentAuthorizations() method to clear all current validation windows.

**Execute Role:** Administrator

**Method Type:** Static

**Return values:**

The ClearAllCurrentAuthorizations() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 21 (Not found)
- 30 (Transaction failed)
- .. (Internal error)

**Related topics:**

- "STC_AdminUser" on page 93

## ClearCurrentAuthorization() method

You can use the ClearCurrentAuthorization() method to clear a user's validation window.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The ClearCurrentAuthorization() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 21 (Not found)
- 30 (Transaction failed)
- .. (Internal error)

**Related topics:**

- "STC_AdminUser" on page 93

## STC_AvailableLUNs

The STC_AvailableLUNs class represents available Fibre Channel logical unit numbers (LUNs). These LUNs are the storage volumes exposed using the small computer system interface (SCSI) LUNs on the engines of the cluster. The STC_AvailableLUNs class extends the CIM_StorageVolume.

**Properties:**

The STC_AvailableLUNs class has the following properties:

*Table 24. STC_AvailableLUNs class properties*

| Name | Type | Description |
|---|---|---|
| LunID | uint64 | The LUN identifier. |
| NodeWWN | string | The engine World-Wide Name (WWN) providing the LUN. A 16-digit hexidecimal number in the form xx:xx:xx:xx:xx:xx:xx:xx. Maximum is 23 characters. |
| PortWWN | string | The port WWN on the engine that is providing the LUN. A 16-digit hexidecimal number in the form xx:xx:xx:xx:xx:xx:xx:xx. Maximum is 23 characters. |
| Vendor | string | The name of the vendor supplying the product. Maximum is 256 characters. |
| Product | string | The product name. Maximum is 256 characters. |
| Version | string | The version of the product. Maximum is 256 characters. |
| Size | uint64 | Storage size, in megabytes, of the LUN. |
| State | uint32 | Indicates the availability state of this LUN as a volume. Possible values are:<br>0: Available - This LUN is available to be added as a volume.<br>1: Assigned - This LUN is already assigned to SAN File System as a volume. The VolumeName property value identifies the specific volume.<br>2: Error - An error occurred determining the properties of the LUN.<br>3: Unknown - The Metadata server is not running. Cannot determine the availability of the LUN.<br>4: Unusable - This LUN is unsuitable as a volume. One reason the LUN is not suitable is that the (inherited) Access property shows the LUN does not support read/write. Other reasons include inconsistent availability of this LUN from all engines of the cluster. |
| VolumeName | string | If the LUN is already assigned to a SAN File System storage pool, this is the volume name that was given when the LUN was added. Otherwise, the value will be null. Maximum is 256 characters. |

**Related topics:**

- "Volumes" on page 27

## STC_Cluster

The STC_Cluster class, along with the STC_MasterService class, provides cluster operations. It extends the CIM_Cluster class.

**Properties:**

The STC_Cluster class has the following properties:

*Table 25. STC_Cluster class properties*

| Name | Type | Description |
|---|---|---|
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_Cluster. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| ClusterId | uint32 | A unique integer identifier for the cluster. |
| ConfiguredNumberOfNodes | uint32 | The number of configured engines in the cluster. |
| CurrentNumberOfNodes | uint32 | Number of engines participating in the cluster. If this number is not the same as the configured number, the cluster is not operating at its full potential. |

**Related topics:**

- "Cluster" on page 6
- "STC_MasterService" on page 110

## STC_ComputerSystem

The STC_ComputerSystem class represents each engine in the cluster. This class extends the CIM_ComputerSystem class.

**Properties:**

The STC_ComputerSystem class has the following properties:

*Table 26. STC_ComputerSystem class properties*

| Name | Type | Description |
|---|---|---|
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_ComputerSystem. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| TotalPowerOnHours | uint64 | Total number of hours the engine has been powered on. This is a read-only counter property. |
| RestartCount | uint16 | Total number of times the engine has been power cycled. This is a read-only counter property. |
| IsPowerOn | boolean | Indicator of whether the engine is powered on. This property is read-only. |

*Table 26. STC_ComputerSystem class properties (continued)*

| Name | Type | Description |
|---|---|---|
| ASMTime | datetime | Current time on the Advanced System Management Processor's local clock. It is the time reference that has to be used to schedule a power off using the SetPowerState() method. This time is independent of the date and time on the server. This property is read-only. |
| CurrentState | uint32 | State of the system. This property is read-only. Possible values are:<br>• 0: Unknown/Power Off<br>• 1: In Power-On Self Test (POST)<br>• 2: Stopped in POST<br>• 3: Booted Flash<br>• 4: Booting operating system<br>• 5: In operating system<br>• 6: CPU is held in reset<br>• 7: Before POST |
| UUID | string | Universal unique identifier of the engine. This string represents a 128-bit number in the form of 32 consecutive hexadecimal numbers with no delimiters. This property is read-only. |

**Related topics:**


- "Engines" on page 11
- "SetPowerState() method" on page 98


## OneButtonDataCollector() method

You can use theOneButtonDataCollector() method to invoke the one-button data collector utility that collects server and system information needed for problem determination. See the *Maintenance and Problem Determination Guide* for details about the information that the utility collects.

By default, the information collects in the /usr/tank/pmf directory. You can specify a different directory using the TANKDIR environment variable. Any stdout and stderr output generated by the utility collects in the /tmp/obdcout file on the local disk of the engine.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The OneButtonDataCollector() method returns one of the following codes:

- 0 (Completed successfully)
- .. (Internal error)

**Related topics:**


- "STC_ComputerSystem" on page 96

## SetPowerState() method

You can use the SetPowerState() method to set the power state of the engine. This class overrides the SetPowerState() method in its parent class, CIM_ComputerSystem. It supports only a limited subset of power setting capabilities that are fully described in the parent class.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 27 describes the parameters you can specify for the SetPowerState() method:

*Table 27. SetPowerState() method parameters*

| Name | Type | Description |
|---|---|---|
| PowerState | uint16 | Input parameter that is the power setting of the engine. Possible values are:<br>• 1: Full Power<br>• 2: Power Save - Low Power Mode<br>• 3: Power Save - Standby<br>• 4: Power Save - Other<br>• 5: Power Cycle<br>• 6: Power Off<br>• 7: Hibernate<br>• 8: Soft Off |
| Time | datetime | Input parameter that is the time that the power setting should occur. If value is zero, the setting occurs immediately. |

**Return values:**

The SetPowerState() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 4 (Command failed)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 71 (RSA unavailable)
- .. (Internal error)

**Related topics:**

- "Engines" on page 11
- "STC_ComputerSystem" on page 96

# STC_Container

The STC_Container class represents a fileset (also known as a container). It extends the CIM_ManagedSystemElement class.

There is an instance of this class for every fileset that exists in a SAN File System. The list of instances and the methods defined in this class are available only on the master Metadata server.

**Properties:**

The STC_Container class has the following properties:

*Table 28. STC_Container class properties*

| Name | Type | Description |
|---|---|---|
| Name | string | Your label for the fileset when you create, move, or rename it. This property is key. Maximum length is 256 characters. |
| Description | string | Your description of the fileset. This property is writeable. Maximum length is 256 characters. |
| InstallDate | datetime | Time when the fileset was created. A lack of a value does not indicate that the fileset does not exist. This property is read-only. |
| State | uint32 | State of the fileset. This property is read-only. Possible values are:<br>0: Detached<br>1: Attached |
| AttachPoint | string | Attach point of this fileset in the file system namespace. An attach point is the combined path formed by the directory path used to attach and the directory name. This property gives the fully qualified directory name of this fileset. It combines the DirectoryPath property with the DirectoryName property. If the fileset is not attached, this value is null. This property is read-only. |
| DirectoryName | string | The name of the fileset as known to the file system. A fileset is made available to the file system using a different name than the Name property, called the directory name. A directory name is attached to an existing directory path that can be another fileset's attach point or a file directory. This name will appear as a directory under the path shown by DirectoryPath property. You specify this name when you create the fileset and can change it by reattaching the fileset. This property is read-only. |
| DirectoryPath | string | The directory path under which a fileset will appear to the file system. The name of the directory is indicated by the DirectoryName property. The fully qualified directory name of this fileset is indicated by the AttachPoint property. This property is read-only. |
| Parent | string | The name of the parent fileset. If this is the global fileset, the value is null. Default is null. This property is read-only. |
| NumberOfChildren | uint32 | Number of immediate child filesets. Default is 0. This property is read-only. |
| Quota | uint32 | Maximum size limit, in megabytes, for the fileset. A value of zero, the default, indicates that there is no limit. The maximum value is 1 024 petabytes. This property is writeable. |

*Table 28. STC_Container class properties (continued)*

| Name | Type | Description |
|---|---|---|
| IsHardQuota | boolean | Indicator of whether a quota limit cannot be extended, which is a hard quota. This property is used when a quota limit exists and the fileset's allocated size reaches the quota limit. If the value is True, the server does not extend the allocated size of the fileset beyond the quota limit. It sends a Severe alert message and logs the message in the server message log. If False, the quota is soft. The server extends the allocated size of the fileset and logs a Warning alert message. This property is writeable. Changing this property from soft quota (False) to hard quota (True) when the fileset has exceeded its quota causes a Hard Quota Violation (72) exception. |
| AlertPercentage | uint16 | Percentage of the fileset size that, when reached, causes the server to generate an alert message. An alert is generated only if all the following conditions are met:<br>• The Quota property value is greater than zero.<br>• An AlertPercentage property value is greater than zero.<br>• The SizeAllocatedPercentage property value equals or exceeds the AlertPercentage property value.<br>This property is writeable. Minimum is 0% and indicates that the server should not generate an alert. Maximum is 100%. The default is 80%. |
| SizeAllocated | uint64 | Size, in megabytes, of the fileset. This size can change as files are added and deleted in the fileset. This property is read-only. |
| SizeAllocatedPercentage | uint16 | Percentage of the size allocated compared to the quota in the pool. This can be compared directly with the AlertPercentage to determine how close the fileset is to causing an alert. This property is read-only. Minimum is 0%. Maximum is 100%. |
| NumberOfPITCopies | uint16 | Number of existing FlashCopy images. A fileset can have as many as 32 read-only FlashCopy images. When an administrator creates a FlashCopy image that causes the maximum number of images to be exceeded, SAN File System deletes the oldest existing image. This property is read-only. |
| LastPITCopyDate | datetime | The datetime value of the last FlashCopy image. This property is read-only. |
| Server | string | The name of the server hosting this fileset. This property is read-only. Maximum is 256 characters. |
| ServerState | uint32 | State of the server serving the fileset. This property is read-only. Possible values are:<br>0: Offline<br>1: Online |

**Related topics:**


- "Filesets" on page 11
- "Attach() method" on page 101
- "ChangeServer() method" on page 101
- "Create() method" on page 102
- "Delete() method" on page 104
- "Detach() method" on page 104
- "Move() method" on page 105

## Attach() method

You can use the Attach() method to attach an existing fileset to a file system namespace. You can also use this method to reattach an attached fileset to a new filesystem namespace. When you reattach a fileset, you can change the directory path as well as the the directory name.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 29 describes the parameters you can specify for the Attach() method:

*Table 29. Attach() method parameters*

| Name | Type | Description |
|---|---|---|
| ExistingDirPath | string | Input parameter that is an existing directory path to attach to. |
| NewDirName | string | Input parameter that is the new directory name of the fileset. |

**Return values:**

The Attach() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 9 (Invalid name)
- 18 (The name given in the NewDirName parameter exists.)
- 21 (Directory path given in ExistingDirPath parameter not found)
- 22 (Not the primary Administrative server)
- 23 (Not viable - For reattach, the new path for the fileset to be attached already contains the original attach point name for the fileset.)
- 30 (Transaction failed)
- 36 (Is already attached)
- 44 (Incompatible operation)
- 61 (Cannot connect to server - The Administrative server could not contact the local Metadata server.)
- 62 (Too many connections)
- 65 (Server state offline)
- .. (Internal error)

**Related topics:**

- "Filesets" on page 11
- "STC_Container" on page 98

## ChangeServer() method

You can use the ChangeServer() method to change the Metadata server hosting the fileset. You can change the server under the following conditions:

- The specified server acting as the new host must be part of the cluster.
- The cluster and the specified server must be either online or in a quiescent state.

- The current host server can be down. If it is not down, it and the cluster must be in a quiescent state.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 30 describes the parameters you can specify for the ChangeServer() method:

*Table 30. ChangeServer() method parameters*

| Name | Type | Description |
|---|---|---|
| Server | string | Input parameter that is the name of the Metadata server to host this fileset. Maximum is 32 characters. |

**Return values:**

The ChangeServer() method returns one of the following codes:
- 0 (Completed successfully)
- 3 (Already defined)
- 8 (Integrity lost)
- 10 (Invalid parameter)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 27 (Is global fileset)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 45 (Server not found)
- 46 (Invalid cluster state)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**
- "Filesets" on page 11
- "Metadata server" on page 18
- "STC_Container" on page 98

## Create() method

You can use the Create() method to define a new fileset. This method is the constructor for the class. Optionally, you can attach the fileset to an attach point by supplying an existing directory path and the new Directory name.

**Execute Role:** Administrator

**Method Type:** Static

**Parameters:**

Table 31 describes the parameters you can specify for the Create() method:

*Table 31. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| Name | string | Input parameter that is your label for the fileset. Maximum length is 256 characters. |
| Description | string | Input parameter that is your description of the fileset. Maximum is 256 characters. |
| Quota | uint64 | Input parameter that is the maximum size limit, in megabytes, for the fileset. A value of zero indicates that there is no limit. The default is no limit. The maximum value is 1024 petabytes. |
| IsHardQuota | boolean | Input parameter that is an indicator of whether a quota limit cannot be extended. |
| AlertPercentage | uint16 | Input parameter that is the percentage of the fileset; size that, when reached, will cause the server to generate an alert message. Minimum is 0% and indicates that the server should not generate an alert. Maximum is 100%. The default is 90%. |
| ExistingDirPath | string | Input parameter that is an existing directory path to attach to. |
| NewDirName | string | Input parameter that is the new directory name to be given to the fileset. |
| Server | string | Input parameter that is the name of the server to host this fileset. Maximum is 32 characters. |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 10 (Invalid parameter)
- 18 (Name exists)
- 21 (Directory path given in ExistingDirPath parameter not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 43 (The name given in the NewDirName parameter exists.)
- 44 (Incompatible operation)
- 45 (Server not found)
- 57 (No space - The Metadata server ran out of space in system volumes where master metadata is stored.
- 61 (Cannot connect to server - The current hosting server and all other parent hosting servers, if any, must be online.)
- 62 (Too many connections)
- 65 (Server state offline)
- .. (Internal error)

**Related topics:**

- "Filesets" on page 11
- "STC_Container" on page 98

## Delete() method

You can use the Delete() method to delete a fileset. You can delete a fileset under the following conditions:

- The fileset is detached.
- The fileset is not the global fileset.
- The fileset does not have files on it unless the IsForce option is set to True.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 32 describes the parameters you can specify for the Delete() method:

*Table 32. Delete() method parameters*

| Name | Type | Description |
|---|---|---|
| IsForce | boolean | Input parameter that indicates whether to delete the fileset even if it has files on it. |

**Return values:**

The Delete method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use - Fileset has files and IsForce is False. )
- 8 (Integrity lost)
- 14 (Is referenced - Fileset is referenced in an active policy rule.)
- 21 (Not found)
- 22 (Not the primary Administrative server )
- 27 (Is global fileset)
- 30 (Transaction failed)
- 30 (Is attached)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline - With or without the force option, the serving server must be online.)
- .. (Internal error)

**Related topics:**

- "Filesets" on page 11
- "STC_Container" on page 98

## Detach() method

You can use the Detach() method to detach a fileset from a file system namespace. You can detach a fileset under the following conditions:

- Fileset does not have any child filesets attached to it.
- No clients are using files on it unless the IsForce option is set to True.

When you detach a fileset, the attach point does not persist. The directory path and directory name are lost.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 33 describes the parameters you can specify for the Detach() method:

*Table 33. Detach() method parameters*

| Name | Type | Description |
|---|---|---|
| IsForce | boolean | Input parameter that indicates whether to detach the fileset even if clients are using files on it. |

**Return values:**

The Detach() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use)
- 8 (Integrity lost)
- 14 (Is referenced — IsForce is False and clients are using files in the fileset)
- 20 (Not attached)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 27 (Is global fileset)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline)
- .. (Internal error)

**Related topics:**

- "Filesets" on page 11
- "STC_Container" on page 98

## Move() method

You can use the Move() method to move or rename a fileset by creating a new fileset with the specified new name and migrating the data and capabilities to the new name. If successful, the old fileset is deleted.

**Execute Role:**Administrator

**Method Type:** Dynamic

**Parameters:**

Table 34 describes the parameters you can specify for the Move() method:

*Table 34. Move() method parameters*

| Name | Type | Description |
|---|---|---|
| NewName | string | Input parameter that is your new label for the fileset. Maximum is 256 characters. |

**Return values:**

The Move() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 9 (Fileset name is not valid.)
- 18 (Fileset name already exists.)
- 21 (Fileset not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Filesets" on page 11
- "STC_Container" on page 98

## STC_MasterDisruptiveSetting

The STC_MasterDisruptiveSetting class represents the parameter settings for cluster configuration that require a cluster restart for an update to take effect. The read-only properties in this class can be set only during installation. If you have Administrator privileges, you can change the writeable properties using the SetProperty() intrinsic method. This class extends the STC_Setting class.

**Properties:**

The STC_MasterDisruptiveSetting class has the following properties:

*Table 35. STC_MasterDisruptiveSetting class properties*

| Name | Type | Description |
|---|---|---|
| ClusterID | uint32 | A unique cluster ID number that is set only at installation time. This property is read-only. The default is the lower 16-bit value of the system time during installation. |
| ClusterName | string | A unique cluster name that is settable only at installation time. This cluster name determines the name of the root directory in the file system namespace. This property is read-only. Maximum length is 32 characters. |
| ClientTimeoutInterval | uint32 | Amount of time, in milliseconds, to wait between attempted message sends from server to client. This property is read-only. Minimum is 200 and maximum is 1000 milliseconds. Default is 500. |

*Table 35. STC_MasterDisruptiveSetting class properties (continued)*

| Name | Type | Description |
|---|---|---|
| ServerTimeoutInterval | uint32 | Amount of time, in milliseconds, to wait between attempted message sends from server to server. This property is read-only. Minimum is 200 and maximum is 1000 milliseconds. Default is 500. |
| DiskHeartbeatInterval | uint32 | Interval, in milliseconds, between heartbeats written to disk. This property is read-only. Minimum is 200 and maximum is 10 000 milliseconds. Default is 500. |
| LogicalPartitionSize | uint32 | Logical partition size in megabytes. This property is read-only. Default is 16 MB. |
| NWHeartbeatInterval | uint32 | Interval, in milliseconds, between the heartbeats over the network. This property is writeable. Minimum is 200 and maximum is 10 000 milliseconds. Default is 500. |
| NWMaxMissedHeartbeats | uint32 | Maximum number of heartbeats that can be missed before the cluster ejects the server. If a server is ejected, it does not need to be recommissioned into the cluster. This property is writeable. Minimum is 1 and maximum is 100. Default is 3. |
| DiskMaxMissedHeartbeats | uint32 | Maximum number of heartbeats that can be missed before the disk ejects a server. If a server is ejected, it does not need to be recommissioned into the cluster. This property is writeable. Minimum is 1 and maximum is 100. Default is 4. |
| LockLeasePeriod | uint32 | Amount of time, in seconds, a lock is leased to a client when the server grants a lock. The server applies a multiplier, specified by the LockGracePeriodMultiplier property, before actually expiring the lease. This property is writeable. Minimum is 10 and maximum is 120. Default is 20 seconds. |
| LockGracePeriodMultiplier | uint32 | The value that the server would multiply times the lock lease period to determine the amount of time to wait before actually expiring a lease of a lock to a client. The server must receive a lock renewal request from the client during this time to keep the lease active. This property is writeable. Minimum is 0 and maximum is 4. Default is 2 seconds. |
| ClusterTimeout | uint32 | Timeout, in microseconds, for communications within the cluster. This property is writeable. Minimum is 500 000 and maximum is 10 000 000. Default is 1 000 000. |
| RetriesToClient | uint32 | Number of times a server attempts to send to a client before declaring the client dead. This property is writeable. Minimum is 1 and maximum is 100. Default is 5. |

**Related topics:**

- "SetProperty()" on page 84
- "STC_Setting" on page 136

## STC_MasterDynamicSetting

The STC_MasterDynamicSetting class represents the parameter settings for cluster configuration that you can dynamically update, without a cluster restart. These parameters persist across cluster restarts. If you have Administrator privileges, you can change the writeable properties in this class using the SetProperty() intrinsic method. This class extends the STC_Setting class.

**Properties:**

The STC_MasterDynamicSetting class has the following properties:

*Table 36. STC_MasterDynamicSetting class properties*

| Name | Type | Description |
|---|---|---|
| MasterBufferSize | uint32 | Buffer cache size for master database space in 4 KB pages. This property is writeable. Minimum is 2 048 and maximum is 8 192 KB. Default is 2 048 KB. |
| SubordinateBufferSize | uint32 | Buffer-cache size for subordinate database space in 4 KB pages. This property is writeable. Minimum is 30 000 and maximum is 250 000 KB. Default is 30 000 KB. |
| SpaceReclaimDelay | uint32 | Interval, in minutes, that the space-reclamation thread waits between runs. A value of zero indicates that space reclamation is disabled. This property is writeable. Minimum is 0 and maximum is 1 440 minutes. Default is 60 minutes. |
| PrivilegedFSClients | string | Comma-separated list of client names for whom administrator privileges are granted in the file system namespace. This property is writeable. |
| SNMPEvents | uint16 | Filter that decides if a Simple Network Management Protocol (SNMP) trap is to be generated when a significant event occurs in a server. You can choose the severity of the events that generate an SNMP trap by setting the corresponding bit in this property. This property is writeable. Possible values are:<br>• 0: Information<br>• 1: Warning<br>• 2: Error<br>• 3: Severe<br>If this value is set to zero (no bits set), SNMP trap generation is disabled. If all the bits are set to one, all event messages generate SNMP traps. |
| SNMPManagers | string | Comma-separated list of destination Internet Protocol (IP) addresses, in dotted decimal format, of the SNMP managers. If an SNMP trap is generated, the trap is sent to this list of managers. This property is writeable. |
| NumAdminThreads | uint32 | Number of threads for administrative operations. This value can only be increased and not decreased. This property is writeable. Minimum is 1 and maximum is 10. Default is 4. |
| NumWorkerThreads | uint32 | Number of threads for general operations. This value can only be increased and not decreased. This property is writeable. Minimum is 10 and maximum is 50. Default is 10. |

**Related topics:**


- "SetProperty()" on page 84
- "STC_Setting" on page 136
- "SNMP" on page 20
- Appendix B, "SNMP Trap MIB", on page 255

## STC_MasterMetrics

The STC_MasterMetrics class represents the metrics for a cluster. Only one instance of this class should exist. This class extends the CIM_ServiceStatisticalInformation class.

The metrics include the current totals for the following types of buffers:

- Clean - Buffers that contain data but are available for reuse.
- Dirty - Buffers that contain data that is awaiting input/output (I/O) to disk.
- Free - Buffers that are available because they are currently not in use.

**Properties:**

The STC_MasterMetrics class has the following properties:

*Table 37. STC_MasterMetrics class properties*

| Name | Type | Description |
|---|---|---|
| TotalSystemMetaActivity | uint64 | Total number of transactions relating to metadata activity for system objects. System objects include storage pools, filesets, volumes, policies, and engines. The activity includes read, create, delete, and modify operations on these objects. This counter property is read-only. |
| TotalSystemMetaUpdateActivity | uint64 | Total number of transactions relating to metadata updates for system objects. This counter property is read-only. |
| TotalSystemBuffers | uint32 | Current number of total buffers for system metadata activity. This property is read-only. |
| CleanSystemBuffers | uint32 | Current number of clean buffers for system metadata activity. Clean buffers contain data but the buffers are available for reuse. This property is read-only. |
| DirtySystemBuffers | uint32 | Current number of dirty buffers for system metadata activity. Dirty buffers contain data awaiting I/O to disk. This property is read-only. |
| FreeSystemBuffers | uint32 | Current number of free buffers for system metadata activity. Free buffers are available because they are currently not in use. This property is read-only. |

**Related topics:**

- "Cluster" on page 6

## STC_MasterSAP

The STC_MasterSAP class represents the service access point of the master Metadata server in a cluster. It extends the STC_TankSAP class.

**Properties:**

The STC_MasterSAP class has the following properties:

*Table 38. STC_MasterSAP class properties*

| Name | Type | Description |
|---|---|---|
| IsLocal | boolean | Indicator of whether the local server is the master Metadata server in the cluster. |

**Related topics:**

- "Metadata server" on page 18
- "STC_TankSAP" on page 148

## STC_MasterService

The STC_MasterService class, along with the STC_Cluster class, provides cluster services. It extends the CIM_ClusteringService class.

**Properties:**

The STC_MasterService class has the following properties:

*Table 39. STC_MasterService class properties*

| Name | Type | Description |
|---|---|---|
| CurrentState | uint32 | Indicates the state of the cluster. This property is read-only. Possible values are:<br>0: Down<br>1: Online<br>2: Partly Quiescent - Only Metadata server I/O operations are suspended.<br>3: Fully Quiescent - All background I/O, client, and Metadata server operations are suspended.<br>4: Administrative Quiescent - No longer servicing clients.<br>5: Forming a cluster<br>6: Not the master Metadata server anymore<br>7: Unknown - Master Metadata server could not be contacted to determine the cluster state. The probable reasons for this would be that the master is down or a network partition. |
| PendingState | uint32 | The current state of the cluster transitions to this state if the current state is different from this pending state. This property is read-only. Possible values are the same as the CurrentState property. |
| LastCurrentStateChangeTime | datetime | Time passed since the cluster changed its current state. This property is read-only. |
| LastPendingStateChangeTime | datetime | Time passed since the cluster has had a state change pending. This property is read-only. |
| CommittedVersion | string | Committed software release version. This property is read-only. |
| CommittedUpgradeTimestamp | datetime | Timestamp when the latest upgrade was committed. This property is read-only. |
| CurrentVersion | string | Current software release version. This version will be different from the previous (committed) version if there was an upgrade done but the commit action was not yet activated. This property is read-only. |
| IsUpgradeInProgress | boolean | Indicator of whether an upgrade is in progress. After a version is committed, the system might take some time to sync up all the internal data structure versions. This property indicates if such a change is still in progress. This property is read-only. |

**Related topics:**

- "Cluster" on page 6
- "STC_Cluster" on page 96
- "CommitUpgrade() method"
- "FileSystemCheck() method"
- "QuiesceService() method" on page 113
- "ResumeService() method" on page 114
- "StartService() method" on page 114
- "StopFileSystemCheck() method" on page 115
- "StopService() method" on page 116

## CommitUpgrade() method

You can use the CommitUpgrade() method to commit the cluster to start using an upgraded software version level. All the Metadata servers in the cluster must first be upgraded to this version level.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The CommitUpgrade() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 68 (Already in progress - Another commit is already in progress.)
- 69 (Up-to-date - The committed version is the same as the software version of all the servers.)
- 70 (All servers are not at the same version - All the servers in the cluster are not at the same software version.)
- .. (Internal error)

**Related topics:**

- "Cluster" on page 6
- "STC_MasterService" on page 110

## FileSystemCheck() method

You can use the FileSystemCheck() method to check and repair metadata. It provides options for you to:

- Check the integrity of the structure and the content of the metadata.
- Check the integrity of the system metadata and the user (fileset) metadata.
- Limit the user metadata checking to a subset of filesets.

You can restrict this operation to check-only or check and repair. The message log contains a report generated by this method. If you did not limit the mode to check-only, the system automatically salvages and repairs the damaged data if

possible. Some types of repair require manual intervention from the administrator. In those cases, the cluster state is placed in Administrative mode. You can invoke the StopFileSystemCheck() method to stop a check and repair of metadata.

**Notes:**

1. This method is a long-running process. If there is a cluster reformation while the method is running, this method might stop.
2. Only one FileSystemCheck() operation can be in progress at time.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 40 describes the parameters you can specify for the FileSystemCheck() method:

*Table 40. FileSystemCheck() method parameters*

| Name | Type | Description |
|---|---|---|
| IsCheckOnly | boolean | This input parameter is an indicator of whether to only check and not repair. |
| CheckScope | uint16 | This input parameter is a bitmap indicating the scope of the check. Possible values are:<br>0: Structure - Checks the structure of the metadata.<br>1: Content - Checks the contents of the metadata.<br>You can set both bits to check the structure and content. |
| Type | uint16 | This input parameter is a bitmap indicating the type of the metadata to be checked. Possible values are:<br>0: System - Checks the system metadata.<br>1: User - Checks the user (fileset) metadata.<br>You can set both bits to check the system and user metadata. |
| ContainerList | string[ ] | This input parameter is a list of filesets to be checked or repaired if the Type parameter is set to User and not System. |

**Return values:**

The FileSystemCheck() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use - Another FileSystemCheck operation is active.)
- 8 (Integrity lost - Check-only option was chosen and the server detects corruption.)
- 10 (Invalid Parameter)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 60 (Canceled - Due to StopFileSystemCheck method.)
- 61 (Cannot connect to server)
- 62 (Too many connections)

- 73 (Salvaged - Repair was requested. Corruption was detected and repaired successfully. This return code indicates success, not an error. )
- 76 (Cancel pending - A FileSystemCheck() method is active with a cancel pending. Make sure that the previous FileSystemCheck() method has stopped completely before issuing another check.)
- 77 (Salvage failed - A repair attempt failed. You can look in the server log files for details. Contact technical support and initiate metadata and data recovery actions.)
- .. (Internal error)

**Related topics:**

- "StopFileSystemCheck() method" on page 115

## QuiesceService() method

You can use the QuiesceService() method to place the cluster in a quiescent state to perform some backup-and-restore and administrative operations. All the servers that currently belong to the cluster are brought into a quiescent state. A cluster might temporarily leave the quiescent state if a server leaves or joins the cluster. You can use the ResumeService() method to return a cluster to a fully online state from a quiescent state.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 41 describes the parameters you can specify for the QuiesceService() method:

*Table 41. QuiesceService() method parameters*

| Name | Type | Description |
|---|---|---|
| Mode | uint32 | This input parameter is the quiescent state. Possible values are:<br>0: Partly Quiescent - A limited quiescent mode that allows client file data activity to continue but prevents client metadata activity and new client connections. This state allows a backup with metadata integrity but might not preserve file data integrity.<br>1: Fully Quiescent - A full quiescent mode that suspends all client metadata activity and file data activity and terminates all client sessions. This state allows a backup with metadata and file data integrity.<br>2: Administrative Quiescent -Administrative operations that do not permit client activity can be performed safely. |

**Return values:**

The QuiesceService() method returns one of the following codes:

- 0 (Completed successfully.)
- 1 (Not Supported)
- 4 (Command Failed)
- 8 (Integrity lost)
- 10 (Invalid Parameter)

- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**
- "Cluster" on page 6
- "STC_MasterService" on page 110
- "ResumeService() method"

## ResumeService() method

You can use the ResumeService() method to return the cluster to a fully online state from the quiescent state.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The ResumeService() method returns one of the following codes:
- 0 (Completed successfully)
- 1 (Not supported)
- 4 (Command failed)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**
- "Cluster" on page 6
- "STC_MasterService" on page 110

## StartService() method

You can use the StartService() method to bring up all precommissioned servers on all engines. This method starts the master Metadata server, checks that the master Metadata server is online and then starts all subordinate servers.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The StartService() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 4 (Command failed)
- 5 (In use - The cluster is running already. The master Metadata server is running.)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 24 (Server timed out - The Metadata server server in the cluster was launched successfully but failed to come online after a maximum wait period.)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 66 (Metadata server restart service state cannot continue - This is only a warning; the cluster is started successfully.)
- 89 (Server exited - A server in the cluster exited prematurely.)
- .. (Internal error)

**Related topics:**

- "Cluster" on page 6
- "STC_MasterService" on page 110

## StopFileSystemCheck() method

You can use the StopFileSystemCheck() method to stop a FileSystemCheck() method that is in progress.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return Values:**

The StopFileSystemCheck() method returns one of the following codes:

- 0 (Completed successfully. - The current FileSystemCheck() method is marked for cancellation.)
- 8 (Integrity lost)
- 21 (Not Found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 76 (Cancel pending -A cancel has been issued using StopFileSystemCheck, but the cancel is still pending. A check operation might not stop immediately after you invoke the StopFileSystemCheck() method.
- .. (Internal error)

**Related topics:**

- "FileSystemCheck() method" on page 111

## StopService() method

You can use the StopService() method to gracefully bring down servers on all engines of a cluster.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The StopService() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 4 (Command failed)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 24 (Server timed out)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 66 (Metadata server restart service state cannot continue)
- .. (Internal error)

**Related topics:**

- "Cluster" on page 6
- "STC_MasterService" on page 110

# STC_MDSAuditLog

The STC_MDSAuditLog class represents the aggregated, audit log file for a Metadata server. This class extends the STC_MessageLog class. It inherits methods from the STC_MessageLog class that enable you to traverse the log and retrieve a specified number of log records.

**Properties:**

The STC_MDSAuditLog class has the following properties:

*Table 42. STC_MDSAuditLog class properties*

| Name | Type | Description |
|---|---|---|
| BackupLogFile Name | string | The absolute path and name of the backup log file. This is consistent across all engines in the cluster. |

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## STC_MDSEventLog

The STC_MDSEventLog class represents the event log file for a Metadata server. This class extends the STC_MessageLog class.

Though the event log file is represented as a separate file, the event log is primarily a filter that groups all event records from the STC_MDSMessageLog.

**Properties:**

The STC_MDSEventLog class has the following properties:

*Table 43. STC_MDSEventLog class properties*

| Name | Type | Description |
|---|---|---|
| BackupLogFile Name | string | The absolute path and name of the backup log file. This is consistent across all engines in the cluster. |

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog"

## STC_MDSMessageLog

The STC_MDSMessageLog class represents the aggregated, message log file for a Metadata server. This class extends the STC_MessageLog class. It inherits methods from the STC_MessageLog class that enable you to traverse the log and retrieve a specified number of log records.

**Properties:**

The STC_MDSMessageLog class has the following properties:

*Table 44. STC_MDSMessageLog class properties*

| Name | Type | Description |
|---|---|---|
| BackupLogFile Name | string | The absolute path and name of the backup log file. This is consistent across all engines in the cluster. |

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog"

## STC_MessageLog

The STC_MessageLog class represents log files that are present in SAN File System. This class extends the CIM_MessageLog class.

The STC_MessageLog class enhances the iterator methods of traversing the log file as defined in CIM_MessageLog class in the following ways:

- It has a PositionToLastRecord() method in addition to the PositionToFirstRecord() method.

- The GetNextRecords() and GetPreviousRecords() methods set the direction of traversal from a given iterator and return one or more log records using a set of array output parameters. The parameter values at a given index into this array constitute a log record.

**Properties:**

The STC_MessageLog class has the following properties:

*Table 45. STC_MessageLog class properties*

| Name | Type | Description |
|---|---|---|
| LogFileName | string | The absolute path and name of the log file. This is consistent across all engines in the cluster. |

**Related topics:**

- "Logs" on page 16
- "ClearLog() method"
- "GetNextRecords() method"
- "GetPreviousRecords() method" on page 120
- "PositionToFirstRecord() method" on page 121
- "PositionToLastRecord method" on page 122

## ClearLog() method

You can use the ClearLog() method to clear a message or audit log of all entries.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The ClearLog() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported. Check that the Capabilities property defined in the parent class specifies that the message log can be cleared.)
- 4 (Command failed - The clear operation failed on the Metadata server)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 41 (Partial data - The clear operation partially cleared the log.)
- 61 (Cannot connect to Metadata server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## GetNextRecords() method

You can use the GetNextRecords() method to retrieve a specified number of records from a message log, starting from the record indicated by the

IterationIdentifier parameter. After the method retrieves the records, it advances the IterationIdentifier parameter to the record after the last record returned. If the traversal reaches the last record in the file, the method returns an End of Iteration return code. Subsequent calls to this method might return new records if they have been written to the log.

This method can return one or more log records. It uses a set of array output parameters to represent the log records. The parameter values at a given index into this array constitute a log record. These parameter values provide information about the message as well as its content.

**Execute Role:** Monitor

**Method Type:** Dynamic

**Parameters:**

Table 46 describes the parameters you can specify for the Attach() method:

*Table 46. GetNextRecords() method parameters*

| Name | Type | Description |
|---|---|---|
| IterationIdentifier | string | Input/Output parameter that is an identifier for the iterator. Maximum is 100 characters. |
| NumberOfEntries | uint32 | Input/Output parameter that indicates the number of records to be retrieved and returns the actual number of log records that were retrieved. |
| MessageTimestamp | datetime[] | Output parameter with the timestamp for the message. |
| MessageID | string[] | Output parameter with the identifier for the message. |
| MessageType | uint8[] | Output parameter with the type of message. Possible values are:<br>• 1: Normal<br>• 2: Event<br>• 3: Audit<br>• 4: Trace |
| SourceNode | string[] | Output parameter with the identifier of the engine that originated the message. |
| Severity | uint8[] | Output parameter with the severity of the message. Possible values are:<br>• 0: Information<br>• 1: Warning<br>• 2: Error<br>• 3: Severe |
| MessageString | string[] | Output parameter with the content of the message. |

**Return values:**

The GetNextRecords() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 8 (Integrity lost)
- 10 (Invalid parameter)

- 30 (Transaction failed)
- 37 (End of iteration)
- 38 (Invalid IterationIdentifier)
- 39 (File not found)
- 40 (Cannot read file)
- 41 (Partial data)
- .. (Internal error)

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## GetPreviousRecords() method

You can use the GetPreviousRecords() method to retrieve a specified number of records from a message log, ending at the record indicated by the IterationIdentifier parameter. After the method retrieves the records, it positions the IterationIdentifier parameter to the record before the first record returned. If the traversal reaches the first record in the file, the method returns an End of Iteration return code. Subsequent calls to this method will have no effect.

This method can return one or more log records. It uses a set of array output parameters to represent the log records. The parameter values at a given index into this array constitute a log record. These parameter values provide information about the message as well as its content.

**Execute Role:** Monitor

**Method Type:** Dynamic

**Parameters:**

Table 47 describes the parameters you can specify for the GetPreviousRecords() method:

*Table 47. GetPreviousRecords() method parameters*

| Name | Type | Description |
|---|---|---|
| IterationIdentifier | string | Output parameter that is an identifier for the iterator. Maximum is 100 characters. |
| NumberOfEntries | uint32 | Input/Output parameter that indicates the number of records to be retrieved and returns the actual number of log records that were retrieved. |
| MessageTimestamp | datetime[] | Output parameter with the timestamp for the message. |
| MessageID | string[] | Output parameter with the identifier for the message. |
| MessageType | uint8[] | Output parameter with the type of message. Possible values are:<br>• 1: Normal<br>• 2: Event<br>• 3: Audit<br>• 4: Trace |
| SourceNode | string[] | Output parameter with the identifier of the engine that originated the message. |

*Table 47. GetPreviousRecords() method parameters (continued)*

| Name | Type | Description |
|---|---|---|
| Severity | uint8[] | Output parameter with the severity of the message. Possible values are:<br>• 0: Information<br>• 1: Warning<br>• 2: Error<br>• 3: Severe |
| MessageString | string[] | Output parameter with the content of the message. |

**Return values:**

The GetPreviousRecords() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 8 (Integrity lost)
- 10 (Invalid parameter)
- 30 (Transaction failed)
- 37 (End of iteration)
- 38 (Invalid IterationIdentifier)
- 39 (File not found)
- 40 (Cannot read file)
- 41 (Partial data)
- .. (Internal error)

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## PositionToFirstRecord() method

You can use the PositionToFirstRecord() method to establish an iteration of a message log and set the iterator to the first entry in the log. An identifier for the iterator is returned as an output parameter.

**Execute Role:** Monitor

**Method Type:** Dynamic

**Parameters:**

Table 48 describes the parameters you can specify for the PositionToFirstRecord() method:

*Table 48. PositionToFirstRecord() method parameters*

| Name | Type | Description |
|---|---|---|
| IterationIdentifier | string | Output parameter that is an identifier for the iterator. |

**Return values:**

The PositionToFirstRecord() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 39 (File not found)
- 40 (Cannot read file)
- 41 (Partial data)
- .. (Internal error)

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## PositionToLastRecord method

You can use the PositionToLastRecord() method to establish an iteration of a message log and set the iterator to the last entry in the log. An identifier for the iterator is returned as an output parameter.

**Execute Role:** Monitor

**Method Type:** Dynamic

**Parameters:**

Table 49 describes the parameters you can specify for the PositionToLastRecord() method:

*Table 49. PositionToLastRecord() method parameters*

| Name | Type | Description |
|---|---|---|
| IterationIdentifier | string | Output parameter that is an identifier for the iterator. |

**Return values:**

The PositionToLastRecord() method returns one of the following codes:

- 0 (Completed successfully)
- 1 (Not supported)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 39 (File not found)
- 40 (Cannot read file)
- 41 (Partial data)
- .. (Internal error)

**Related topics:**

- "Logs" on page 16
- "STC_MessageLog" on page 117

## STC_NodeFan

The STC_NodeFan class represents status of a engine's fan. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_NodeFan class has the following properties:

*Table 50. STC_NodeFan class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreation ClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassN ame | string | The name of the class or the subclass used in the creation of an instance: STC_NodeFan. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| DeviceID | string. | An address or other identifying information to uniquely name the specific fan. This property is key. Maximum is 64 characters. |
| Speed | uint32 | Speed of the fan in percentage of the optimal speed of 100%. |

**Related topics:**

- "Engines" on page 11

## STC_NodeTemperature

The STC_NodeTemperature class represents the temperature state of hardware components of an engine, as reported by the Advanced System Management Processor. An instance of this class exists for each temperature sensor available on every engine of a cluster. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_NodeTemperature class has the following properties:

*Table 51. STC_NodeTemperature class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreation ClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassN ame | string | The name of the class or the subclass used in the creation of an instance: STC_NodeTemperature. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| DeviceID | string | An address or other identifying information to uniquely name the temperature sensor, for example, CPU temperature. This property is key. Maximum is 64 characters. |

*Table 51. STC_NodeTemperature class properties (continued)*

| Name | Type | Description |
|---|---|---|
| Value | real32 | Current temperature, in degrees Celsius, of this hardware component. This property is read-only. |
| HasThresholds | boolean | Indicator of whether thresholds are available for this hardware component. This property is read-only. |
| WarningReset | real32 | Temperature threshold value for warning reset. If the temperature exceeds the Warning property value and then drops below this value, the Advanced System Management Processor clears any active temperature events. A value of zero means that this threshold is disabled. This property is read-only. |
| Warning | real32 | Temperature threshold value for warning. If the temperature reaches this value, the Advanced System Management Processor generates a warning event. A value of zero means that this threshold is disabled. This property is read-only. |
| SoftShutdown | real32 | Temperature threshold value for soft shutdown. If the temperature reaches this value, a critical event is generated and the server is powered off after the operating system is shut down. A value of zero means that this threshold is disabled. This property is read-only. |
| HardShutdown | real32 | Temperature threshold value for hard shutdown. If the temperature reaches this value, a critical event is generated and the server is powered off immediately. A value of zero means that this threshold is disabled. This property is read-only. |

**Related topics:**


- "Engines" on page 11


## STC_NodeVitalProductData

The STC_NodeVitalProductData class represents vital product data about the components of a engine. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_NodeVitalProductData class has the following properties:

*Table 52. STC_NodeVitalProductData class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_NodeVitalProductData. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| DeviceID | string | An address or other identifying information to uniquely name the logical device. This property is key. Maximum is 64 characters. |

*Table 52. STC_NodeVitalProductData class properties (continued)*

| Name | Type | Description |
|---|---|---|
| MachineModel | string | Model identifier of the logical device's host machine. This property is read-only. |
| SerialNumber | string | Serial number of the logical device's host machine. This property is read-only. |
| Revision | string | Firmware revision of the logical device. This property is read-only. |
| RevisionDate | datetime | Firmware revision date of the logical device. This property is read-only. |
| FirmwareFileName | string | Firmware file name of the logical device. This property is read-only. |
| FirmwareBuildID | string | Firmware build ID of the logical device. This property is read-only. |

**Related topics:**


- "Engines" on page 11


## STC_NodeVoltage

The STC_NodeVoltage class represents the state of the voltage sources of a engine, as reported by the Advanced System Management Processor. There is an instance of this class for each voltage source available on every engine of a cluster. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_NodeVoltage class has the following properties:

*Table 53. STC_NodeVoltage class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_NodeVoltage. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| DeviceID | string | An address or other identifying information to uniquely name the logical device. This property is key. Maximum is 64 characters. |
| CurrentVoltage | real32 | Current voltage of a voltage source line on this engine . |
| DefaultVoltage | real32 | Default voltage of a voltage source line on this engine. |
| HasThresholds | boolean | Indicator of whether thresholds are available for this device. |
| WarningLow | real32 | Low value for a warning on this voltage line. If the voltage drops below this value, a warning event is generated. A value of -99.99 means that this threshold is disabled. |

*Table 53. STC_NodeVoltage class properties (continued)*

| Name | Type | Description |
|---|---|---|
| WarningHigh | real32 | High value for a warning on this voltage line. If the voltage rises above this value, a warning event is generated. A value of -99.99 means that this threshold is disabled. |

**Related topics:**

- "Engines" on page 11

## STC_NodeWatchdog

The STC_NodeWatchdog class represents the settings for an Advanced System Management Processor watchdog. There is an instance of this class for each engine in the cluster. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_NodeWatchdog class has the following properties:

*Table 54. STC_NodeWatchdog class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_NodeWatchdog. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | An address or other identifying information to uniquely name the watchdog. This property is key. Maximum is 64 characters. |
| POSTTimeout | uint32 | Watchdog timeout value for the the Power-On Self Test (POST). The POST watchdog is active once when the power is coming up. If the engine fails to complete POST within the time indicated by this timeout value, the Advanced System Management Processor generates a POST timeout alert and automatically restarts the system once. When the system restarts, the POST watchdog is automatically disabled until the operating system is shut down and the server is power cycled. A value of zero indicates that this watchdog is disabled. |
| OSMonitorInterval | uint32 | The frequency, in seconds, that the Advanced System Management Processor checks that the operating system is running properly. The operating system watchdog checks the state of the operating system at periodic intervals of time. A value of zero indicates that this watchdog is disabled. |
| OSTimeout | uint32 | Watchdog timeout value, in seconds, for the operating system. If the operating system fails to respond to these checks within this timeout value, the Advanced System Management Processor generates an operating system Timeout alert and automatically restarts the system. When the operating system restarts, the operating system watchdog is automatically disabled until the operating system is shut down and the server is power cycled. |

Table 54. STC_NodeWatchdog class properties (continued)

| Name | Type | Description |
|---|---|---|
| LoaderTimeout | uint32 | Watchdog timeout value, in seconds, for the operating system boot process or loader. The timeout value indicates the amount of time the Advanced System Management Processor waits between the completion of POST and the end of loading the operating system. If the interval is exceeded, the Advanced System Management Processor generates a Loader Timeout alert and automatically restarts the system once. When the system restarts, the Loader Timeout is disabled until the operating system is shut down and the server is power cycled. A value of zero indicates that this watchdog is disabled. |
| PowerOffDelay | uint32 | Amount of time, in seconds, the Advanced System Management Processor waits for the operating system to shut down before powering off the system. |

**Related topics:**


- "Engines" on page 11


## STC_PitImage

The STC_PitImage class represents the FlashCopy images (also known as point-in-time images) of a fileset. This class extends the CIM_ManagedSystemElement class.

There is an instance of this class for every FlashCopy images of the filesets that exists in a SAN File System. The space used by a FlashCopy image is accounted for in the space used by a container for quota calculations. When a FlashCopy image is created, it does not use any space. FlashCopy images use space when files within the fileset are modified after a FlashCopy image is taken.

**Properties:**

The STC_PitImage class has the following properties:

Table 55. STC_PitImage class properties

| Name | Type | Description |
|---|---|---|
| Caption | string | One-line description of the object. This property is read-only. Maximum length is 64 characters. |
| ContainerName | string | Your label for the fileset to which this FlashCopy image belongs. This property is key. Maximum length is 256 characters. |
| Name | string | Your administrative name for the FlashCopy image. This property is key. Maximum length is 256 characters. |
| Description | string | Your description of the fileset. Maximum length is 256 characters. |
| InstallDate | datetime | Time when the the FlashCopy image was created. A lack of a value does not indicate that the fileset does not exist. The alias is CreationDate. |
| DirectoryName | string | The directory name containing this FlashCopy image. The full path for the FlashCopy image in the file system is given by AttachPoint/.pit/DirectoryName where AttachPoint is the attach point of the fileset. |

**Related topics:**

- "FlashCopy images" on page 13
- "Create() method"
- "Delete() method" on page 129
- "Revert() method" on page 130

## Create() method

You can use the Create() method to create a new FlashCopy image for a fileset. This method is the constructor for this class. When you create a FlashCopy image, the fileset can be attached or detached. Thirty-two FlashCopy images can exist at any given time. When this limit is reached, a Create operation fails unless the IsForce parameter is set to True. In this case, the oldest FlashCopy image is deleted so the new one can be created.

**Execute Role:** Backup

**Method Type:** Static

**Parameters:**

Table 56 describes the parameters you can specify for the Create() method:

*Table 56. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| ContainerName | string | Input parameter that is your label for the fileset to which this FlashCopy image belongs. Maximum length is 256 characters. |
| Name | string | Input parameter that is your administrative name for the FlashCopy image. Maximum length is 256 characters. |
| Description | string | Input parameter that is your description of the fileset. Maximum length is 256 characters. |
| DirectoryName | string | Input parameter that is the new directory name to be given to the FlashCopy image. |
| IsForce | boolean | Input parameter that indicates whether to delete the oldest FlashCopy image copy to create this one when the limit is reached. |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 10 (Invalid parameter - The FlashCopy image name length is greater than the maximum, or Description length is greater than the maximum, or DirectoryName length is greater than the maximum, or the DirectoryName contains directory separators.)
- 18 (FlashCopy image name already exists for the fileset.)
- 21 (Fileset not found or no server serving the fileset.)
- 22 (Not the primary Administrative server)
- 30 (Transaction Failed - Other concurrent activity in the server caused this create operation to fail.)

- 42 (Table full - The number of FlashCopy images taken are at the maximum limit already.)
- 43 (Directory name already exists for another FlashCopy image for the same fileset.)
- 44 (Incompatible operation - The server is executing an incompatible operation to this Create.)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline - Any operation on a FlashCopy image needs the server serving the fileset to be online.)
- .. (Internal error)

**Related topics:**

- "FlashCopy images" on page 13
- "STC_PitImage" on page 127

## Delete() method

You can use the Delete method to delete a FlashCopy image of a fileset. You cannot delete a FlashCopy image that has client activity (session locks open) unless the IsForce parameter is set to True. If the IsForce parameter is set to True, all the client activity is terminated (session locks revoked) before the delete.

**Execute Role:** Backup

**Method Type:** Dynamic

**Parameters:**

Table 57 describes the parameters you can specify for the Delete() method:

*Table 57. Delete method parameters*

| Name | Type | Description |
|---|---|---|
| IsForce | boolean | Input parameter that indicates whether to delete the fileset even if client activity exists. |

**Return values:**

The Delete() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use - Client session locks are open and IsForce option is False.)
- 8 (Integrity lost)
- 21 (Not found - FlashCopy image not found or fileset not found, or no server serving the fileset.)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline - Any operation on a FlashCopy image needs the server serving the fileset to be online.)

- .. (Internal error)

**Related topics:**

- "FlashCopy images" on page 13
- "STC_PitImage" on page 127

## Revert() method

You can use the Revert() method to revert a fileset to this instance of the FlashCopy image. You cannot revert a fileset that has children filesets. Detach children filesets, if any, manually.

A Revert operation deletes all the FlashCopy images that are more recent than this instance, including the current fileset image. If the IsForce parameter is False, you cannot revert a fileset to this instance under any of the following conditions:

- Any client activity exists (session locks open) in the FlashCopy images (including the current fileset) to be deleted.
- Any client activity exists (session locks open) in the current PIT instance.

In other words, the client activity can continue only in FlashCopy images taken earlier than this instance.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 58 describes the parameters you can specify for the Revert() method:

*Table 58. Revert() method parameters*

| Name | Type | Description |
|---|---|---|
| IsForce | boolean | Input parameter that indicates whether to revert a fileset to this instance even if client activity exists. |

**Return values:**

The Revert() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use - Client session locks are open and IsForce option is False.)
- 8 (Integrity lost)
- 21 (Not found - FlashCopy image not found or fileset not found, or no server serving the fileset.)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 36 (Is attached - Fileset has child filesets.)
- 41 (Partial data - FlashCopy image contains incomplete files so the fileset is not reverted.)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)

- 65 (Server state offline - Any operation on a FlashCopy image needs the server serving the fileset to be online.)
- .. (Internal error)

**Related topics:**

- "FlashCopy images" on page 13
- "STC_PitImage" on page 127

## STC_PolicySet

The STC_PolicySet class represents a policy, which is a list of file-placement and service-class rules that define characteristics and placement of files. It extends the CIM_PolicySet class.

There is an instance of this class for every policy that exists in a SAN File System. Although multiple policies can exist in the system, only one policy can be active. The system defines a default policy set that assigns files to the Default storage pool.

**Properties:**

The STC_PolicySet class has the following properties:

*Table 59. STC_PolicySet class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_Cluster. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_PolicySet. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | A label for this policy. This property is key. Maximum is 256 characters. |
| State | uint16 | Indicates whether or not this policy is administratively active. Only one policy can be active at any time. Possible values are:<br>0: Not Active<br>1: Active<br>The default value is 0. |
| PolicyRules | string | The set of policy rules belonging to this policy. |
| Description | string | Your description of this policy. Maximum is 256 characters. |
| CreationDate | datetime | Date and time when this policy was created. |
| LastModificationDate | datetime | Date and time when the rules in this policy were last modified. If the rules were never modified, this value will be the same as the creation date. The policy rules can be modified as a whole by creating a policy with the same name and using the IsForce option. |

*Table 59. STC_PolicySet class properties (continued)*

| Name | Type | Description |
|---|---|---|
| LastActiveDate | datetime | Date and time when this policy was last active. This is actually the date and time when another policy was made active (enabled) instead of this one. If the policy was never activated or is currently active, the value is null. |

**Related topics:**

- "Policies and rules" on page 24
- "Activate() method"
- "Create() method" on page 133
- "Delete() method" on page 134
- "GetRules() method" on page 134

## Activate() method

You can use the Activate() method to activate a stored policy.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Properties:**

Table 60 describes the parameters you can specify for the Activate() method:

*Table 60. Activate() method parameters*

| Name | Type | Description |
|---|---|---|
| Errors | string[] | Output parameter that contains information about a policy bind error if one occurred. |

**Return values:**

The Activate() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 25 (Policy bind errors - See the Errors output parameter for more information about this error.)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Policies and rules" on page 24
- "STC_PolicySet" on page 131

## Create() method

You can use the Create() method to create a new policy. This method is the constructor for the class.

**Execute Role:** Administrator

**Method Type:** Static

**Properties:**

Table 61 describes the parameters you can specify for the Create() method:

*Table 61. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| Name | string | Input parameter that is a label for this policy. Maximum is 256 characters. |
| Description | string | Input parameter that is your description of this policy. Maximum is 256 characters. |
| PolicyRules | string | Input parameter that is the set of policy rules belonging to this policy. |
| IsForce | boolean | Input parameter that indicates whether an existing policy with the same name will be overwritten by this policy. |
| Errors | string [] | Output parameter that contains information about a policy syntax error if one occurred. |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)
- 3 (Already defined - Another policy with the same name exists and the IsForce flag is False. )
- 5 (In use)
- 8 (Integrity lost)
- 9 (Invalid name - The name has invalid characters)
- 13 (Is default - The name is DEFAULT_POLICY and the IsForce flag is True. Cannot overwrite the default policy.)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 26 (Policy syntax error - See the Errors output parameter for more information about this error including its location.)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Policies and rules" on page 24
- "STC_PolicySet" on page 131

## Delete() method

You can use the Delete() method to delete an existing stored policy. The policy must be inactive for delete to succeed.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The Delete() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (In use)
- 8 (Integrity lost)
- 13 (Is Default - The name is DEFAULT_POLICY and the IsForce flag is True. Cannot delete default policy.)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Policies and rules" on page 24
- "STC_PolicySet" on page 131

## GetRules() method

You can use the GetRules() method to retrieve the set of rules associated with this policy.

**Execute Role:** Monitor

**Method Type:** Dynamic

**Parameters:**

Table 62 describes the parameters you can specify for the GetRules() method:

*Table 62. GetRules() method parameters*

| Name | Type | Description |
|---|---|---|
| RulesList | string | This output parameter is the set of policy rules belonging to this policy. |

**Return values:**

The GetRules() method returns one of the following codes:

- 0 (Completed successfully)

- 8 (Integrity lost)
- 21 (Not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**


- "Policies and rules" on page 24
- "STC_PolicySet" on page 131


## STC_RegisteredFSClients

The STC_RegisteredFSClients class represents the registered clients of a Metadata server. Every client-server registration pair is unique. The same client that registers to two servers appears as two instances. This class extends the CIM_LogicalElement class.

**Properties:**

The STC_RegisteredFSClients class has the following properties:

*Table 63. STC_RegisteredFSClients class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_ComputerSystem. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| ServiceCreationClassName | string | The class name of the scoping service: STC_TankService. This property is key. Maximum is 256 characters. |
| ServiceName | string | The instance name of the scoping service. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_RegisteredFSClients. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | The name of the client. Maximum is 256 characters. |
| Id | uint64 | The ID of the client. |
| IPAddress | string | The IP network address of the client. |
| IPPort | uint32 | The IP network port address of the client. |
| Platform | string | The operating system platform of the client. |
| Version | string | The SAN File System version of the client. |
| LeaseRenewals | uint64 | Total number of times the client has renewed a lease. This property indicates a measure of how long the client was active. |

*Table 63. STC_RegisteredFSClients class properties (continued)*

| Name | Type | Description |
|---|---|---|
| State | uint16 | The state of the client's lease. Possible values are:<br>• 0: Expired lease<br>• 1: Valid lease |
| IsPrivileged | boolean | Indicator of whether this SAN File System client has Administrator privileges to the SAN File System namespace. |
| LastLeaseTimeStamp | datetime | The timestamp when the server issued or extended the lease. |
| ResidualLeaseTime | uint32 | Countdown timer, in seconds, indicating how long the current lease will last. The lease time is two times the lease renewal interval, a configurable parameter, beginning from the LastLeaseTimeStamp property value. |
| Transactions | uint64 | Total number of transactions started by this client. |
| CompletedTransactions | uint64 | Total number of transactions completed. |
| SessionLocks | uint32 | Current number of session locks this client is holding. |
| DataLocks | uint32 | Current number of data locks this client is holding. |
| ByteRangeLocks | uint32 | Current number of byte range locks this client is holding. |

**Related topics:**

• "Metadata server" on page 18

## STC_RemoteServiceAccessPoint

The STC_RemoteServiceAccessPoint class represents a remote service access point. It provides information that you can use to access the SAN File System console. This class extends the CIM_RemoteServiceAccessPoint class.

**Related topics:**

• "User interfaces" on page 25

## STC_Setting

The STC_Setting class is the base class for cluster and server configuration parameters. This class extends the CIM_Setting class.

**Properties:**

The STC_Setting class has the following properties:

*Table 64. STC_Setting class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system. For the STC_MasterDisruptiveSetting and STC_MasterDynamicSetting classes, the class name of the scoping system is STC_Cluster. For the STC_TankDisruptiveSetting and STC_TankTransientSetting classes, the class name of the scoping system is STC_ComputerSystem. This property is key. Maximum is 256 characters. |

*Table 64. STC_Setting class properties (continued)*

| Name | Type | Description |
|---|---|---|
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| ServiceCreation ClassName | string | The class name of the scoping service. For the STC_MasterDisruptiveSetting and STC_MasterDynamicSetting classes, the class name of the scoping service is STC_MasterService. For the STC_TankDisruptiveSetting and STC_TankTransientSetting classes, the class name of the scoping service is STC_TankService. This property is key. Maximum is 256 characters. |
| ServiceName | string | The instance name of the scoping service. This property is key. Maximum is 64 characters. |

**Related topics:**

- "STC_MasterDisruptiveSetting" on page 106
- "STC_MasterDynamicSetting" on page 107
- "STC_TankDisruptiveSetting" on page 144
- "STC_TankTransientSetting" on page 152

## STC_StoragePool

The STC_StoragePool class represents a storage pool. It extends the CIM_ManagedSystemElement class.

There is an instance of this class for every storage pool that exists in a SAN File System. The instances include a System storage pool that the SAN File System uses to maintain system metadata. At least one other storage pool must exist for client files, which is called the Default storage pool. The list of instances and the methods defined in this class are available only on the master Metadata server.

**Properties:**

The STC_StoragePool class has the following properties:

*Table 65. STC_StoragePool class properties*

| Name | Type | Description |
|---|---|---|
| Caption | string | One-line description of the object. This property is read-only. Maximum length is 64 characters. |
| Name | string | Your label for the storage pool. This property is key. Maximum length is 256 characters. |
| PoolType | uint32 | Type of pool. This property is read-only. Possible values are:<br>0: User<br>1: User Default<br>2: System<br>The default is User. You can use the SetDefault() method to change the storage pool type to User Default if the storage pool type is User. |
| PartitionSize | uint64 | Partition size, in megabytes, to use when a fileset allocates space. This property is read-only. Possible values are 16, 64, and 256 MB. The default is 16 MB. |

*Table 65. STC_StoragePool class properties (continued)*

| Name | Type | Description |
|---|---|---|
| AllocSize | uint32 | Allocation strategy to use for files on this storage pool:<br>• System allocation - An escalation algorithm is used to allocate blocks to a file placed on this storage pool. Indicated by a AllocSize value of zero. This is the default.<br>• Fixed allocation - The file is extended by a chosen fixed size every time. The fixed allocation size is indicated by this AllocSize value, either 4 KB or 128 KB.<br>This property is read-only. Possible values are 0, 4, or 128. Default is 0. |
| AlertPercentage | uint16 | Percentage of the estimated storage pool size that, when reached, causes the server to generate an alert message. This property is writeable. Minimum is 0% and indicates that the server should not generate an alert. Maximum is 100%. The default is 80%. |
| Size | uint64 | Size, in megabytes, of the storage pool. The size of the storage pool is the sum of the sizes of the volumes within the storage pool. The size of the storage pool can change as volumes are added and deleted. This property is read-only. |
| SizeAllocated | uint64 | Size, in megabytes, of the storage pool allocated to filesets. The files within a fileset use a portion of the size allocated to a fileset. The rest is free size. This property is the sum of the allocated sizes of the volumes within the storage pool. It is read-only. |
| SizeAllocatedPercentage | uint16 | Percentage of the size allocated in the storage pool. You can compare this value to the AlertPercentage property value to determine how close the storage pool is to causing an alert. This property is read-only. Minimum is 0%. Maximum is 100%. |
| NumberOfVolumes | uint32 | The total number of volumes assigned to this storage pool. This property is read-only. |
| Description | string | Your description of the storage pool. This property is writeable. Maximum length is 256 characters. You cannot change the description for System storage pool |

**Related topics:**

- "Storage pools" on page 20
- "Create() method"
- "Delete() method" on page 140
- "Move() method" on page 140
- "SetDefault() method" on page 141

## Create() method

You can use the Create() method to define a new user storage pool. This method is the constructor for the class.

**Execute Role:** Administrator

**Method Type:** Static

**Parameters:**

Table 66 describes the parameters you can specify for the Create() method:

*Table 66. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| Name | string | Input parameter that is your label for the storage pool. Maximum is 256 characters. |
| Description | string | Input parameter that is your description of the storage pool. Maximum is 256 characters. |
| PartitionSize | uint32 | Input parameter that is the partition size, in megabytes, to use when a fileset allocates space. Possible values are 16, 64, and 256 MB. The default is 16 MB. |
| AllocSize | uint32 | Input parameter that is the allocation strategy to use for files on this storage pool:<br>• System allocation - An escalation algorithm is used to allocate blocks to a file placed on this storage pool. Indicated by a AllocSize value of zero. This is the default.<br>• Fixed allocation - The file is extended by a chosen fixed size every time. The fixed allocation size is indicated by this AllocSize value, either 4 KB or 128 KB.<br>This property is read-only. Possible values are 0, 4, or 128. Default is 0. |
| AlertPercentage | uint16 | Input parameter that is the percentage of the storage pool size that, when reached, causes the server to generate an alert message. Minimum is 0%. Maximum is 100%. |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 9 (Name not valid - The name has invalid characters.)
- 13 (Is Default - DEFAULT is a reserved name and cannot be used as a name for a user storage pool.
- 15 (Is System - SYSTEM is a reserved name and cannot be used as a name for a user storage pool.)
- 18 (Storage pool name already exists.)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- “Storage pools” on page 20
- “STC_StoragePool” on page 137

## Delete() method

You can use the Delete() method to delete an existing, empty, unreferenced storage pool. You cannot delete a storage pool that contains volumes or that has references in an active policy.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The Delete() method returns one of the following codes:

- 0 (Completed successfully)
- 5 (Storage pool is in use.)
- 8 (Integrity lost)
- 13 (Storage pool is the Default storage pool.)
- 14 (The current active policy references the storage pool.)
- 15 (Storage pool is a System storage pool.)
- 21 (Storage pool not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Storage pools" on page 20
- "STC_StoragePool" on page 137

## Move() method

You can use the Move() method to move or rename a storage pool by creating a new storage pool with the specified name and migrating the data and capabilities to the new name. If successful, the old storage pool will be removed.

You cannot rename a System storage pool. You cannot use SYSTEM or DEFAULT for the new name.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 67 describes the parameters you can specify for the Move() method:

*Table 67. Move() method parameters*

| Name | Type | Description |
|---|---|---|
| NewName | string | Input parameter that is your new label for the storage pool. |

**Return values:**

The Move() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 9 (Name not valid - The name has invalid characters.)
- 13 (Is Default - DEFAULT is a reserved storage pool name)
- 15 (Is System - Cannot rename the System storage pool or SYSTEM cannot be the new name.)
- 18 (Storage pool name already exists.)
- 21 (Storage pool not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Storage pools" on page 20
- "STC_StoragePool" on page 137

## SetDefault() method

You can use the SetDefault() method to change a user storage pool to the default storage pool. The STC_StoragePool class PoolType property changes from User to User Default.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The SetDefault() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 13 (Is already Default)
- 15 (Storage pool is a System storage pool)
- 21 (Storage pool not found)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Storage pools" on page 20

- "STC_StoragePool" on page 137

## STC_SystemMDRAid

The STC_SystemMDRAid class supports recovering metadata for a cluster. It provides a mechanism to extract system metadata information into a recovery file on the local disk of the system (not on the SAN). This class extends the CIM_ManagedSystemElement class.

You can extract multiple recovery files to save the state of the system metadata at various points in time. You can generate administrative commands from the recovery file, to re-create metadata using the GenerateCommandFiles() method. After you re-create the metadata, you can create the recovery file again and verify it with the original recovery file.

**Properties:**

The STC_SystemMDRAid class has the following properties:

*Table 68. STC_SystemMDRAid class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_Cluster. This property is key. Maximum is 256 characters. |
| SystemName | string | The instance name of the scoping system. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_SystemMDRAid. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | Name of the extracted metadata recovery file in the format <name>.dump. This property is key. Maximum length is 256 characters. |
| LocalDirectoryName | string | The Metadata server local disk directory name where the recovery files and the generated command files are stored. This property is read-only. Maximum is 256 characters. |
| CLIGeneratorName | string | The script used to generate command files from the metadata recovery file. This property is read-only. |
| InstallDate | datetime | The date when the recovery file was created. This property is read-only. |
| Size | uint64 | The size, in kilobytes, of the metadata recovery file. This property is read-only. |

**Related topics:**

- "Create() method"
- "Delete() method" on page 143
- "GenerateCommandFiles() method" on page 144

### Create() method

You can use the Create() method to create a new metadata recovery file. This method is the constructor for the class.

**Execute Role:** Administrator

**Method Type:** Static

**Parameters:**

Table 69 describes the parameters you can specify for the Create() method:

*Table 69. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| Name | string | Input parameter that is the name of the extracted metadata recovery file in the format <name>.dump. Maximum length is 256 characters. |
| IsForce | boolean | Input parameter that indicates whether to overwrite an existing recovery file |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)
- 7 (Insufficient space)
- 8 (Integrity lost)
- 9 (Invalid name)
- 18 (Recovery file name already exists)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed.)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "STC_SystemMDRAid" on page 142

## Delete() method

You can use the Delete() method to delete an existing metadata recovery file.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The Delete() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- .. (Internal error)

**Related topics:**

- "STC_SystemMDRAid" on page 142

### GenerateCommandFiles() method

You can use the GenerateCommandFiles() method to generate commands for re-creating metadata from a recovery file. Any existing command files are overwritten. The method can generate the following set of command files:

- TankSysCLI.auto - This file contains commands to re-create storage pools, filesets, and policies. In case of disaster, this file can be run without manual intervention.
- TankSysCLI.volume - This file contains commands to re-create volumes. This file cannot be run without manual verification and editing.
- TankSysCLI.attachpoint - This file contains commands to re-create fileset attach points. This file cannot be run without manual verification, editing, and intervention.

These command files are needed only for recovery, and their generation can be postponed until needed. Also the Metadata server does not have to be up and running to generate these files.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The GenerateCommandFiles() method returns one of the following codes:

- 0 (Completed successfully)
- 7 (Insufficient space)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 22 (Not the primary Administrative server)
- .. (Internal error)

**Related topics:**

- "STC_SystemMDRAid" on page 142

## STC_TankDisruptiveSetting

The STC_TankDisruptiveSetting class contains the settings for server-specific configuration parameters that need a Metadata server restart for an update to take effect. These parameters are read-only and can only be specified on the command line when the server is started. This class extends the STC_Setting class.

**Properties:**

The STC_TankDisruptiveSetting class has the following properties:

*Table 70. STC_TankDisruptiveSetting class properties*

| Name | Type | Description |
|---|---|---|
| ServerName | string | Unique server name that can be set only at installation. This property is read-only. Maximum is 32 characters. |

*Table 70. STC_TankDisruptiveSetting class properties (continued)*

| Name | Type | Description |
|---|---|---|
| ProtocolType | uint32 | Client-server and server-server communication protocol type.<br>• 0: UDP<br>• 1: TCP<br>Default is 0. |
| ClientNetwork Protocol | uint32 | Client-server communication protocol type. This property is read-only. Possible values are:<br>• 0: UDP<br>• 1: TCP<br>Default is 1. |
| ServerNetwork Protocol | uint32 | Server-server communication protocol type. This property is read-only. Possible values are:<br>• 0: UDP<br>• 1: TCP<br>Default is 0. |
| NumDeleteThr eads | uint32 | Number of threads for garbage collection of deleted files. This property is read-only. Minimum is 1 and maximum is 4. |
| PrimaryIP | string | IP address of the Ethernet interface for the server. This property is read-only. |
| ClusterPort | uint32 | Cluster port used by internal group services infrastructure communication. This port must be free on the interface when the server is started. This property is read-only. Minimum is 1 024 and maximum is 65 535. Default is 1 737. |
| HeartbeatPort | uint32 | Heartbeat port used by internal group services infrastructure communication on which to receive heartbeats. This port must be free on the interface when the server is started. This property is read-only. Minimum is 1 024 and maximum is 65 535. Default is 1 738. |
| STPPort | uint32 | SAN File System protocol port used for communication with file system clients. This port must be free on the interface when the server is started. This property is read-only. Minimum is 1 024 and maximum is 65 535. Default is 1 700. |
| AdminPort | uint32 | The port to receive administrative requests. This port must be free on the interface when the server is started. This property is read-only. Minimum is 1 024 and maximum is 65 535. Default is 1 800. |

**Related topics:**

• "STC_Setting" on page 136

## STC_TankEvents

The STC_TankEvents class represents the possible events that a Metadata server can generate. This class extends the CIM_LogicalElement class.

A master Metadata server can generate any of these events and a subordinate Metadata server can generate only a subset. An instance of this class is not the list of events that occurred in a server, just the possible events that could occur.

SNMP traps are either generic or specific traps as indicated by the SNMPTrap property. The meaning of a generic trap can be interpreted by the message content. Thus, only generic traps contain the message actually logged in the event log.

A specific trap is specific to a particular event. A specific trap does not contain the message content. Rather, the name of the trap itself indicates what specific event has occurred. For example, the tankClusterStateChangeTrap trap is generated whenever the primary Administrative server changes the cluster state. This trap contains the OldState and the CurrentState values of the cluster. Each specific trap also contains context (varbinds) that further identifies what happened. Some other specific traps are:

- tankLogRotateTrap - The Metadata server log has been rotated.
- tankStoragePoolSpaceTrap - The storage pool usage has exceeded its alert percentage with a new allocation.
- tankContainerQuotaTrap - Fileset hard or soft quota violation.

For definitions of all the specific traps, see Appendix B, "SNMP Trap MIB", on page 255.

**Properties:**

The STC_TankEvents class has the following properties:

*Table 71. STC_TankEvents class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreation ClassName | string | The class name of the scoping system. This property is key. Maximum is 256 characters. |
| SystemName | string | The name of the scoping system. This property is key. Maximum is 256 characters. |
| ServiceCreation ClassName | string | The class name of the scoping service. This property is key. Maximum is 256 characters. |
| ServiceName | string | The name of the scoping service. This property is key. Maximum is 256 characters. |
| MessageID | string | The ID associated with the message that will be logged in the Metadata server log when this event occurs. This property is key. |
| Severity | uint8 | The severity level of the event. This property is read-only. Possible values are:<br>• 0: Information<br>• 1: Warning<br>• 2: Error<br>• 3: Severe |
| Message | string | The message format the Metadata server will use to log a message when this event occurs in the server. If there are any parameter format specifications in this string, they are replaced with actual values when this message is logged. This property is read-only. |
| SNMPTrap | string | The name of the SNMP trap generated by this event. All instances with this property value set to tankGenericTrap are generic traps. The rest of the instances are specific traps. This property is read-only. |

*Table 71. STC_TankEvents class properties (continued)*

| Name | Type | Description |
|---|---|---|
| IsSNMPTrapEnabled | boolean | Indicator of whether the SNMPEvents configuration parameter filter allows the generation of the SNMP trap if this event occurs. This property is read-only. |

**Related topics:**

- "SNMP" on page 20
- "Test() method"

## Test() method

You can use the Test() method to generate a test event. You can check that an SNMP manager can receive a trap. This event causes a trap with severity information. Make sure that the following conditions are met:

- SNMPEvents configuration parameter is set to allow event with severity information.
- SNMPManagers configuration parameter is also set properly.
- Specified SNMP managers are configured properly and active to receive traps. The SNMPManagers property in the "STC_MasterDynamicSetting" on page 107 class lists the SNMP managers' IP addresses.

**Execute Role:** Administrator

**Method Type:** Static

**Return values:**

The Test() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- .. (Internal error)

**Related topics:**

- "SNMP" on page 20
- "STC_TankEvents" on page 145

# STC_TankMetrics

The STC_TankMetrics class represents the metrics for each Metadata server. Only one instance of this class should exist for each server running including the master Metadata server. This class extends the CIM_ServiceStatisticalInformation class.

The metrics include the current totals for the following types of buffers:

- Clean - Buffers that contain data but are available for reuse.
- Dirty - Buffers that contain data that is awaiting I/O to disk.
- Free - Buffers that are available because they are currently not in use.

**Properties:**

The STC_TankMetrics class has the following properties:

*Table 72. STC_TankMetrics class properties*

| Name | Type | Description |
|---|---|---|
| TotalUserMeta Activity | uint64 | Total number of transactions relating to file system metadata activity including fileset attach points, creating directories, and extending files. This property is read-only. |
| TotalUserMeta UpdateActivity | uint64 | Total number of transactions relating to file system updates for system objects. This property is read-only. |
| SessionLocks | uint64 | Current number of session locks held in the Lock Manager for this server. Session lock holds a reference in each file that it manages. You must acquire a session lock to perform any actions with the file, such as a stat, lstat, or opendir operation. This property is read-only. |
| DataLocks | uint32 | Current number of data locks held in the Lock Manager. This property is read-only. |
| ByteRangeLoc ks | uint32 | Current number of byte range locks held in the Lock Manager. This property is read-only. |
| TotalBuffers | uint32 | Current number of total buffers for user metadata activity. This property is read-only. |
| CleanBuffers | uint32 | Current number of clean buffers for user metadata activity. Clean buffers contain data but the buffers are available for reuse. This property is read-only. |
| DirtyBuffers | uint32 | Current number of dirty buffers for user metadata activity. Dirty buffers contain data awaiting I/O to disk. This property is read-only. |
| FreeBuffers | uint32 | Current number of free buffers for user metadata activity. Free buffers are available because they are currently not in use. This property is read-only. |

**Related topics:**


- "Metadata server" on page 18


## STC_TankSAP

The STC_TankSAP class represents a Metadata server service access point. It extends the CIM_ServiceAccessPoint class.

**Properties:**

The STC_TankSAP class has the following properties:

*Table 73. STC_TankSAP class properties*

| Name | Type | Description |
|---|---|---|
| TypeOfAddress | uint16 | An enumeration that defines how to format the address and mask of the address range that defines this IP subnet. Whenever possible, IPv4-compatible addresses should be used instead of IPv6 addresses (see RFC 2373, section 2.5.4). To have a consistent format for IPv4 addresses in a mixed IPv4 and IPv6 environment, all IPv4 addresses and both IPv4-compatible IPv6 addresses and IPv4-mapped IPv6 addresses, per RFC 2373, section 2.5.4, should be formatted in standard IPv4 format. However, the 2.2 version of the Network Common Model will not explicitly support mixed IPv4 and IPv6 environments. This support will be added in a future release. This property is read-only. Possible values are:<br>0: Unknown<br>1: IPv4<br>2: IPv6<br>The default is 1. |
| Ip | string | IP address of the Ethernet network interface of a Metadata server. The Group Services and the SAN File System protocol are bound to this IP at boot time. The HeartBeat protocol and the Admin Service are also bound to this port for service. This property is read-only. |
| RSAIp | string | The IP address for the Remote Supervisory Adapter (RSA) card on the engine. This property is read-only. Default is 0.0.0.0. |
| ClusterPort | uint32 | Cluster port used by internal group services infrastructure communication. This port must be free on both the interfaces when the server is started. This property is read-only. Default value is 1 737. Minimum is 1 024. Maximum is 65 535. |
| HeartbeatPort | uint32 | Heartbeat port used by internal infrastructure communication to receive heartbeats. This port must be free on both the interfaces when the engine is started. This property is read-only. Default value is 1 738. Minimum is 1 024. Maximum is 65 535. |
| STPPort | uint32 | SAN File System protocol port used for communication with file system clients. This port must be free on both the interfaces when the engine is started. This property is read-only. Default value is 1 700. Minimum is 1 024. Maximum is 65 535. |
| AdminPort | uint32 | The port to receive administrative requests. This port must be free on both the interfaces when the engine is started. This property is read-only. Minimum is 1 024. Maximum is 65 535. |

**Related topics:**

• "Metadata server" on page 18

## STC_TankService

The STC_TankService class represents a Metadata server and provides server services. It extends the CIM_Service class.

**Properties:**

The STC_TankService class has the following properties:

*Table 74. STC_TankService class properties*

| Name | Type | Description |
|---|---|---|
| CurrentState | uint32 | Indicates the state of the Metadata server. This property is read-only. Possible values are:<br>0: Down<br>1: Online<br>2: Partly Quiescent - Only server I/O operations are suspended.<br>3: Fully Quiescent - All background I/O, client, and server operations are suspended.<br>4: Administrative Quiescent - No longer servicing clients.<br>5: Initializing for the first time<br>6: FailedInit - Encountered an error during startup or group formation.<br>7: UnCommissioned - Not commissioned into a cluster.<br>8: Joining a cluster.<br>9: Unknown |
| PendingState | uint32 | The current state of the server transitions to this state if the current state is different from this pending state. This property is read-only. Possible values are the same as the CurrentState property. |
| LastBootUpTime | datetime | Time when the server was last started. This property is read-only. |
| LocalDateTime | datetime | Local date and time of day according to the server. This property is read-only. |
| LastCurrentStateChangeTime | datetime | Time since the server changed its current state. This property is read-only. |
| LastPendingStateChangeTime | datetime | Time since the server has a state change pending. This property is read-only. |
| CurrentVersion | string | Current software release version. This is the version of the latest upgrade. This will be different from the previous (committed) version if there was an upgrade but the commit was not yet activated. This property is read-only. |
| IsMaster | boolean | Indicates if this is the master Metadata server. This property is read-only. |
| NumberOfContainers | uint32 | Number of filesets served from this server. This property is read-only. |

**Related topics:**

- "Metadata server" on page 18
- "BecomeMaster() method"
- "StartService() method" on page 151
- "StopService() method" on page 152

## BecomeMaster() method

You can use the BecomeMaster() method to make this server the master Metadata server, if the master Metadata server is irrecoverably lost. A master can be lost because of hardware failures, software failures, or partitioned networks.

**Notes:**

1. You must ensure that the previous master is down to prevent rogue server issues. Turn the power off to the engine hosting the master Metadata server that is down.
2. You cannot change the master when the cluster is down.
3. When a master is lost, the subordinate servers are no longer in operational states.
4. You must manually log in to the Metadata server and edit the STC_TankService class property file to set the IsMaster property to True.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The BecomeMaster() method returns one of the following codes:

- 0 (Completed successfully. This server has become the master Metadata server.)
- 8 (Integrity lost)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 58 (The server is already the master Metadata server.)
- 59 (The server is a subordinate, but it is not in the right state to become the master Metadata server.)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 67 ( Cannot become the primary Administrative server)
- .. (Internal error)

**Related topics:**

- "Metadata server" on page 18
- "STC_TankService" on page 149

## StartService() method

You can use the StartService() method to start the Metadata server on this engine.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The StartService() method returns one of the following codes:

- 0 (Completed successfully - The Metadata server started.)
- 1 (Not supported)
- 4 (Command failed)
- 5 (In use)
- 8 (Integrity lost)
- 21 (Not found)
- 24 (Server timed out)

- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 66 (Metadata server restart service state cannot continue)
- 89 (Server exited after starting - The Metadata server restart service is still enabled and tries to start the server again until the retry limit is reached. If the server does not start within the limited number of retries, the Metadata server restart service is disabled.)
- .. (Internal error)

**Related topics:**
- "Metadata server" on page 18
- "STC_TankService" on page 149

### StopService() method

You can use the StopService() method to stop the Metadata server on this engine.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The StopService() method returns one of the following codes:
- 0 (Completed successfully)
- 1 (Not supported)
- 4 (Command failed)
- 8 (Integrity lost)
- 21 (Not found)
- 24 (Server timed out)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 66 (Metadata server restart service state cannot continue)
- .. (Internal error)

**Related topics:**
- "Metadata server" on page 18
- "STC_TankService" on page 149

## STC_TankTransientSetting

The STC_TankTransientSetting class contains server-specific configuration parameters that are effective only until the next restart. This class extends the STC_Setting class.

**Related topics:**
- "STC_Setting" on page 136

## STC_TankWatchdog

The STC_TankWatchdog class represents the Metadata server restart service operations. There is an instance of this class for each Metadata server in the cluster. It extends the CIM_LogicalElement class.

This administrative service keeps track of the vitality of a Metadata server. If enabled, the service restarts the server when it detects positively that the server is down.

**Properties:**

The STC_TankWatchdog class has the following properties:

*Table 75. STC_TankWatchdog class properties*

| Name | Type | Description |
|---|---|---|
| SystemCreationClassName | string | The class name of the scoping system: STC_Cluster. This property is key. Maximum is 256 characters. |
| SystemName | string | The name of the STC_Cluster instance that is the scoping system. This property is key. Maximum is 256 characters. |
| ServiceCreationClassName | string | The class name of the scoping service: STC_TankService. This property is key. Maximum is 256 characters. |
| ServiceName | string | The name of the STC_TankService instance that is the the scoping service. This property is key. Maximum is 256 characters. |
| CreationClassName | string | The name of the class or the subclass used in the creation of an instance: STC_TankWatchdog. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified. This property is key. Maximum is 256 characters. |
| Name | string | An address or other identifying information to uniquely name the Metadata server restart service. This property is key. Maximum is 256 characters. |
| State | uint32 | State of the Metadata server restart service. This property is read-only. Possible values are:<br>0: Off - The Metadata server restart service is manually turned off.<br>1: On - The Metadata server restart service is manually turned on.<br>2: Standby - The Metadata server restart service is in a passive standby mode because the server it is probing has been manually shut down. The watchdog automatically turns on when this server is restarted.<br>3: Aborted - The Metadata server restart service reached the retry limit for detecting server liveness and the Metadata server restart service was turned off. An administrator must manually turn on the watchdog again to continue.<br>Unknown — The Metadata server restart service is in an indeterminate state because the Metadata server restart service server could not be reached.<br>The state of the Metadata server restart service persists with a server restart. Default is 0. |

*Table 75. STC_TankWatchdog class properties (continued)*

| Name | Type | Description |
|---|---|---|
| ProbeState | uint32 | Server status found in the current probe cycle. This property is read-only. Possible values are:<br>0: Not Probed - Metadata server restart service has not started the probe because it is off or in a standby or an aborted state<br>1: Probing - Metadata server restart service has started a probe.<br>2: Server Live - Metadata server restart service detected that the server is live. There is no need to restart the server.<br>3: Server Absent - Metadata server restart service positively detected that the server is absent. The Metadata server restart service attempts to restart the server.<br>Unknown - The liveness test failed and the absence test failed. The Metadata server restart service does not attempt to restart the server.<br>Default is 0. |
| ProbeInterval | uint32 | Interval, in seconds, at which the Metadata server restart service will periodically start a probe. This property is read-only. Minimum is 10 seconds and maximum is 60. Default is 10. |
| LiveTestTimeoutInterval | boolean | Maximum time, in seconds, to wait for the server to respond to a liveness test request before deciding that the server is not live. This property is read-only. Minimum is 1 seconds and maximum is 10. Default is 2. |
| RetryLimit | uint32 | Number of times to try detecting liveness of the server if the server is declared not live. When this retry limit is reached, the watchdog is turned off. This property is the number of tries, not the number of retries. For example, if this value is 3, the probe is sent three times, the original time plus two retries. This property is read-only. Minimum is 1 and maximum is 10. Default is 3. |
| **The following properties represent statistics initialized when watchdog becomes active.** | | |
| StartTimeStamp | datetime | The date and time when the Metadata server restart service was started. This property is read-only. |
| LastProbeTimeStamp | datetime | The date and time when the last probe was started. This property is read-only. |
| TotalProbes | uint64 | Counter of the total number of probes done so far. This property is read-only. |
| LiveTestTimeouts | uint64 | Counter of the total number of times the liveness test has taken longer than the test timeout interval and caused a timeout error to occur. This property is read-only. |
| **The following properties represent statistics for retries.** | | |
| TotalRetries | uint64 | Counter of the total number of retries done so far. This property is read-only. |
| CurrentRetries | uint32 | Number of retries in the current probe cycle. This property is read-only. The value of this property ranges from 0 to one less than the limit as set by the RetryLimit property. |
| RetriesLWM | uint32 | The lowest number of retries reached so far. This property is read-only. |
| RetriesHWM | uint32 | The highest number of retries reached so far. This property is read-only. |

*Table 75. STC_TankWatchdog class properties (continued)*

| Name | Type | Description |
|---|---|---|
| **The following properties represent statistics for liveness tests.** | | |
| LastLiveTestTime | uint32 | Time taken by the last liveness test. This property is read-only. |
| LiveTestTimeLWM | uint32 | Low watermark for time, in milliseconds, taken by the liveness test. This property is read-only. |
| LiveTestTimeHWM | uint32 | High watermark for time, in milliseconds, taken by the liveness test. This property is read-only. |
| **The following properties represent statistics absence tests.** | | |
| TotalAbsenceTests | uint64 | Counter of the total number of time absence test was started. This property is read-only. |
| LastAbsenceTestTime | uint32 | Time, in milliseconds, taken by the last absence test. This property is read-only. |
| AbsenceTestTimeLWM | uint32 | Low watermark time, in milliseconds, taken by the last absence test. This property is read-only. |
| AbsenceTestTimeHWM | uint32 | High watermark for time, in milliseconds, taken by the last absence test. This property is read-only. |

**Related topics:**

- "Metadata server" on page 18
- "Enable() method"
- "Disable() method"

## Disable() method

You can use the Disable method to disable the Metadata server restart service.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The Disable() method returns one of the following codes:

- 0 (Completed successfully)
- 61 (Cannot connect to Metadata server)
- 64 (Metadata server restart service is already disabled)
- 66 (Cannot continue Metadata server restart service)
- .. (Internal error)

**Related topics:**

- "Metadata server" on page 18
- "STC_TankWatchdog" on page 153

## Enable() method

You can use the Enable() method to enable the Metadata server restart service.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The Enable() method returns one of the following codes:

- 0 (Completed successfully)
- 61 (Cannot connect to Metadata server)
- 63 (Metadata server restart service is already enabled)
- 66 (Cannot continue Metadata server restart service)
- .. (Internal error)

**Related topics:**

- "Metadata server" on page 18
- "STC_TankWatchdog" on page 153

## STC_Volume

The STC_Volume class represents a volume. The list of instances and the methods defined in this class are available only on the master Metadata server. This class extends the CIM_ManagedSystemElement class.

A volume is a logical unit number (LUN) labeled by SAN File System for its use. You can use theSTC_AvailableLUNs class to see all the LUNs available on a host engine.

**Properties:**

The STC_Volume class has the following properties:

*Table 76. STC_Volume class properties*

| Name | Type | Description |
|---|---|---|
| StoragePoolName | string | Storage pool to which this volume belongs. This property is read-only. |
| Caption | string | One-line description of the object. This property is read-only. Maximum length is 64 characters. |
| Name | string | Your label for the volume. This property is key. Its alias is VolumeName. Maximum length is 256 characters. |
| OSDeviceName | string | The file path to the storage device. This property is read-only. Maximum length is 256 characters. |
| State | uint32 | State of the volume. This property is read-only. Possible values are:<br>0: Normal - Volume is available for reading, writing and allocation.<br>1: Suspend Allocations - Allocation of new partitions to filesets is suspended. A client can still read from and write to the volume and a fileset can still allocate a new file on this volume if sufficient space exists.<br>2: Volume being Deleted - Volume is processing a deletion request. |
| Size | uint64 | Size, in megabytes, of the volume. This property is read-only. |
| SizeAllocated | uint64 | Size, in megabytes, of the volume allocated to filesets. This property is read-only. |

*Table 76. STC_Volume class properties (continued)*

| Name | Type | Description |
|---|---|---|
| SizeAllocatedPercentage | uint16 | Percentage of the size allocated in the volume. This property is read-only. Minimum is 0%. Maximum is 100%. |
| Description | string | Your description of the volume. This property is writeable. Maximum length is 256 characters. |

**Related topics:**

- "Volumes" on page 27
- "Create() method"
- "Delete() method" on page 158
- "GetNextFOV() method" on page 159
- "Move() method" on page 160
- "ResetFOV() method" on page 161
- "ResumeAllocation() method" on page 162
- "SuspendAllocation() method" on page 162

## Create() method

You can use the Create() method to attach a volume to a storage pool. This method is the constructor for the class.

**Execute Role:** Administrator

**Method Type:** Static

**Parameters:**

Table 77 describes the parameters you can specify for the Create() method:

*Table 77. Create() method parameters*

| Name | Type | Description |
|---|---|---|
| OSDeviceName | string | Input parameter that is the file path to the storage device. This property is read-only. Maximum length is 256 characters. |
| StoragePoolName | string | Input parameter that is your label for the storage pool to which you are attaching the volume. Maximum is 256 characters. |
| VolumeName | string | Input parameter that is your label for the volume. Maximum length is 256 characters. |
| Description | string | Input parameter that is your description of the volume. Maximum is 256 characters. |
| IsForce | boolean | Input parameter that indicates whether a volume will be deleted even if it has files on it. |
| IsSuspendAllocations | boolean | Input parameter that indicates whether the volume state is set to Suspend Allocations. |

**Return values:**

The Create() method returns one of the following codes:

- 0 (Completed successfully)

- 3 (Already defined - The storage device has a label and IsForce parameter is set to True but the device is already defined as another volume.)
- 5 (In use - The storage device has a label and IsForce parameter is set to False.)
- 8 (Integrity lost)
- 9 (Volume name is not valid)
- 10 (Parameter not valid)
- 12 (Storage device I/O failed)
- 18 (Volume name already exists.)
- 21 (Storage device not found)
- 22 (Not the primary Administrative server)
- 28 (Storage pool not found)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 52 (Disk not viable for one of the following reasons:
  - The volume does not hold at least one partition
  - The specified local storage device is not viable as a global disk
  - Hashing using World-Wide Name (WWN) conflicts with an existing hash
  - Sector size is less than 512 or greater than 4096
  - If a volume is being added to the System storage pool, the sector sizes of all volumes are not the same.
- 56 (Storage device access denied)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

If errors such as I/O failed, invalid size or internal error occur, you can decommission the disk by deleting the volume.

**Related topics:**

- "Volumes" on page 27
- "STC_Volume" on page 156

## Delete() method

You can use the Delete() method to delete an existing volume. If the IsForce parameter is True, this method deletes all the files that partly or fully exist on the volume before it deletes the volume. If the IsForce parameter is False, this method drains the volume first by moving the file data that resides on the volume to other volumes in the same storage pool. If the volume drain fails, the volume enters the Suspend Allocations state requiring manual administrative action.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 78 on page 159 describes the parameters you can specify for the Delete() method:

*Table 78. Delete() method parameters*

| Name | Type | Description |
|---|---|---|
| IsForce | boolean | Input parameter that indicates whether a volume will be deleted even if it has files on it. |

**Return values:**

The Delete() method returns one of the following codes:

- 0 (Completed successfully)
- 2 (Access failed)
- 5 (In use - Volume drain failed and volume is not empty.)
- 8 (Integrity lost)
- 12 (I/O failed)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 33 (Volume not found)
- 44 (Incompatible operation)
- 56 (Access denied)
- 57 (No space - No space in other volumes of the storage pool for volume drain.)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline - Force option removes files from filesets. The serving server must be online. Without force, volume is drained; need to revoke locks; the serving fileset must be up.)
- .. (Internal error)

**Related topics:**

- "Volumes" on page 27
- "STC_Volume" on page 156

## GetNextFOV() method

You can use the GetNextFOV() method to get the next file on volume (FOV) entry, by providing the FOV iterator in the FOVHandle input parameter. The file entry is made available in the FOVEntry output parameter. The iterator to use for the next call is returned in the FOVHandle parameter.

**Note:** This method can be a long-running process depending on the number of files and size of storage.

**Execute Role:** Backup

**Method Type:** Dynamic

**Parameters:**

Table 79 on page 160 describes the parameters you can specify for the GetNextFOV() method:

*Table 79. GetNextFOV() method parameters*

| Name | Type | Description |
|---|---|---|
| FOVHandle | string | Input/output parameter that is the FOV iteration identifier. |
| FOVEntry | string | Output parameter that is the file entry. |

**Return values:**

The GetNextFOV() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 37 (End of Iteration - End of file reached.)
- 38 (Invalid IterationIdentifier)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Volumes" on page 27
- "STC_Volume" on page 156

## Move() method

You can use the Move() method to move or rename a volume. This method creates a new volume with the specified new name and migrates the data and capabilities to the new volume. If successful, this method deletes the old volume.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Parameters:**

Table 80 describes the parameters you can specify for the Move() method:

*Table 80. Move() method parameters*

| Name | Type | Description |
|---|---|---|
| NewName | string | Input parameter that is your new label for the volume. |

**Return values:**

The Move() method returns one of the following codes:

- 0 (Completed successfully)
- 8 (Integrity lost)
- 9 (Name not valid - The new name has invalid characters.)
- 22 (Not the primary Administrative server)

- 30 (Transaction failed)
- 44 (Incompatible operation)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**
- "Volumes" on page 27
- "STC_Volume" on page 156

## ResetFOV() method

You can use the ResetFOV() method to reset a Files-On-Volume (FOV) iterator. This method creates and returns an iterator that can be used to locate each file entry that resides on this volume instance. You pass in the iterator when you invoke the GetNextFOV() method, which returns one successive file entry per call. Only one iterator can be active at any given instance.

**Note:** This method can be a long-running process depending on the number of files and size of storage.

**Execute Role:** Backup

**Method Type:** Dynamic

**Parameters:**

Table 81 describes the parameters you can specify for the ResetFOV() method:

*Table 81. ResetFOV() method parameters*

| Name | Type | Description |
|---|---|---|
| FOVHandle | string | Output parameter that is the FOV iteration identifier. |

**Return values:**

The ResetFOV() method returns one of the following codes:
- 0 (Completed successfully)
- 2 (Access failed)
- 7 (Insufficient space - Not enough temporary space on the local disk.)
- 8 (Integrity lost)
- 21 (Not found - No files found on this volume.)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 44 (Incompatible operation)
- 56 (Access denied)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- 65 (Server state offline - The volume has files of a fileset with its server offline.)
- .. (Internal error)

**Related topics:**

- "Volumes" on page 27
- "STC_Volume" on page 156

## ResumeAllocation() method

You can use the ResumeAllocation() method to resume suspended partition allocations on a volume.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The ResumeAllocation() method returns one of the following codes:

- 0 (Completed successfully)
- 2 (Access failed)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 33 (Volume not found)
- 35 (Allocations were not suspended.)
- 44 (Incompatible operation)
- 56 (Access denied)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**

- "Volumes" on page 27
- "STC_Volume" on page 156

## SuspendAllocation() method

You can use the SuspendAllocation() method to suspend partition allocations on a volume. A Metadata server cannot allocate new data on the volume.

**Execute Role:** Administrator

**Method Type:** Dynamic

**Return values:**

The SuspendAllocation() method returns one of the following codes:

- 0 (Completed successfully)
- 2 (Access failed)
- 8 (Integrity lost)
- 22 (Not the primary Administrative server)
- 30 (Transaction failed)
- 33 (Volume not found)
- 34 (Allocations already suspended.)

- 44 (Incompatible operation)
- 56 (Access denied)
- 61 (Cannot connect to server)
- 62 (Too many connections)
- .. (Internal error)

**Related topics:**
- "Volumes" on page 27
- "STC_Volume" on page 156

# Chapter 5. SAN File System MOF

The MOF defines the SAN File System classes and instances.

**Related topics:**

- "CIM-related concepts" on page 2

## MOF introduction

```
/**************************************************************************
**  Description  : CIM classes for SAN File System                       **
**                                                                       **
**  Dependency:                                                          **
**        See SubSchema27.mof                                            **
**************************************************************************/

#pragma locale("en_US")

/*
 * Qualifiers for role-based security:
 *
 *  ReadRole     - The role required by the principle to read the property
 *  WriteRole    - The role required by the principle to write the property
 *  ExecuteRole  - The role required by the principle to execute the method
 *
 * The list of roles are:
 *
 *    Administrator
 *                - Executes any and all commands in the SAN FS cluster.
 *
 *    Operator    - Provides use of all commands available for Backup and
 *                  Monitor roles, Also allows user to perform day to day
 *                  operations and tasks requiring frequent modifications.
 *
 *    Backup      - Allows the user to use all commands available to the
 *                  Monitor role.  In addition, it provides the user with the
 *                  ability to perform backup and restoration tasks.
 *
 *    Monitor     - Allows the user to get basic status on the SAN FS
 *                  cluster, display the log, display the rules in a Policy,
 *                  and list information regarding elements such as pools,
 *                  volumes, service classes, etc.
 *
 */

Qualifier ReadRole    : string = "Monitor",       Scope(property), Flavor(DisableOverride);
Qualifier WriteRole   : string = "Administrator", Scope(property), Flavor(DisableOverride);
Qualifier ExecuteRole : string = "Administrator", Scope(method),   Flavor(DisableOverride);

/*
```

*Figure 19. MOF introduction (Part 1 of 2)*

```
*  Model Notes
*       . For various reasons, we explicitly do not use the following CIM
*         objects.
*         Associations, Indications, References.
*
*  General Notes
*  . Valid Storage Tank Object names are as follows.
*       . Maximum 256 characters long unless otherwise stated.
*       . At least one alpha-numeric character.
*       . Should not have leading white space.
*       . May contain '-_.'
*
*  . SAN FS server may return the following codes for any method.
*       . Transaction Failed
*       . Integrity Lost
*
*
```

*Figure 19. MOF introduction (Part 2 of 2)*

**Related topics:**


- "Programming considerations" on page 51

## MOF STC_StoragePool class definition

```
/*
 *  STC_StoragePool
 *
 *  Notes:
 *
 *  There are two reserved names SYSTEM and DEFAULT. The SYSTEM pool is
 *  inherent to SAN.FS.
 *  The DEFAULT is introduced by the new CLI syntax grammar for adding
 *  a volume to the default storage pool using the reserved word DEFAULT
 *  in place of the actual storage pool name.
 *
 */
[Description ("A storage pool is a top level object that represents storage "
              "space in a SAN FS Server. A storage pool is made up of volumes "
              "a.k.a FC LUNs. A policy based placement is used to assign space "
              "for files of a Container. The policy is determined by the "
              "current active policy set during file creation. "
              "Space is allocated for a container in units of partition sizes. "
              "A storage pool chooses a volume to get a free partition in a "
              "round-robin fashion. "
              "This class has an instance for every storage pool that exists "
              "in a SAN FS. The instance include a SYSTEM storage pool that "
              "the SAN FS uses to maintain system meta-data. "
              "At least one other storage pool must exist for client files "
              "called the default storage pool. "
              "The list of instances and the methods defined in this class are "
              "available only on the Master SAN FS server."),
 provider("com.ibm.storage.storagetank.provider.xnp.StoragePool")
]
class STC_StoragePool: CIM_ManagedSystemElement
{
        [Override ("Caption"),
         Description("Provide a one-line caption.")]
    string Caption = "Storage Tank Storage Pool";

        [Override ("Name"),
         Key, MaxLen(256), Required,
         Description("The user-supplied administrative name for the "
                     "storage pool.")]
    string Name;

        [Description("The type of pool.  One of: User, UserDefault, or System. "
                     "You can change a user pool to a userDefault pool by "
                     "using the setDefault() method."),
         ValueMap {"0"   , "1"           , "2"},
         Values   {"User", "UserDefault", "System"}]
    uint32 PoolType;

        [Description("Current partition size to use when a container "
                     "needs to allocate space. The partition size can be "
                     "16 MBytes, 64 MBytes or 256 MBytes. Once defined, "
                     "the partition size cannot be changed. "
                     "The default is 16 MBytes."),
         ValueMap {"16", "64", "256"},
         Units("MegaBytes")]
    uint64 PartitionSize = 16;
```

*Figure 20. MOF STC_StoragePool class definition (Part 1 of 5)*

```
    [Description("Allocation strategy to use for files on this "
             "Storage Pool. "
             "This property of a storage pool dictates how blocks for "
             "a file are allocated. There are two types of allocation "
             "strategies. One type is the system strategy in which an "
             "escalation algorithm is used to allocate blocks to a file "
             "placed on this storage pool. Another type is the fixed "
             "allocation strategy in which the file is extended by a "
             "chosen fixed size every time. "
             "A value of zero indicates that the system allocation "
             "strategy is in effect. A value other than zero indicates "
             "that a fixed allocation strategy is in effect. "
             "The fixed allocation size can be either 4 KBytes or "
             "128 KBytes. The default is system allocation strategy."),
     MinValue(0), MaxValue(128), Units("KiloBytes"),
     ValueMap {"0", "4", "128"}]
uint32 AllocSize = 0;

    [Description("Current percentage used of the storage pool's estimated "
             "size that, when reached, causes the server to "
             "generate an alert message. The default value is 80%. "
             "A value of zero indicates that no alert is generated."),
     MinValue(0), MaxValue(100), Units("Percent"),
     Write]
uint16 AlertPercentage = 80;

    [Units("MegaBytes"),
     Description("Current size of the storage pool in MBytes. "
                 "Size of the storage pool is the sum of the sizes "
                 "of the volumes (LUNs) within the storage pool. The "
                 "size of the storage pool may grow and shrink as volumes "
                 "are added and deleted in the storage pool.")]
uint64 Size;

    [Units("MegaBytes"),
     Description("Current size of the storage pool allocated to "
                 "containers. A portion of the size allocated to a "
                 "container is used by the files within the container. "
                 "The rest of the portion is the free size."
                 "This is the sum of the allocated sizes of the volumes "
                 "within the storage pool.")]
uint64 SizeAllocated;

    [Description("Current percentage of size allocated in the pool. This "
                 "can be compared directly with the AlertPercentage to "
                 "determine how close the storage pool is to causing an "
                 "alert."),
     MinValue(0), MaxValue(100), Units("Percent")]
uint16 SizeAllocatedPercentage;
```

*Figure 20. MOF STC_StoragePool class definition (Part 2 of 5)*

```
    [Description("The total number of volumes assigned to this "
                 "storage pool.")]
uint32 NumberOfVolumes;

    [Override ("Description"),
     Write, MaxLen(256),
     Description("A description that is writable. You cannot change "
                 "the description for SYSTEM storage pool.")]
string Description = "";

/*
 * Methods
 */

    [Description("Define a new user storage pool. Constructor for this "
                 "class. "
                 "The return codes are as follows. "
                 " Ok - Successful. "
                 " Invalid Name - The name has invalid characters. "
                 " Is Default - DEFAULT is a reserved name and cannot "
                 "     be used as a name for user storage pool. "
                 " Is System - SYSTEM is a reserved name and cannot "
                 "     be used as a user storage pool. "
                 " Name Exists - Duplicate name. "
                 " Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 " Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 " Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 " Too Many Connections - Exceeded the limit on number "
                 "     of parallel administrative operations allowed. "
                 " Internal Error - The SAN FS server encountered an "
                 "     internal error. Please see the message log and call "
                 "     Technical Support for resolution."),

     Static,
     ValueMap{ "0", "9", "13", "15", "18", "22", "44", "61", "62", ".."},
     Values {"OK", "Invalid Name", "Is Default", "Is System", "Name Exists",
             "Not the Admin Master Server", "Incompatible Operation",
             "Cannot Connect to Server", "Too Many Connections",
             "Internal Error"}]
uint32 Create(
               [IN, MaxLen(256)] string Name,
               [IN, MaxLen(256)] string Description,
               [IN, ValueMap {"16", "64", "256"}, Units("MegaBytes")]
                    uint32 PartitionSize,
               [IN, MinValue(0), MaxValue(128),
                    Units("KiloBytes"), ValueMap {"0", "4", "128"}]
                    uint32 AllocSize,
               [IN, MinValue(0), MaxValue(100), Units("Percent")]
                    uint16  AlertPercentage
             );
```

*Figure 20. MOF STC_StoragePool class definition (Part 3 of 5)*

```
    [Description("Set the pool from type user to user-default.  This "
                 "will only work when the pool type is user. "
                 "The return codes are as follows. "
                 " Ok - Successful. "
                 " Is Default - Redundant operation. "
                 " Is System - SYSTEM storage pool cannot "
                 "     be used as a default storage pool. "
                 " Not Found - Storage Pool not found. Deprecated? "
                 " Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 " Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 " Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 " Too Many Connections - Exceeded the limit on number "
                 "     of parallel administrative operations allowed. "
                 " Internal Error - The SAN FS server encountered an "
                 "     internal error. Please see the message log and call "
                 "     Technical Support for resolution."),
     ValueMap{ "0", "13", "15", "21", "22", "44", "61", "62", ".."},
     Values {"OK", "Is Default", "Is System", "Not Found",
             "Not the Admin Master Server", "Incompatible Operation",
             "Cannot Connect to Server", "Too Many Connections",
             "Internal Error"}]
uint32 SetDefault();

    [Description("Delete an existing storage pool. Note that you cannot "
                 "delete a storage pool that has volumes in it. You cannot "
                 "delete a storage pool that has references in an active "
                 "policy set. "
                 "The return codes are as follows. "
                 " Ok - Successful. "
                 " In Use - Storage Pool has volumes in it. "
                 " Is Default - Storage pool is the default storage pool. "
                 " Is Referenced - The storage pool is referenced by "
                 "     the current active policy set. "
                 " Is System - SYSTEM storage pool cannot be deleted. "
                 " Not Found - Storage pool not found. Deprecated? "
                 " Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 " Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 " Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 " Too Many Connections - Exceeded the limit on number "
                 "     of parallel administrative operations allowed. "
                 " Internal Error - The SAN FS server encountered an "
                 "     internal error. Please see the message log and call "
                 "     Technical Support for resolution."),
```

*Figure 20. MOF STC_StoragePool class definition (Part 4 of 5)*

```
     ValueMap{ "0", "5", "13", "14", "15", "21", "22", "44", "61", "62", ".."},
     Values {"OK", "In Use", "Is Default", "Is Referenced", "Is System",
             "Not Found", "Not the Admin Master Server",
             "Incompatible Operation", "Cannot Connect to Server",
             "Too Many Connections", "Internal Error"}]
    uint32 Delete();

        [Description("Moves or renames a storage pool by creating a new "
                     "storage pool with the specified name and migrating "
                     "the data and capabilities to the new name.  If, "
                     "successful, the old storage pool will be removed."
                     "You cannot rename a SYSTEM storage pool. You cannot "
                     "use SYSTEM or DEFAULT for the new name. "
                     "The return codes are as follows. "
                     " Ok - Successful. "
                     " Invalid Name - The new name has invalid characters. "
                     " Is Default - DEFAULT is a reserved storage pool name "
                     "               as a new name. "
                     " Is System - Cannot rename SYSTEM storage pool or "
                     "             SYSTEM cannot be the new name. "
                     " Name Exists - There is another storage pool with the "
                     "                same name. "
                     " Not Found - Storage pool not found. Deprecated? "
                     " Not the Admin Master Server - This administrative "
                     "     operation can be invoked only on the master. "
                     " Incompatible Operation - Some other incompatible "
                     "     operation is currently running. Check the "
                     "     currently running administrative processes "
                     "     and try again. "
                     " Cannot Connect to Server - The admin server could not "
                     "     connect to the local SAN FS server. "
                     " Too Many Connections - Exceeded the limit on number "
                     "     of parallel administrative operations allowed. "
                     " Internal Error - The SAN FS server encountered an "
                     "     internal error. Please see the message log and call "
                     "     Technical Support for resolution."),
     ValueMap{ "0", "9", "13", "15", "18", "21", "22", "44", "61", "62", ".."},
     Values {"OK", "Invalid Name", "Is Default", "Is System", "Name Exists",
             "Not Found", "Not the Admin Master Server",
             "Incompatible Operation", "Cannot Connect to Server",
             "Too Many Connections", "Internal Error"}
    ]
    uint32 Move([IN] string NewName);
};
```

*Figure 20. MOF STC_StoragePool class definition (Part 5 of 5)*

**Related topics:**


- "STC_StoragePool" on page 137

## MOF STC_Volume class definition

```
/*
 *  STC_Volume
 *
 *  Modeling Issues:
 *    . Although the ID is the WWN and globally unique, we know about a
 *      volume once it is added to a storage pool. Therefore, we make a
 *      volume 'weak' w.r.t a storage pool.
 *    . This is not the same as CIM_StorageVolume?
 *    . Maybe we can use CIM_StorageExtent?
 *
 */

[Description ("In SAN FS, a volume is a unit of raw storage device providing "
              "space to a storage pool. One or more volumes can be added to a "
              "storage pool. On the host engine of the SAN.FS server, a "
              "volume device is also known as a LUN. You can use the "
              "STC_AvailableLUNs class for getting instances of all LUNs "
              "available on an host engine. "
              "As a SAN FS is made up of a cluster of engines, in order "
              "to operate properly, every engine of the cluster should be "
              "able to access the same LUN in order to add the LUN as a "
              "volume to a storage pool. "
              "The list of instances and the methods defined in this class are "
              "available only on the Master SAN FS server."),
 provider("com.ibm.storage.storagetank.provider.xnp.Volume")
]class STC_Volume: CIM_ManagedSystemElement
{
        [Description("Storage Pool to which this volume belongs."),
         Propagated("STC_StoragePool.Name")]
    string StoragePoolName;

        [Override ("Caption"),
         Description("A one-line caption.")]
    string Caption = "Storage Tank Volume";

        [Override ("Name"),
         Key, Alias ("VolumeName"), MaxLen(256),
         Description("A user-supplied label.")]
    string Name;

        [Description("The file path to the storage device."),
        MaxLen(256)]
    string OSDeviceName;
```

*Figure 21. MOF STC_Volume class definition (Part 1 of 7)*

```
        [Description("State of the volume. A value of zero indicates that the "
                 "volume is OK for reading, writing and allocation. By default "
                 "this is the state a volume will come to as soon as a LUN is "
                 "added to a storage pool. A state value of 1 (one) "
                 "indicates that allocation of new partitions to containers "
                 "is suspended. This does not prevent a client from reading "
                 "or writing to a volume. As long as there is enough space, "
                 "this state does not prevent a container from allocating a "
                 "new file on this volume. A value of 2 indicates that the "
                 "volume is processing a deletion request."),
         ValueMap {"0", "1", "2"},
         Values   {"OK", "Allocations Suspended", "Volume Being Deleted"}]
    uint32 State;

        [Units("MegaBytes"),
         Description("Size of the volume. ")]
    uint64 Size;

        [Units("MegaBytes"),
         Description("Current size of the volume used up by allocating to "
                     "containers.")]
    uint64 SizeAllocated;


    [Description("Current percentage of size allocated in the volume."),
     MinValue(0), MaxValue(100), Units("Percent")
    ]
    uint16 SizeAllocatedPercentage;


    [Override ("Description"),
     Write,
     MaxLen(256),
     Description("A description that is writable.")]
    string Description;

/*
 * Methods
 */
    [Description("Attach a volume to a Storage Pool."
                 "Constructor for the STC_Volume class."
                 "Explanation for return values - "
                 "  OK - Successful. "
                 "  Already Defined - The OSDeviceName has a label, "
                 "      IsForce is true, but it is already defined as "
                 "      another volume. "
                 "  In Use - The OSDeviceName has a label and IsForce is "
                 "      false. "
```

*Figure 21. MOF STC_Volume class definition (Part 2 of 7)*

```
                  "  Invalid Name - Volume name is invalid. "
                  "  Invalid Parameter - Deprecate? "
                  "  IO Failed  - OSDeviceName IO failed. "
                  "  Name Exists  - Volume name exists already. "
                  "  Not Found - OSDeviceName not found. "
                  "  Not the Admin Master Server - This administrative "
                  "     operation can be invoked only on the master. "
                  "  SP Not Found - Storage Pool not found "
                  "  Incompatible Operation - Some other incompatible "
                  "     operation is currently running. Check the "
                  "     currently running administrative processes "
                  "     and try again. "
                  "  Disk Not Viable - The volume does not hold at least "
                  "      one partition or "
                  "      Local OSDeviceName is not viable as a global "
                  "      disk or "
                  "      Hashing using WWN conflicts with an existing "
                  "      hash or "
                  "      Sector size < 512 or > 4096 or "
                  "      if a volume is added to the SYSTEM storage pool, "
                  "      sector sizes of all volumes is not the same. "
                  "  Access Denied - OSDeviceName access denied. "
                  "  Cannot Connect to Server - The admin server could not "
                  "      connect to the local SAN FS server. "
                  "  Too Many Connections - Exceeded the limit on number "
                  "      of parallel administrative operations allowed. "
                  "  Internal Error - The SAN FS server encountered an "
                  "      internal error. Please see the message log and "
                  "      call Technical Support for resolution."
                  "Note - If some errors occur (IO Failed, Disk Not Viable, "
                  "       Internal Error), decommission the disk by "
                  "       deleting the volume. It will be nice if at least "
                  "       the volume allocation is suspended by the "
                  "server."),
      Static,
      ValueMap{ "0", "3", "5", "9", "10", "12", "18", "21", "22", "28", "44",
                "52", "56", "61", "62", ".."},
      Values {"OK", "Already Defined", "In Use", "Invalid Name",
              "Invalid Parameter", "IO Failed", "Name Exists", "Not Found",
              "Not the Admin Master Server", "SP Not Found",
              "Incompatible Operation", "Disk Not Viable", "Access Denied",
              "Cannot Connect to Server", "Too Many Connections",
              "Internal Error"}]
   uint32 Create([IN, MaxLen(256)] string  OSDeviceName,
                 [IN, MaxLen(256)] string  StoragePoolName,
                 [IN, MaxLen(256)] string  VolumeName,
                 [IN, MaxLen(256)] string  Description,
                 [IN] boolean IsForce,
                 [IN] boolean IsSuspendAllocations);

      [Description("Suspend further partition allocations on the volume. "
                   "Explanation for return values - "
                   "  OK - Successful. "
                   "  Access Failed  - Deprecate? "
                   "  Not the Admin Master Server - This administrative "
                   "     operation can be invoked only on the master. "
                   "  Volume Not Found - Deprecate? "
```

*Figure 21. MOF STC_Volume class definition (Part 3 of 7)*

```
                  "  Allocations Already Suspended - Obvious. "
                  "  Incompatible Operation - Some other incompatible "
                  "     operation is currently running. Check the "
                  "     currently running administrative processes "
                  "     and try again. "
                  "  Access Denied - OSDeviceName access denied. "
                  "  Cannot Connect to Server - The admin server could not "
                  "      connect to the local SAN FS server. "
                  "  Too Many Connections - Exceeded the limit on number "
                  "      of parallel administrative operations allowed. "
                  "  Internal Error - The SAN FS server encountered an "
                  "      internal error. Please see the message log and "
                  "      call Technical Support for resolution."),
     ValueMap{"0", "2", "22", "33", "34", "44", "56", "61", "62", ".."},
     Values {"Ok", "Access Failed", "Not the Admin Master Server",
             "Volume Not Found", "Allocations Already Suspended",
             "Incompatible Operation", "Access Denied",
             "Cannot Connect to Server", "Too Many Connections",
             "Internal Error"}]
   uint32 SuspendAllocation();

       [Description("Resume suspended partition allocations on the volume. "
                  "Explanation for return values - "
                  "  OK - Successful. "
                  "  Access Failed  - Deprecate? "
                  "  Not the Admin Master Server - This administrative "
                  "     operation can be invoked only on the master. "
                  "  Volume Not Found - Deprecate? "
                  "  Allocations Are Not Suspended Before - Obvious. "
                  "  Incompatible Operation - Some other incompatible "
                  "     operation is currently running. Check the "
                  "     currently running administrative processes "
                  "     and try again. "
                  "  Access Denied - OSDeviceName access denied. "
                  "  Cannot Connect to Server - The admin server could not "
                  "      connect to the local SAN FS server. "
                  "  Too Many Connections - Exceeded the limit on number "
                  "      of parallel administrative operations allowed. "
                  "  Internal Error - The SAN FS server encountered an "
                  "      internal error. Please see the message log and "
                  "      call Technical Support for resolution."),
     ValueMap{"0", "2", "22", "33", "35", "44", "56", "61", "62", ".."},
     Values {"Ok", "Access Failed", "Not the Admin Master Server",
             "Volume Not Found", "Allocations Are Not Suspended Before",
             "Incompatible Operation", "Access Denied",
             "Cannot Connect to Server", "Too Many Connections",
             "Internal Error"}
   ]
   uint32 ResumeAllocation();

      [Description("Reset the Files-On-Volume iterator. "
                  "This method creates an iterator that fetches each file "
                  "entry that resides on this volume instance by creating "
```

*Figure 21. MOF STC_Volume class definition (Part 4 of 7)*

```
                  "and returning a FOVHandle. The FOVHandle is used as "
                  "input to the GetNextFOV method that fetches one "
                  "successive file entry per call. There can be only one "
                  "FOV iterator active at any " "given instance. "
                  "Explanation for return values - "
                  "  OK - Successful. "
                  "  Access Failed  - Deprecate? "
                  "  Insufficient Space - Not enough temporary space "
                  "     on the local disk for the list. "
                  "  Not Found - No files found on this volume. "
                  "  Not the Admin Master Server - This administrative "
                  "     operation can be invoked only on the master. "
                  "  Incompatible Operation - Some other incompatible "
                  "     operation is currently running. Check the "
                  "     currently running administrative processes "
                  "     and try again. "
                  "  Access Denied - OSDeviceName access denied. "
                  "  Cannot Connect to Server - The admin server could not "
                  "      connect to the local SAN FS server. "
                  "  Too Many Connections - Exceeded the limit on number "
                  "      of parallel administrative operations allowed. "
                  "  Serve State Offline - The volume has files of a "
                  "      container whose server is offline. "
                  "  Internal Error - The SAN FS server encountered an "
                  "      internal error. Please see the message log and "
                  "      call Technical Support for resolution."),
     ValueMap{"0", "2", "7", "21", "22", "44", "56", "61", "62", "65",
              ".."},
     Values { "Ok", "Access Failed", "Insufficient Space", "Not Found",
              "Not the Admin Master Server", "Incompatible Operation",
              "Access Denied", "Cannot Connect to Server",
              "Too Many Connections", "Serve State Offline",
              "Internal Error"},
     Expensive, ExecuteRole("Backup")]
uint32 ResetFOV([IN(false), OUT]string FOVHandle);

   [Description("Get the next Files-On-Volume entry, given the FOV "
                "iterator handle FOVHandle. The file entry is made "
                "available in the string FOVEntry. The iterator handle to "
                "use for the next call is placed in the FOVHandle, i.e., "
                "the FOVHandle is an IN/OUT parameter. "
                "Explanation for return values - "
                "  OK - Successful. "
                "  Not the Admin Master Server - This administrative "
                "     operation can be invoked only on the master. "
                "  End of Iteration - EOF is reached. "
                "  Invalid IterationIdentifier - Obvious. "
                "  Incompatible Operation - Some other incompatible "
                "     operation is currently running. Check the "
                "     currently running administrative processes "
                "     and try again. "
                "  Cannot Connect to Server - The admin server could not "
                "      connect to the local SAN FS server. "
                "  Too Many Connections - Exceeded the limit on number "
                "      of parallel administrative operations allowed. "
                "  Internal Error - The SAN FS server encountered an "
                "      internal error. Please see the message log and "
                "      call Technical Support for resolution."),
```

*Figure 21. MOF STC_Volume class definition (Part 5 of 7)*

```
        ValueMap{"0", "22", "37", "38", "44", "61", "62", ".."},
        Values { "Ok", "Not the Admin Master Server", "End of Iteration",
                 "Invalid IterationIdentifier", "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Internal Error"},
        Expensive, ExecuteRole("Backup")]

    uint32 GetNextFOV([IN, OUT]string FOVHandle,
                      [IN(false), OUT]string FOVEntry);

       [Description("Delete an existing volume.  If IsForce is false, "
                    "the volume is drained first by moving the file data that "
                    "resides on this volume to other volumes in the same "
                    "storage pool as this one. If the volume drain fails, the "
                    "volume will be put in the Allocations Suspended state "
                    "requiring manual administrative action. "
                    "The IsForce parameter specifies if the removal will  "
                    "happen even if the volume has files on it. "
                    "In this case, all the files that exist on this volume "
                    "partly or fully will be deleted in full before deleting "
                    "a volume. "
                    "If a volume in SYSTEM pool is deleted, the IsForce flag "
                    "is ignored and the volume is always drained. "
                    "Explanation for return values - "
                    "  OK - Successful. "
                    "  Access Failed  - Deprecate? "
                    "  In Use -  Volume drain failed for some reason and "
                    "            volume is not empty. "
                    "  IO Failed - Device IO failed. "
                    "  Not the Admin Master Server - This administrative "
                    "     operation can be invoked only on the master. "
                    "  Volume Not Found - Deprecate? "
                    "  Incompatible Operation - Some other incompatible "
                    "     operation is currently running. Check the "
                    "     currently running administrative processes "
                    "     and try again. "
                    "  Access Denied - OSDeviceName access denied. "
                    "  No Space - No space in other volumes of the "
                    "      storage pool for volume drain. "
                    "  Cannot Connect to Server - The admin server could not "
                    "      connect to the local SAN FS server. "
                    "  Too Many Connections - Exceeded the limit on number "
                    "      of parallel administrative operations allowed. "
                    "  Serve State Offline - The volume has files of a "
                    "      container whose server is offline. "
                    "  Internal Error - The SAN FS server encountered an "
                    "      internal error. Please see the message log and "
                    "      call Technical Support for resolution."),
```

*Figure 21. MOF STC_Volume class definition (Part 6 of 7)*

```
        ValueMap{"0", "2", "5", "12", "22", "33", "44", "56", "57",
                 "61", "62", "65", ".."},
        Values {"Ok", "Access Failed", "In Use", "IO Failed",
                "Not the Admin Master Server",
                "Volume Not Found", "Incompatible Operation", "Access Denied",
                "No Space", "Cannot Connect to Server", "Too Many Connections",
                "Serve State Offline", "Internal Error"}]
    uint32 Delete([IN] boolean IsForce);

        [Description("Move or rename a volume by creating a new volume and "
                     "migrating the data and capabilities of this volume to "
                     "it. If successful, the original volume will be deleted. "
                     "Explanation for return values - "
                     "  OK - Successful. "
                     "  Invalid Name - The new name has invalid characters. "
                     "  Not the Admin Master Server - This administrative "
                     "     operation can be invoked only on the master. "
                     "  Incompatible Operation - Some other incompatible "
                     "     operation is currently running. Check the "
                     "     currently running administrative processes "
                     "     and try again. "
         "  Cannot Connect to Server - The admin server could not "
                     "      connect to the local SAN FS server. "
                     "  Too Many Connections - Exceeded the limit on number "
                     "      of parallel administrative operations allowed. "
                     "  Internal Error - The SAN FS server encountered an "
                     "      internal error. Please see the message log and "
                     "      call Technical Support for resolution."),
        ValueMap{"0", "9", "22", "44", "61", "62", ".."},
        Values {"Ok", "Invalid Name", "Not the Admin Master Server",
                "Incompatible Operation", "Cannot Connect to Server",
                "Too Many Connections", "Internal Error"}]
    uint32 Move([IN] string NewName);
};
```

*Figure 21. MOF STC_Volume class definition (Part 7 of 7)*

**Related topics:**

- "STC_Volume" on page 156

## MOF STC_Container class definition

```
/*
 *  STC_Container
 *
 *  Modeling Issues:
 *
 *    . Candidate keys are ID and Name. Since Name can be changed, we use
 *      the ID as the key.
 *
 */
    [Description ("A container is a top level object that represent a set of "
              "files in a SAN FS Server. The latest nomenclature for a "
              "container is a fileset. A container is a group of files that "
              "can be managed administratively as a whole (delete, make it "
              "available to FS clients, take backup, decide placement etc). "
              " A policy based placement is used to assign space "
              "for files of a Container. The policy is determined by the "
              "current active policy set during file creation. "
              "Space is allocated for a container in units of partition sizes. "
              "The files of a container are accessible to SAN FS clients only "
              "if the container is attached. The files are available "
              "in the directory given by the attachpoint on the FS client "
              "machine. A container is assigned to a specific server "
              "for balancing the FS client request load. "
              "This class has an instance for every container that exists "
              "in a SAN FS. The list of instances and the methods defined in "
              "this class are available only on the Master SAN FS server."),
     provider("com.ibm.storage.storagetank.provider.xnp.Container")]
class STC_Container: CIM_ManagedSystemElement
{
        [Override ("Caption"), Description("Provide a one-line caption.")]
    string Caption = "Storage Tank Container";

        [Override ("Name"),
         Key,
         MaxLen(256),
         Description("A user-supplied label.")]
    string Name;

        [Override ("Description"),
         Write,
         MaxLen(256),
         Description("A description that is writable.")]
    string Description;

        [Override ("InstallDate"),
         Description("A datetime value when the container was created."),
         Alias("CreationDate")]
    datetime InstallDate;

         [Description("State of the container."),
         ValueMap {"0", "1"},
         Values   {"Detached", "Attached"}]
    uint32 State;
```

*Figure 22. MOF STC_Container class definition (Part 1 of 7)*

```
        [Description("Attach Point of this container in the filesystem "
                     "namespace. If the container is not attached, this "
                     "value is null."
                     "A container is made available to the file system using "
                     "a different name than the container name "
                     "called its directory name. A directory name is attached "
                     "to an existing directory path that can be another "
                     "container's attach point or a file directory. "
                     "An attach point is the combined path formed by the "
                     "directory path used to attach and the directory name.")]
    string AttachPoint = "";

        [Description("The name of the container as known to the filesystem. "
                     "This name will appear as a directory under the path "
                     "shown by DirectoryPath. This name is given to the "
                     "container at creation time and can be changed by "
                     "reattaching the container. The fully qualified "
                     "directory name along with its path is given by the "
                     "AttachPoint property."),
         ModelCorrespondence {"AttachPoint"}]
    string DirectoryName;

        [Description("The directory path under which a container will appear "
                     "to the file system. The name of the directory is as "
                     "indicated by the DirectoryName property. The fully "
                     "qualified directory name of this container is formed "
                     "combining the DirectoryPath value with the DirectoryName "
                     "property value and is given by the AttachPoint"
                     "property."),
         ModelCorrespondence {"AttachPoint"}]
    string DirectoryPath;

        [Description("The name of the parent container. If this is "
                     "the root container, the value is null.")
        ]
    string Parent = "";

        [Description("Number of immediate child containers.")]
    uint32 NumberOfChildren = 0;

        [Description("Current maximum size limit for the container. "
                     "A value of zero indicates that there is no limit."
                     "The default is no limit. The maximum value is "
                     "1024 PetaBytes."),
         Write,
         MinValue(0), MaxValue(1073741824), Units("MegaBytes")]
    uint64 Quota = 0;

   [Description("This property indicates whether a quota limit is a hard,"
                "quota, or a soft quota. This property is used only when a "
                "quota limit is specified and the container's allocated size "
                "reaches the quota limit. If this property value is true, the "
                "allocated size of the container will NOT be extended"
```

*Figure 22. MOF STC_Container class definition (Part 2 of 7)*

```
                "beyond the quota limit, and a severe alert message will be "
                "sent and logged in the server message log. If false, "
                "the allocated size of the container will be extended, and "
                "a warning alert message will be logged. "
                "Error conditions. "
                " . If you are changing from soft quota to hard quota by "
                "   setting this property from false to true, and if the "
                "   container already exceeds it's quota, you will get a "
                "   Hard Quota Violation (72) exception. "),
     Write]
    boolean IsHardQuota;

    [Description("Current percentage of the container's allocated "
                "size to quota that, when reached, causes the server to "
                "generate an alert message. The default value is 80%. "
                "An alert is generated only if all the following "
                "conditions are met. "
                " 1. A quota value greater than zero. "
                " 2. An alert percentage value greater than zero. "
                " 3. The size allocated percentage equals or exceeds the "
                "    current value of alert percentage. "),
     MinValue(0), MaxValue(100), Units("Percent"),
     Write
    ]
    uint16 AlertPercentage = 80;

        [Units("MegaBytes"), Gauge,
         Description("Size allocated to the container in MBytes. "
                     "The size allocated to the container may grow and shrink "
                     "as files are added and deleted in the container.")]
    uint64 SizeAllocated;

        [Description("Current percentage of size allocated compared to the "
                     "quota in the pool. This can be compared directly with "
                     "the AlertPercentage to determine how close the container "
                     "is to causing an alert."
                     "Note - For soft quota, the following values are "
                     "       possible. "
                     "     - A value of zero is reported if there is no limit "
                     "     - A value > 100%  if SizeAllocated > Quota "),
         MinValue(0), MaxValue(100), Units("Percent")]
    uint16 SizeAllocatedPercentage;

        [Description("Current number of existing PIT copies. There is a limit "
                     "on number of PIT copies that can be made.")]
    uint16 NumberOfPITCopies;

        [Description("The datetime of the last PIT copy.")]
    datetime LastPITCopyDate;

        [Description("The name of the server node hosting this container."),
         MaxLen(256)]
    string Server;

         [Description("State of the server serving the container."),
         ValueMap {"0", "1"},
         Values   {"Offline", "Online"}]
    uint32 ServerState;
```

*Figure 22. MOF STC_Container class definition (Part 3 of 7)*

```
/*
 * Methods
 */

    [Description("Define a new container. Constructor for this class. "
                 "Optionally, you can attach the container to a "
                 "attach point by supplying an existing directory path "
                 "and the new Directory name. "
                 "Explanation for return values - "
                 "  OK - Successful. "
                 "  Invalid Parameter - If NewDirName contains directory "
                 "     separators. "
                 "  Name Exists  - Container name exists already. "
                 "  Not Found - ExistingDirPath not found. "
                 "  Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 "  Transaction Failed - Some other parallel activity "
                 "     in the server caused this create to fail. Try "
                 "     again. "
                 "  Directory Exists - The name given in NewDirName exists."
                 "  Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 "  Server Not Found - Obvious "
                 "  No Space - The SAN.FS server ran out of space "
                 "      in system volumes where master metadata is "
                 "      stored. "
                 "  Cannot Connect to Server - The admin server could not "
                 "      connect to the local SAN FS server. "
                 "  Too Many Connections - Exceeded the limit on number "
                 "      of parallel administrative operations allowed. "
                 "  Server State Offline - The assigned server is "
                 "     not online for the attach to work. "
                 "     If it is a create without attach, should not occur. "
                 "  Internal Error - The SAN FS server encountered an "
                 "      internal error. Please see the message log and "
                 "      call Technical Support for resolution."),
     Static,
     ValueMap {"0", "10", "18", "21", "22", "30", "43", "44", "45", "57",
               "61", "62", "65", ".."},
     Values {"Ok", "Invalid Parameter", "Name Exists", "Not Found",
             "Not the Admin Master Server",
             "Transaction Failed", "Directory Exists",
             "Incompatible Operation", "Server Not Found", "No Space",
             "Cannot Connect to Server", "Too Many Connections",
             "Serve State Offline", "Internal Error"}
    ]
uint32 Create(
                [IN, MaxLen(256)] string Name,
                [IN, MaxLen(256)] string Description,
                [IN, MinValue(0), MaxValue(1073741824), Units("MegaBytes")]
                     uint64 Quota,
                [IN] boolean IsHardQuota,
                [IN, MinValue(0), MaxValue(100), Units("Percent")]
                     uint16 AlertPercentage,
                [IN, Description("An existing directory path to attach to.")]
                     string ExistingDirPath,
```

*Figure 22. MOF STC_Container class definition (Part 4 of 7)*

```
                [IN, Description("The new directory name to be given to "
                                 "the container.")]
                     string NewDirName,
                [IN, Description("The name of the server to host this "
                                 "container."), MaxLen(32)]
                     string Server
               );

      [Description("Attach an existing container to a filesystem namespace. "
                   "Reattach the container if it is already attached. Note "
                   "that you can change the directory path as well as the "
                   "the directory name for reattach. "
                   "The return codes are as follows. "
                   "  OK - Success "
                   "  Not Found - Path as given in ExistingDirPath not "
                   "              found. "
                   "  Not Viable - For reattach, the new path for the "
                   "               container to be attached already "
                   "               contains the original attach point name "
                   "               for the container"
                   "  Is Attached - The container is already attached."
                   "  Name Exists - The directory name given in NewDirName "
                   "                already exists. "
                   "  Invalid Name - The directory name is invalid. "
                   "  Server State Offline - The serving server is offline. "
                   "  Cannot connect to Server - The Admin Server could "
                   "      not contact the local MDS server. "
                   ),
       ValueMap {"0", "9", "18", "21", "22", "23", "30", "36", "44",
                 "61", "62", "65", ".."},
       Values {"Ok", "Invalid Name", "Name Exists", "Not Found",
               "Not the Admin Master Server", "Not Viable",
               "Transaction Failed", "Is Attached",
               "Incompatible Operation", "Cannot Connect to Server",
               "Too Many Connections", "Serve State Offline",
               "Internal Error"}
      ]
   uint32 Attach([IN, Description("An existing directory path to attach to.")]
                       string ExistingDirPath,
                  [IN, Description("The directory name of the container.")]
                       string NewDirName
                 );

      [Deprecated {"STC_Container.Attach"},
       Description("This method is deprecated as the Attach method can "
                   "function as a reattach method. "
                   "ReAttach an attached container to a new filesystem "
                   "namespace. This operation is not disruptive to "
                   "a filesystem client."),
       ValueMap {"0", "9", "18", "21", "22", "61", "62", "44", ".."},
       Values {"Ok", "Invalid name", "Name Exists", "Not Found",
               "Not the Admin Master Server",
               "Cannot Connect to Server", "Too Many Connections",
               "Incompatible Operation", "Internal Error"}
      ]
```

*Figure 22. MOF STC_Container class definition (Part 5 of 7)*

```
    uint32 ReAttach([IN, Description("An existing directory path to attach.")]
                        string ExistingDirPath,
                    [IN, Description("The directory name of the container.")]
                        string NewDirName
                   );

        [Description("Detach a container from a filesystem namespace. You "
                     "cannot detach a container if any child containers are "
                     "attached to it. You cannot detach a container with "
                     "clients using files on it unless the IsForce option "
                     "is set to true. The attach point is not persisted. "
                     "Once you detach, the directory path and directory name "
                     "are lost. Return Codes are as follows. "
                     "  Is Referenced - IsForce is false and clients are "
                     "                  using files in the container. "),
         ValueMap {"0", "5", "14", "20", "21", "22", "27", "30", "44", "61",
                   "62", "65", ".."},
         Values {"Ok", "In Use", "Is Referenced", "Not Attached", "Not Found",
                 "Not the Admin Master Server", "Is Root", "Transaction Failed",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Serve State Offline",
                 "Internal Error"}
        ]
    uint32 Detach([IN] boolean IsForce);

        [Description("Delete a container. You cannot delete a container "
                     "that is attached. You cannot delete the ROOT "
                     "container. You cannot delete a container "
                     "that has files on it unless the IsForce option "
                     "is set to true. "
                     "The return codes are as follows. "
                     "  Ok - Successful. "
                     "  In Use - Container is not empty of files and IsForce "
                     "           is false. "
                     "  Is Referenced - Container is referenced in an "
                     "                  active policy rule. "
                     "  Is Root - Container is the ROOT container. "
                     ),
         ValueMap {"0", "5", "14", "21", "27", "22", "30", "36", "44",
                   "61", "62", "65", ".."},
         Values {"Ok", "In Use", "Is Referenced", "Not Found", "Is Root",
                 "Not the Admin Master Server", "Transaction Failed",
                 "Is Attached", "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Serve State Offline", "Internal error"}
        ]
    uint32 Delete([IN] boolean IsForce);
```

*Figure 22. MOF STC_Container class definition (Part 6 of 7)*

```
        [Description("Change the server hosting this container. "
                     "The specified server acting as the new host must be "
                     "part of the cluster. The cluster and the current server "
                     "must be either online, or in one of the administrative "
                     "modes for the change to succeed. The current host server "
                     "can be down. If the current host server is not down, "
                     "then the cluster and the current server must be in one "
                     "of the administrative modes. The new server can be in "
                     "any state but must be part of the cluster."),
         ValueMap {"0", "3", "10", "21", "22", "27", "30",  "44", "45",
                   "46", "61", "62", ".."},
         Values {"Ok", "Already Defined", "Invalid Parameter", "Not Found",
                 "Not the Admin Master Server", "Is Root", "Transaction Failed",
                 "Incompatible Operation", "Server Not Found",
                 "Invalid Cluster State", "Cannot Connect to Server",
                 "Too Many Connections", "Internal error"}
        ]
    uint32 ChangeServer([IN, Description("The name of the server to host this "
                                         "container."), MaxLen(32)]
                        string Server);

        [Description("Move or rename a container by creating a new container "
                     "and migrating the data and capabilities of this "
                     "container to. If successful, the original container "
                     "will be deleted."),
         ValueMap{"0", "9", "18", "21", "22", "30", "44", "61", "62", ".."},
         Values  {"Ok", "Invalid Name", "Name Exists", "Not found",
                  "Not the Admin Master Server", "Transaction Failed",
                  "Incompatible Operation", "Cannot Connect to Server",
                  "Too Many Connections", "Internal Error"}
        ]
    uint32 Move([IN, MaxLen(256)] string NewName);
};
```

*Figure 22. MOF STC_Container class definition (Part 7 of 7)*

**Related topics:**

- "STC_Container" on page 98

## MOF STC_PitImage class definition

```
/*
 *  STC_PitImage
 */
    [Description (" STC_PitImage class represent the point-in-time image "
                  "copies of a StorageTank container. The instances of this "
                  "class contain all the PIT images of all the containers in a "
                  "StorageTank. The methods include creating a PIT image, "
                  "deleting an existing PIT image, and reverting a container "
                  "to an existing PIT image. The space used by a PIT image "
                  "is accounted in the space used by a container for quota "
                  "calculations. When A PIT image is created, it does not "
                  "use any space. PIT images use space when files within "
                  "the container are modified after a PIT image is taken."),
     provider("com.ibm.storage.storagetank.provider.xnp.PitImage")]
class STC_PitImage: CIM_ManagedSystemElement
{
        [Override ("Caption"), Description("Provide a one-line caption.")]
    string Caption = "Storage Tank Container PIT Image";

        [Description("Container to which this PIT image belongs."),
         Key, MaxLen(256),
         Propagated("STC_Container.Name")
        ]
    string ContainerName;

        [Override ("Name"),
         Key, MaxLen(256),
         Description("A user-supplied administrative name for the PIT image.")]
    string Name;

        [Override ("Description"),
         MaxLen(256),
         Description("A user-supplied description.")]
    string Description;

        [Override ("InstallDate"),
         Description("The datetime value when this PIT image was created."),
         Alias("CreationDate")]
    datetime InstallDate;

        [Description("The directory name under which this PIT image will "
                     "appear. The full path for the PIT image in the"
                     "file system is given by AttachPoint/.pit/DirectoryName "
                     "where AttachPoint is the attach point of the container.")
        ]
    string DirectoryName;
```

*Figure 23. MOF STC_PitImage class definition (Part 1 of 3)*

```
/*
 * Methods
 */

    [Description("Create a new PIT image for a container. Constructor for "
                 "this class. There is a limit (32) on the number of PIT "
                 "images that can exist at any given time. When this "
                 "limit is reached, a create will fail unless the IsForce "
                 "flag is set to true. In this case, the oldest PIT image "
                 "will be deleted using the IsForce for delete also set to "
                 "true. Please see the description for the Delete method "
                 "for the explanation of the IsForce flag for delete. The "
                 "container can be attached or detached when the PIT image "
                 "copy is taken. The return codes are as follows. "
                 " OK  - Successful                              "
                 " Invalid Parameter -  The PIT Name length is > maximum, "
                 "    or Description length is > maximum, "
                 "    or DirectoryName length is > maximum, "
                 "    or DirectoryName contains directory separators. "
                 " Name Exists - The PIT Name already exists for the "
                 "    container. "
                 " Directory Exists - DirectoryName already exists for "
                 "    another PIT image for the same container. "
                 " Not Found - Container not found or no server serving "
                 "             the container. "
                 " Not the Admin Master Server - Obvious "
                 " Transaction Failed - Other concurrent activity in the "
                 "     server caused this create to fail. "
                 " Incompatible Operation - The server is executing an "
                 "     incompatible operation to this create."
                 " Table Full - The number of PIT images taken are "
                 "     at the maximum limit already. "
                 " Internal Error - Any other errors. "),
     Static,
     ExecuteRole("Backup"),
     ValueMap {"0", "10", "18", "21", "22", "30", "42", "43", "44", "61",
               "62", "65", ".."},
     Values {"Ok", "Invalid Parameter", "Name Exists", "Not Found",
             "Not the Admin Master Server", "Transaction Failed",
             "Table Full", "Directory Exists", "Incompatible Operation",
             "Cannot Connect to Server", "Too Many Connections",
             "Serve State Offline", "Internal error"}
    ]
uint32 Create(
                [IN, MaxLen(256)] string ContainerName,
                [IN, MaxLen(256)] string Name,
                [IN, MaxLen(256)] string Description,
                [IN, Description("The new directory name to be given to "
                                 "the PIT Image.")]
                     string DirectoryName,
                [IN] boolean IsForce
              );
```

*Figure 23. MOF STC_PitImage class definition (Part 2 of 3)*

```
        [Description("Revert the container to this instance of the PIT image. "
                     "You cannot revert a container that has child containers. "
                     "Detach child containers, if any, manually. "
                     "A revert involves deleting all the more recent PIT "
                     "images than this instance, including the current "
                     "container image. If there is any client activity "
                     "(session locks open) in the PIT images (including the "
                     "current container) to be deleted, and also in the "
                     "current PIT instance, you cannot revert a container "
                     "to this instance, unless the IsForce option is true."
                     "In other words, the client activity can continue only "
                     "in PIT images taken earlier than this instance. "
                     "The return codes are as follows. "
                     " OK - Successful. "
                     " In Use - Client session locks are open and IsForce "
                     "          option is not true. "
                     " Is Attached - Container has child containers. "
                     " Partial Data - The PIT Image contains incomplete "
                     "     files. Hence, the container is not reverted. "
                     " Not Found - PIT Image not found or Container not found, "
                     "             or no server serving the container. "
                     " Not the Admin Master Server - Obvious."),
         ValueMap {"0", "5", "21", "22", "30", "36", "41", "44", "61", "62",
                   "65", ".."},
         Values {"Ok", "In Use", "Not Found", "Not the Admin Master Server",
                 "Transaction Failed", "Is Attached", "Partial Data",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Serve State Offline",
                 "Internal Error"}
        ]
    uint32 Revert([IN] boolean IsForce);

        [Description("Delete a PIT image of a container. You cannot delete a "
                     "PIT image that has client activity (session locks open) "
                     "unless the IsForce option is set to true. If the "
                     "IsForce flag is set to true, all the client activity "
                     "is terminated (session locks revoked) before delete. "
                     "The meaning of the return codes are the same as for "
                     "the Revert method."),
         ExecuteRole("Backup"),
         ValueMap {"0", "5", "21", "22", "30", "44", "61", "62", "65", ".."},
         Values {"Ok", "In Use", "Not Found", "Not the Admin Master Server",
                 "Transaction Failed", "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Serve State Offline", "Internal Error"}
        ]
    uint32 Delete([IN] boolean IsForce);
};
```

*Figure 23. MOF STC_PitImage class definition (Part 3 of 3)*

**Related topics:**

- "STC_PitImage" on page 127

## MOF STC_PolicySet class definition

```
/*
 *  STC_PolicySet
 *
 *  Modeling Issues:
 *  . We can model policy sets using CIM_PolicyRule, CIM_VendorPolicyCondition,
 *    VendorPolicyActions etc. Unfortunately, we are not able to do so as
 *    StorageTank does not operate at this level of granularity. Instead,
 *    we just deal with policy set 'blobs'.
 *
 *  . The TLO (Top Level Object) is STC_Cluster for SystemCreationClassName,
 *    and the SystemName.
 *
 *  . TBD KH - Determine if there is a limit on the size of the PolicyRules and
 *             set it for the property and the parameter for create.
 *  . Instead of a string PolicyRules property, we provide a GetRules() method
 *    to get a text 'blob' of rules.
 *      . This will enable a CIM client to get (and display) rules separately
 *        from the other properties of a policy.
 *      . We can handle the case where there is an error fetching the text
 *        from the server.
 *
 *  . Note that there is no Move method to rename a policy.
 *
 */
    [Description ("A Policy Set is a set of rules that assigns files "
                  "of a container to specific storage pools at creation "
                  "time. This class has an instance for each policy set "
                  "defined. Although multiple policy sets can exist in the "
                  "system, only one policy set can be active. The system "
                  "defines a default policy set that assigns files to default "
                  "storage pool."),
     provider("com.ibm.storage.storagetank.provider.xnp.PolicySet")]
class STC_PolicySet: CIM_PolicySet
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.")
        ]
    string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.")
        ]
    string SystemName;

        [Key, MaxLen (256), Description (
        "CreationClassName indicates the name of the class or the "
        "subclass used in the creation of an instance. When used "
        "with the other key properties of this class, this property "
        "allows all instances of this class and its subclasses to "
        "be uniquely identified.") ]
    string CreationClassName;
```

*Figure 24. MOF STC_PolicySet class definition (Part 1 of 4)*

```
    [Key, MaxLen (256),
     Description ("A user-friendly name of this PolicySet.")
    ]
string Name;

    [Description (
    "Indicates whether this PolicySet is administratively "
    "active, or administratively not active. Only one policy set "
    "can be active at any time. The default value is 0."),
    ValueMap { "0", "1" },
    Values { "NotActive", "Active" }
    ]
uint16 State;

    [Description("The set of policy rules belonging to this policy set.")]
string PolicyRules;

[Override ("Description"),
 MaxLen(256),
 Description("A description given by the user.")]
string Description = "";

    [Description("Date and time when this policy set was created.")]
datetime CreationDate;

    [Description("Date and time when this policy set rules were last "
                 "modified. If the rules were never modified, this value "
                 "will be the same as the creation date. The policy rules "
                 "can be modified as a whole by creating a policy "
                 "set with the same name and using the force option.")]
datetime LastModificationDate;

    [Description("Date and time when this policy set was last active. "
                 "This is actually the data and time when another policy "
                 "set was made active (enabled) instead of this one. If "
                 "the policy set was never activated, the value is null. "
                 "If this policy set is currently active (enabled), this "
                 "value is null.")]
datetime LastActiveDate;

    [Description("Create a new policy set. Constructor. If the IsForce "
                 "option is set to true, then an existing policy set "
                 "will be modified."
                 "If there is a Policy Syntax Error, the Errors array "
                 "output parameter will have further information "
                 "where the syntax error occurred. "
                 "The return codes are as follows."
                 "  OK - Successful. "
                 "  Already Defined - Another policy set with the same "
                 "    name exists and the IsForce flag is false. "
```

*Figure 24. MOF STC_PolicySet class definition (Part 2 of 4)*

```
                  "  In Use - The policy set you are attempting to modify "
                  "     is active. "
                  "  Invalid Name - The name has invalid characters. "
                  "  Is Default - The name is DEFAULT_POLICY and the "
                  "    IsForce flag is true. Cannot overwrite default "
                  "    policy. If IsForce is false, Already Defined will "
                  "    be returned instead. "
                  "  Not Found - Deprecate? "
                  "  Not the Admin Master Server - This administrative "
                  "     operation can be invoked only on the master. "
                  "  Policy Syntax Error - Syntax error in the policy "
                  "    rules. The Errors string array output parameter "
                  "    for more information on what and where the error "
                  "    occurred. "
                  "  Incompatible Operation - Some other incompatible "
                  "     operation is currently running. Check the "
                  "     currently running administrative processes "
                  "     and try again. "
                  "  Cannot Connect to Server - The admin server could not "
                  "      connect to the local SAN FS server. "
                  "  Too Many Connections - Exceeded the limit on number "
                  "      of parallel administrative operations allowed. "
                  "  Internal Error - The SAN FS server encountered an "
                  "      internal error. Please see the message log and "
                  "      call Technical Support for resolution."),
      Static,
      ValueMap {"0", "3", "5", "9", "13", "21", "22", "26", "44", "61", "62",
                ".."},
      Values {"Ok", "Already Defined", "In Use", "Invalid Name",
              "Is Default", "Not Found", "Not the Admin Master Server",
              "Policy Syntax Error", "Incompatible Operation",
              "Cannot Connect to Server", "Too Many Connections",
              "Internal Error"}
     ]
  uint32 Create(
             [IN, MaxLen(256)] string Name,
             [IN, MaxLen(256)] string Description,
             [IN] string PolicyRules,
             [IN] boolean IsForce,
             [IN(false), OUT] string Errors[] );
```

*Figure 24. MOF STC_PolicySet class definition (Part 3 of 4)*

```
        [Description("Delete an existing stored policy set. The policy set "
                     "must be inactive for delete to succeed. "
                     "The return codes are as follows. "
                     "  OK - Successful. "
                     "  In Use - The policy set you are attempting to modify "
                     "     is active. "
                     "  Is Default - The name is DEFAULT_POLICY and the "
                     "    IsForce flag is true. Cannot overwrite default "
                     "    policy. If IsForce is false, Already Defined will "
                     "    be returned instead. "
                     "  Not Found - Deprecate? "
                     "  Not the Admin Master Server - This administrative "
                     "     operation can be invoked only on the master. "
                     "  Incompatible Operation - Some other incompatible "
                     "     operation is currently running. Check the "
                     "     currently running administrative processes "
                     "     and try again. "
                     "  Cannot Connect to Server - The admin server could not "
                     "      connect to the local SAN FS server. "
                     "  Too Many Connections - Exceeded the limit on number "
                     "      of parallel administrative operations allowed. "
                     "  Internal Error - The SAN FS server encountered an "
                     "      internal error. Please see the message log and "
                     "      call Technical Support for resolution."),
         ValueMap {"0", "5", "13", "21", "22", "44", "61", "62", ".."},
         Values {"Ok", "In Use", "Is Default", "Not Found",
                 "Not the Admin Master Server",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Internal error"}
        ]
    uint32 Delete();

        [Description("Activate a stored policy set"),
         ValueMap{"0", "21", "22", "25", "44", "61", "62", ".."},
         Values {"Ok", "Not Found", "Not the Admin Master Server",
                 "Policy Bind Errors", "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Internal Error"}
        ]
    uint32 Activate([IN(false), OUT] string Errors[]);

        [Description("Retrieve the set of rules associated with this policy."),
         ValueMap{"0", "21", "22", "44", "61", "62", ".."},
         Values {"Ok", "Not Found", "Not the Admin Master Server",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Internal Error"},
        ExecuteRole("Monitor")]
    uint32 GetRules([IN(false), OUT] string RulesList);
};
```

*Figure 24. MOF STC_PolicySet class definition (Part 4 of 4)*

**Related topics:**

- "STC_PolicySet" on page 131

## MOF STC_AvailableLUNs class definition

```
/*
 * STC_AvailableLUNs
 *
 * We will use the the inherited CIM_LogicalDevice:DeviceID key to have the
 * OSDeviceName for the LUN.
 *
 */
    [Description("Available FC LUNs. These are the storage volumes exposed "
                 "via the SCSI LUNs on the nodes of the cluster."),
     provider("com.ibm.storage.storagetank.provider.std.AvailableLUNs")]
class STC_AvailableLUNs: CIM_StorageVolume
{
        [Description("The LUN Id."),
         MaxLen(32)]
    string LunID;

        [Description("The Node WWN providing the LUN.  A 16-digit hex number "
                     "of the form xx:xx:xx:xx:xx:xx:xx:xx"),
        MaxLen(23)]
    string NodeWWN;

        [Description("The Port WWN on the Node that is providing the LUN. "
                     "A 16-digit hex number of the form "
                     "xx:xx:xx:xx:xx:xx:xx:xx."),
         MaxLen(23)]
    string PortWWN;

        [Description("The vendor name supplying the product."),
         MaxLen(256)]
    string Vendor;

        [Description("The product name."),
         MaxLen(256)]
    string Product;

        [Description("The version of the product."),
         MaxLen(256)]
    string Version;

        [Description("Storage size of the LUN."),
         Units("MegaBytes")]
    uint64 Size;
```

*Figure 25. MOF STC_AvailableLUNs class definition (Part 1 of 2)*

```
        [Description("Availability state of this LUN as a volume. "
                     "  Available - This LUN is available to be added as a "
                     "              volume. "
                     "  Assigned  - This LUN is already assigned to the "
                     "              StorageTank as a volume. The VolumeName "
                     "              property value further identifies the "
                     "              specific volume. "
                     "  Error     - There was an error determining the "
                     "              properties of the LUN. "
                     "  Unknown   - The StorageTank server is not running. "
                     "              Cannot determine the availability of the "
                     "              LUN. "
                     "  Unusable  - This LUN is unsuitable as a volume. One "
                     "              reason the LUN is not suitable is that the "
                     "              (inherited) Access property shows "
                     "              the LUN does not support read/write. "
                     "              Other reasons include inconsistant "
                     "              availability of this LUN from all nodes "
                     "              of the cluster."),
         ValueMap {"0", "1", "2", "3", "4"},
         Values {"Available", "Assigned", "Error", "Unknown", "Unusable"}]
    uint32 State;

        [Description("If the LUN is already assigned to a Storage Tank storage "
                     "pool, this is the volume name that was given when the "
                     "LUN was added. Otherwise, the value will be null."),
         MaxLen(256)]
    string VolumeName;
};
```

*Figure 25. MOF STC_AvailableLUNs class definition (Part 2 of 2)*

**Related topics:**

- "STC_AvailableLUNs" on page 95

## MOF STC_Cluster class definition

```
/*
 *  STC_Cluster
 *
 *  This class together with the STC_MasterService provides the cluster
 *  operations.
 *
 *  Modeling Issues:
 *
 *    . Use CIM_Cluster.State to STC_Cluster.CurrentState ModelCorrespondence?
 *      Mapping:
 *        "Unknown"     <-
 *        "Other"       <- CurrentState
 *        "On-line"     <- "Online"
 *        "Off-line",   <- "Down"
 *        "Degraded",   <- "PartlyQuiescent"
 *        "Unavailable" <- "FullyQuiescent", "AdminQuiescent"
 *
 */
    [Description ("STC_Cluster"),
     provider("com.ibm.storage.storagetank.provider.xnp.Cluster")]
class STC_Cluster: CIM_Cluster
{
        [Key, MaxLen(256), Override("CreationClassName"),
         Description("Set the value as this class.")]
    string CreationClassName = "STC_Cluster";

         [Description("A unique integer Id for the cluster.")]
    uint32 ClusterId;

        [Description("Configured number of nodes.")]
    uint32 ConfiguredNumberOfNodes;

        [Description("Current number of nodes participating in the cluster. "
                     "If this number is not the same as the configured "
                     "number, then the cluster is operating in a degraded "
                     "mode.")]
    uint32 CurrentNumberOfNodes;
};
```

*Figure 26. MOF STC_Cluster class definition*

**Related topics:**

- "STC_Cluster" on page 96

## MOF STC_MasterService class definition

```
/*
 * STC_MasterService
 *
 * The Top Level Object (TLO) for this is the STC_Cluster class.
 *
 * We use the inherited StartService() and StopService() methods from
 * CIM_Service to mean Stop Cluster and Start Cluster respectively.
 *
 */
    [Description ("Cluster services to start and stop a cluster are provided "
                  "by this class."),
     provider("com.ibm.storage.storagetank.provider.xnp.MasterService")]
class STC_MasterService : CIM_ClusteringService
{
        [Description("Indicates the StorageTank specific state of the cluster. "
                  "The cluster can be down (value=0), "
                  "Online (value=1), "
                  "put in a partly quiescent state where "
                  "only server (metadata) I/O operations are suspended "
                  "(value=2), "
                  "put in a full quiescent state where all "
                  "background IO, client, and server operations are "
                  "suspended (value=3), "
                  "put in a administrative state no longer servicing "
                  "clients (value=4), "
                  "forming a cluster (value=5), "
                  "or not the master server anymore (value=6). "
                  "A value of Unkown (value=7) is used when the SAN.FS "
                  "cluster master could not be contacted to determine "
                  "the state. The probable reasons for this would be "
                  "that the master is down, network partition, etc."),
         ValueMap {"0", "1", "2", "3", "4", "5", "6", "7"},
         Values {"Down", "Online", "PartlyQuiescent", "FullyQuiescent",
                 "AdminQuiescent", "Forming", "NotMaster", "Unknown"}]
    uint32 CurrentState;

        [Description("The cluster current state will be transitioning to "
                     "this state if the current state is different from "
                     "from this pending state."),
         ValueMap {"0", "1", "2", "3", "4", "5", "6", "7"},
         Values {"Down", "Online", "PartlyQuiescent", "FullyQuiescent",
                 "AdminQuiescent", "Forming", "NotMaster", "Unknown"},
         ModelCorrespondence {"CurrentState"}]
    uint32 PendingState;

        [Description ("Time (delta) since the cluster changed its current "
                      "state.")]
    datetime LastCurrentStateChangeTime;

        [Description ("Time (delta) since the cluster has a state change "
                      "pending. ")]
    datetime LastPendingStateChangeTime;

        [Description("Software release version that was committed.")]
    string CommittedVersion;

        [Description("Timestamp when the latest upgrade was committed.")]
    datetime CommittedUpgradeTimestamp;
```

*Figure 27. MOF STC_MasterService class definition (Part 1 of 6)*

```
    [Description("Current software release version. This is the version "
                 "of the latest upgrade done. This will be different "
                 "from the previous (committed) version if there was "
                 "an upgrade done but the commit was not yet activated.")]
string CurrentVersion;

    [Description("Boolean indicating if an upgrade is in progress. After "
                 "a version is committed, the system may take a while to "
                 "sync up all the internal datastructure versions. This "
                 "property indicates if such change is still in progress.")]
boolean IsUpgradeInProgress;

/*
 * Methods
 */

    [Override("StartService"),
     Description("Bring up all pre-commissioned servers on all nodes. "
                 "This results in a fully online cluster. "
                 "The return codes are as follows. "
                 "  Ok - Successful. "
                 "  Not Supported - This is to comply with the parent "
                 "    definition in the MOF. Never returned. "
                 "  Command Failed - Obvious! "
                 "  In Use - The cluster is running already. This means "
                 "    the master SAN.FS server is running. "
                 "  Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 "  Server Timedout - The master server or any of the  "
                 "     other servers in the cluster was launched "
                 "     successfully but failed to come online after "
             "     a maximum wait period (for each server). "
                 "  Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 "  Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 "  Cannot Persist Watchdog State - This is only a "
                 "    warning and the cluster is started successfully. "
                 "    The watchdog state could not be persisted for some "
                 "    reason. "
                 "  Aborted - One or more servers (including the master) "
                 "    has aborted instead of coming up. "
                 "  Internal Error - The SAN FS server encountered an "
                 "     internal error. Please see the message log and call "
                 "     Technical Support for resolution."),
     ValueMap{"0", "1", "4", "5", "22", "24", "44", "61", "66", "78", ".."},
     Values {"Ok", "Not Supported", "Command Failed", "In Use",
             "Not the Admin Master Server", "Server Timedout",
             "Incompatible Operation", "Cannot Connect to Server",
             "Cannot Persist Watchdog State", "Aborted", "Internal Error"}]
uint32 StartService();
```

*Figure 27. MOF STC_MasterService class definition (Part 2 of 6)*

```
        [Override("StopService"),
         Description("Gracefully bring down servers on all nodes of "
                     "a cluster."),
         ValueMap{"0", "1", "4", "22", "24", "44", "61", "62", "66", ".."},
         Values {"Ok", "Not Supported", "Command Failed",
                 "Not the Admin Master Server", "Server Timedout",
                 "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Cannot Persist Watchdog State", "Internal Error"}]
    uint32 StopService();

        [Description ("Put the cluster in a Quiesce state so that "
                      "(i) backup and restore operations can be performed "
                      " on the SAN. "
                      "Two types of Quiesce for backups are possible - "
                      "A full quiesce mode that suspsends all client metadata "
                      "activity, file data activity and terminates all client "
                      "sessions. This allows a backup with metadata as well "
                      "as file data integrity intact. "
                      "A limited quesce mode that allows client file data "
                      "activity to continue but prevents client metadata "
                      "activity and also prevents new client connections. This "
                      "allows a backup with metadata integrity intact but may "
                      "not preserve file data integrity. "
                      "(ii)Administrative operations that does not permit "
                      "client activity can be performed safely."
                      "Notes on quiscse state - "
                      "The cluster may go out of the quiesce state if there "
                      "is any cluster reformation when already in quiesce "
                      "state. All the servers that currently belong to the "
                      "cluster are brought into the Quiesce state."),
         ValueMap{"0", "1", "4", "10", "22", "44", "61", "62", ".."},
         Values {"Ok", "Not Supported", "Command Failed",
                 "Invalid Parameter", "Not the Admin Master Server",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Internal Error"}]
    uint32 QuiesceService(
        [IN, ValueMap {"0", "1", "2"},
             Values {"PartlyQuiescent", "FullyQuiescent", "AdminQuiescent"}]
        uint32 Mode);

        [Description("Resume the cluster service from the quiescent state "
                     "to a fully online state."),
         ValueMap{"0", "1", "4", "22", "44", "61", "62", ".."},
         Values {"Ok", "Not Supported", "Command Failed",
                 "Not the Admin Master Server",
                 "Incompatible Operation", "Cannot Connect to Server",
                 "Too Many Connections", "Internal Error"}]
    uint32 ResumeService();
```

*Figure 27. MOF STC_MasterService class definition (Part 3 of 6)*

```
    [Description("Check and repair the file system meta-data. You can "
                 "restrict this operation to check-only by setting the "
                 "IsCheckOnly option to true. You can check the "
                 "integrity of the structure of the meta-data and the "
                 "content of the meta-data. You can check the integrity "
                 "of the system meta-data and the user (container) "
                 "meta-data. You can also limit the user meta-data "
                 "checking to a subset of containers. "
                 "Please see the message log to see the report generated "
                 "by this method. If you did not limit the mode to "
                 "check-only (if you choose the repair option), the system "
                 "will automatically salvage and repair the damaged data "
                 "if possible. Some types of repair need manual "
                 "administrator intervention. In those cases, the cluster "
                 "state will be put into Administrative mode. "
                 "Return codes are as follows. "
                 " OK - Successful. "
                 " Is Use - Another FileSystemCheck operation is active. "
                 " Integrity Lost -  Check-only is requested and "
                 "     the server has detected corruption. "
                 " Invalid Parameter - Can't be more specific. "
                 " Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 " Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 " Canceled - Canceled due to StopFileSystemCheck"
                 " Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 " Too Many Connections - Exceeded the limit on number "
                 "     of parallel administrative operations allowed. "
                 " Salvaged - Salvage was requested, corruption was "
                 "   detected and repaired successfully. This is a success "
                 "   return code and not an error. "
                 " Cancel Pending- Another FileSystemCheck operation is "
                 "   active, but a stop has been issued and the cancel "
                 "   is pending. "
                 " Salvage Failed - Repair was requested, the server "
                 "     attempted to salvage but failed. "
                 " Internal Error - The SAN FS server encountered an "
                 "     internal error. Please see the message log and call "
                 "     Technical Support for resolution."),
     Expensive,
     ValueMap {"0", "5", "8", "10", "22", "44", "60", "61", "62", "73",
               "76", "77", ".."},
     Values   {"OK", "In Use", "Integrity Lost", "Invalid Parameter",
               "Not the Admin Master Server", "Incompatible Operation",
               "Canceled", "Cannot Connect to Server",
               "Too Many Connections", "Salvaged", "Cancel Pending",
               "Salvage Failed", "Internal Error"}]
  uint32 FileSystemCheck(
              [Description("A boolean indicating if it is a check-only "
                           "(no repair).")]
   boolean IsCheckOnly,
```

*Figure 27. MOF STC_MasterService class definition (Part 4 of 6)*

```
            [Description("A bitmap indicating the scope of the check. "
                         "A bit value of Structure checks the structure "
                         "of the meta-data. A bit value of content "
                         "checks the contents of the meta-data. You "
                         "can set both the bits to check the structure "
                         "and content."),
             BitMap { "0", "1" },
             BitValues { "Structure", "Content" }]
        uint16 CheckScope,
            [Description("A bitmap indicating the type of the meta-data "
                         "to be checked. A bit value of System checks the "
                         "system meta-data. A bit value of User checks "
                         "the user (container) meta-data. You can set "
                         "both the bits to check the system and user "
                         "meta-data."),
             BitMap { "0", "1" },
             BitValues { "System", "User" }]
        uint16 Type,
            [Description("A comma delimited list of containers to be "
                         "checked or repaired if the Type bitmap is "
                         "set to User. Note that the container list "
                         "is not used if either the System bit is set "
                         "or both the System and User bits are set "
                         "in the Type bitmap. ")]
        string ContainerList);

    [Description("Stop a FileSystemCheck operation that is in progress. "
                 "The return codes are as follows. "
                 "  Ok - Success, requested the check to stop.  Note that "
                 "    a running filesystem check may not stop immediately "
                 "    and this operation withe the check marked for "
                 "    cancel. "
                 "  Not Found - FileSystem check is not running. "
                 " Not the Admin Master Server - This administrative "
                 "     operation can be invoked only on the master. "
                 " Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
                 " Cannot Connect to Server - The admin server could not "
                 "     connect to the local SAN FS server. "
                 " Too Many Connections - Exceeded the limit on number "
                 "     of parallel administrative operations allowed. "
                 "  Cancel Pending - A FileSystem check was found running "
                 "     but it is already marked for stop. "
                 "  Internal Error - "),
     ValueMap {"0", "21", "22", "44", "61", "62", "76", ".."},
     Values   {"OK", "Not Found", "Not the Admin Master Server",
               "Incompatible Operation", "Cannot Connect to Server",
               "Too Many Connections", "Cancel Pending", "Internal Error"}]
uint32 StopFileSystemCheck();
```

*Figure 27. MOF STC_MasterService class definition (Part 5 of 6)*

```
    [Description("Commit the cluster to start using the latest software "
                    "as all the servers in the cluster are upgraded to the "
                    "same latest version level. "
                    "The return codes are as follows. "
                    " OK - Successful. "
                    " Not the Admin Master Server - You must execute this "
                    "      command on the master server. "
                    " Incompatible Operation - Some other operation is active "
                    "      that is incompatible with commit upgrade. "
                    "      Try again later. "
                    " Cannot Connect to Server - Connection to the meta-data "
                    "      server using the administrative port failed. "
                    "      Make sure that the master server is not down and "
                    "      try again. "
                    " Too Many Connections - There are too many "
                    "      administrative commands active at the same time. "
                    "      Try again later. "
                    " Already In Progress - Another commit is already in "
                    "      progress. "
                    " Up To Date - The committed version is the same as the "
                    "      software version of all the servers. "
                    " All Not At Same Version - All the servers in the "
                    "      cluster are not at the same software version. "
                    " Internal Error - Any other error."),
    ValueMap{"0", "22", "44", "68", "69", "70", ".."},
    Values {"Ok", "Not the Admin Master Server", "Incompatible Operation",
            "Already In Progress", "Up To Date", "All Not At Same Version",
            "Internal Error"}]
    uint32 CommitUpgrade();
};
```

*Figure 27. MOF STC_MasterService class definition (Part 6 of 6)*

**Related topics:**

- "STC_MasterService" on page 110

## MOF STC_TankSAP class definition

```
/*
 * STC_TankSAP
 *
 * Issues:
 *
 *    . Should look at CIM_Network26.mof for detailed modelling.
 *    . The TypeOfAddress is taken from CIM__IPAddressRange from the
 *      CIM_Network26 mof.
 *    . The statistical info should go to CIM_SAPStatisticalInformation and
 *      CIM_SAPStatistics.
 *    . Make sure that the port numbers are 32-bit.
 *    . Use CIM_ProtocolEndPoint?
 */
       [Description("The StorageTank server access point."),
    provider("com.ibm.storage.storagetank.provider.std.TankSAP")]
class STC_TankSAP : CIM_ServiceAccessPoint
{
        [Description (
         "An enumeration that defines how to format the address and "
         "mask of the address range that defines this IPSubnet)."
         "\n\n"
         "Whenever possible, IPv4-compatible addresses should "
         "be used instead of IPv6 addresses (see RFC 2373, "
         "section 2.5.4). In order to have a consistent format "
         "for IPv4 addresses in a mixed IPv4/v6 environment, all "
         "IPv4 addresses and both IPv4-compatible IPv6 addresses "
         "and IPv4-mapped IPv6 addresses, per RFC 2373, section "
         "2.5.4, should be formatted in standard IPv4 format. "
         "However, this (the 2.2) version of the Network Common "
         "Model will not explicitly support mixed IPv4/IPv6 "
         "environments. This will be added in a future release."),
        ValueMap { "0", "1", "2" },
        Values { "Unknown", "IPv4", "IPv6" } ]
    uint16 TypeOfAddress = 1;
```

*Figure 28. MOF STC_TankSAP class definition (Part 1 of 2)*

```
    [Description("The IP address of the ethernet network interface "
                 "of a StorageTank server node. The GroupServices (GS) "
                 "and the StorageTank protocol (STP) are bound to this IP "
                 "at boot time. The HeartBeat (HB) protocol and "
                 "the Admin Service are also bound to this port for "
                 "service.")]
    string Ip;

        [Description("Cluster port used by internal group services "
                 "infrastructure communication.This port must be free on "
                 "both the interfaces when the server node is started. "
                 "Default value is 1737."),
         MinValue(1024), MaxValue(65535)]
    uint32 ClusterPort = 1737;

        [Description("Heartbeat port used by internal group services "
                 "infrastructure communication to receive heartbeats "
                 "on. This port must be free on both the interfaces when "
                 "the server node is started. Default value is 1738."),
         MinValue(1024), MaxValue(65535)]
    uint32 HeartbeatPort = 1738;

        [Description("Storage Tank Protocol (STP) port used for communication "
                 "with file system clients.This port must be free on both "
                 "the interfaces when the server node is started. "
                 "Default value is 1700."),
         MinValue(1024), MaxValue(65535)]
    uint32 STPPort = 1700;

        [Description("The port to receive administrative requests. This port "
                 "must be free on both the interfaces when the server node "
                 "is started."),
         MinValue(1024), MaxValue(65535)]
    uint32 AdminPort = 1800;
```

*Figure 28. MOF STC_TankSAP class definition (Part 2 of 2)*

**Related topics:**

- "STC_TankSAP" on page 148

## MOF STC_MasterSAP class definition

```
/*
 * STC_MasterSAP
 *
 */
        [Description("The StorageTank master server access point."),
     provider("com.ibm.storage.storagetank.provider.xnp.MasterSAP")]
class STC_MasterSAP : STC_TankSAP
{
        [Description("Indicates if the local server is the cluster master.")]
    boolean IsLocal;
};
```

*Figure 29. MOF STC_MasterSAP class definition*

**Related topics:**

- "STC_MasterSAP" on page 109

## MOF STC_AdminUser class definition

```
/*
 * STC_AdminUser
 * Issues:
 *     . Use CIM_Account and use the LDAP DN for the key.
 *     . Show - LDAP server config, keys used to look for a user, etc.
 *            - Show vaild window interval.
 *
 *     . Make the method parameter UserName optional.
 *     . Should the clear method be static?
 */
    [Description("This class lists instances of all users in the LDAP server "
                 "that are are admin users of StorageTank. "
                 "StorageTank administration operations are authenticated and "
                 "authorized by using a customer configured LDAP directory "
                 "server. This security service is done in CIMOM. Every "
                 "property and method of a class in the MOF has a Role "
                 "qualifier that determines the minimum role needed to get/set "
                 "a property or invoke a method. When an admin request is "
                 "received by the CIMOM, the CIMOM authenticates the user in "
                 "the LDAP server and extracts the Role of a successfully "
                 "authenticated user. For improving performance, an "
                 "authenticated user's role remains valid for a small interval "
                 "of time and the CIMOM does not consult LDAP again within "
                 "this window. In other words, if the same user makes an "
                 "admin request again when the valid window is open , the "
                 "cimom will authenticate the user without consulting the "
                 "LDAP server. To mitigate the problem of the role being  "
                 "stale, this class defines a method to clear all current "
                 "windows as well as to clear a window of a particular user. "
                 "The effective role of a user is determined by the direct "
                 "role of the user or by the "
                 "groups the user may be in. The strongest role is deemed the "
                 "effective role. This class also shows if a recent "
                 "authentication done for a user, if any, is still valid and "
                 "the interval of time the window will still be open. "
                 "If a user makes a request when the window is open, this "
                 "expiry time is not reset. If a valid window is open for a "
                 "user, it may not mean that the user is active currently."),
     provider("com.ibm.storage.storagetank.provider.std.AdminUser")]
class STC_AdminUser : CIM_LogicalElement
{
   [Propagated ("CIM_System.CreationClassName"), Key,
      MaxLen (256), Description ("The scoping System's CCN.") ]
   string SystemCreationClassName;
```

*Figure 30. MOF STC_AdminUser class definition (Part 1 of 2)*

```
   [Propagated ("CIM_System.Name"), Key,
      MaxLen (256),Description ("The scoping System's Name.") ]
   string SystemName;

   [Key, MaxLen (256), Description (
      "CreationClassName indicates the name of the class or the "
      "subclass used in the creation of an instance. When used "
      "with the other key properties of this class, this property "
      "allows all instances of this class and its subclasses to "
      "be uniquely identified.") ]
   string CreationClassName;

   [Key, Override("Name"), MaxLen (256), Description ("The User name.")]
   string Name;

    [Description("The effective role of the user, as determined by the "
                "authentication (CIMOM) service."),
    ValueMap{"0", "1", "2", "3", ".."},
    Values  {"Administrator", "Operator", "Backup", "Monitor", "Unknown"}]
   uint16 EffectiveRole;

    [Description("If the user's role was validated on the LDAP server recently "
                "the effective role for a user will remain valid for a short "
                "interval of time. If a user makes a request again in this "
                "time interval, the LDAP server will not be contacted for "
                "authentication to improve performance. "
                "This property indicates if there such valid window open.")]
   boolean IsAuthorizationCurrent;

    [Description("The remaining time interval after which if the user makes "
                 "a request, the LDAP server will be contacted again to "
                 "determine the effective role of a user. The value of this "
                 "property is valid only if authorization is current. "
                 "Otherwise, the value will be zero."),
     Units("Seconds")]
    uint32 AuthCurrentRemainingTime;

    /*
     * Methods
     */
    [Description("Clear specified validation. "),
    Values{"Ok", "Not Found", "Internal Error"},
    ValueMap{"0", "21", ".."}]
    uint32 ClearCurrentAuthorization();

    [Description("Clear all current validations. "),
    Static,
    Values{"Ok", "Not Found", "Internal Error"},
    ValueMap{"0", "21", ".."}]
    uint32 ClearAllCurrentAuthorizations();
};
```

*Figure 30. MOF STC_AdminUser class definition (Part 2 of 2)*

**Related topics:**

- "STC_AdminUser" on page 93

## MOF STC_ComputerSystem class definition

```
/*
 *  STC_ComputerSystem
 */
    [Description ("This class represents instances for every node in the "
                  "cluster."),
     provider("com.ibm.storage.storagetank.provider.std.ComputerSystem")]
class STC_ComputerSystem: CIM_ComputerSystem
{
        [Key, MaxLen(256), Override("CreationClassName"),
         Description("Set the value as this class.")]
   string CreationClassName = "STC_ComputerSystem";

        [Description("Total number of hours the node has been powered on"),
        Counter, Units("Hours")]
   uint64 TotalPowerOnHours;

        [Description("Total number of times the Node has been power cycled"),
        Counter]
   uint16 RestartCount;

        [Description("This property shows the power status of the server "
                     "node.")]
   boolean IsPowerOn;

        [Description("ASMTime is the current time on the Advanced Systems "
                     "Management (ASM) device's local clock.It is the time "
                     "reference that has to be used to schedule a power off "
                     "via the SetPowerState function. This time is independent"
                     "of the date and time on the server node.")]
   dateTime ASMTime;

       [Description("This property shows the state of the system in more "
                    "detail."),
        ValueMap {"0", "1", "2", "3", "4", "5", "6", "7"},
        Values {"Unknown/Power Off", "In POST", "Stopped in POST",
                "Booted Flash", "Booting OS", "In OS", "CPU's held in reset",
                "Before Post"}]
   uint32 CurrentState;

        [Description("The Universal Unique Identifier of the Node. This is "
                     "a 128-bit number represented in a hexadecimal string.")]
   string UUID;
```

*Figure 31. MOF STC_ComputerSystem (Part 1 of 2)*

```
    [Description("This method supports only a limited subset of power "
                 "setting capabilities that are fully described in the "
                 "parent class. The Time value supported is the regular "
                 "date-time value and not the interval value."
                 "The following are all the possible combiniations "
                 "of PowerState and Time"

                 "(a) Set state to Full Power at a specified Time."
                 "(b) Set state to Full Power immediately, with Time "
                 "     value set to zero. "
                 "(c) Set state to Power Cycle immediately, with the Time "
                 "     value set to zero. "
                 "(d) Set state to Power Off immediately, with the Time value "
                 "    set to zero. "
                 "(e) Set state to Soft Off immediately, with Time value set "
                 "    to zero. "),
        Override("SetPowerState"),
        ValueMap {"0", "1", "4", "63", "64", "71", ".."},
        Values {"Ok", "Not Supported", "Command Failed", "Already Enabled",
                "Already Disabled", "RSA Not Available", "Internal Error"}]

   uint32 SetPowerState(
      [IN, ValueMap {"1", "2", "3", "4", "5", "6", "7", "8"},
           Values {"Full Power", "Power Save - Low Power Mode",
                "Power Save - Standby", "Power Save - Other",
                "Power Cycle", "Power Off", "Hibernate", "Soft Off"}]
      uint16 PowerState,
      [IN] datetime Time);

     [Description("Invoke the one-button data collector scripts that"
                 "collects all the system information needed for"
                 "problem determination. "
                 "The collected system and server information is"
                 "placed in the file /usr/tank/pmf directory by"
                 "default. The actual directory depends on the"
                 "TANKDIR environment variable. Any stdout and"
                 "stderr output generated by the script is captured"
                 "in /tmp/obdcout file on the local disk of this"
                 "node."
                 " The return codes are as follows. "
                 " Ok - Successful. "
                 " Internal Error - Any other error."),
        ValueMap{"0", ".."},
        Values {"Ok", "Internal Error"}]
    uint32 OneButtonDataCollector();
};
```

*Figure 31. MOF STC_ComputerSystem (Part 2 of 2)*

**Related topics:**

- "STC_ComputerSystem" on page 96

## MOF STC_TankService class definition

```
/*
 * STC_TankService
 *
 * The Top Level Object (TLO) for this is the STC_Computer class.
 *
 * We use the inherited StopService() and StartService() methods to mean
 * Stop server and Start server respectively.
 *
 * The Kill server should be provided by the CIMOM itself.
 *
 */
    [Description ("Server services to start and stop a server are provided "
                  "by this class."),
     provider("com.ibm.storage.storagetank.provider.xnp.TankService")]
class STC_TankService : CIM_Service
{
        [Description("Indicates the StorageTank specific state of the server. "
                  "The server can be down (value=0), "
                  "Online (value=1), "
                  "put in a limited quiescent state where "
                  "only server (metadata) I/O operations are suspended "
                  "(value=2), "
                  "put in a full quiescent state where all "
                  "background IO, client, and server operations are "
                  "suspended (value=3), "
                  "put in a administrative state no longer servicing "
                  "clients (value=4), "
                  "initializing the first time (value=5), "
                  "Failed initialization as it encountered an error during "
                  " startup or group formation (value=6), "
                  "not commissioned into a cluster (value=7), "
                  "or joining a cluster (value=8)."),
         ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8" , "9"},
         Values {"Down", "Online", "PartlyQuiescent", "FullyQuiescent",
                 "AdminQuiescent", "Initializing", "FailedInit",
                 "UnCommissioned", "Joining", "Unknown"}]
    uint32 CurrentState;

        [Description("The server current state will be transitioning to "
                     "this state if the current state is different from "
                     "from this pending state."),
         ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8" , "9"},
         Values {"Down", "Online", "PartlyQuiescent", "FullyQuiescent",
                 "AdminQuiescent", "Initializing", "FailedInit",
                 "UnCommissioned", "Joining", "Unknown"},
         ModelCorrespondence {"CurrentState"}]
    uint32 PendingState;

        [Description("Time when the server was last started")]
    datetime LastBootUpTime;

        [Description ("Server's notion of the local date and time of day.")]
    datetime LocalDateTime;
```

*Figure 32. MOF STC_TankService class definition (Part 1 of 4)*

```
    [Description ("Time (delta) since the server changed its current "
                  "state.")]
datetime LastCurrentStateChangeTime;

    [Description ("Time (delta) since the server has a state change "
                  "pending. ")]
datetime LastPendingStateChangeTime;

    [Description("Current software release version. This is the version "
                 "of the latest upgrade done. This will be different "
                 "from the previous (committed) version if there was "
                 "an upgrade done but the commit was not yet activated.")]
string CurrentVersion;

    [Description("Indicates if this is the cluster master.")]
boolean IsMaster;

    [Description("Number of containers served from this server.")]
uint32  NumberOfContainers;

/*
 * Methods
 */

    [Override("StartService"),
     Description("Start the StorageTank server on this node. This is the "
                 "inherited method from CIM_Service. See the description "
                 "for the CIM_Service:StartService() method. We override "
                 "the method here to provide the proper error returns. "
                 "The return values of 0 and 1 are described by the parent "
                 "and therefore, should mean the same here. Note that the "
                 "parent does not define the return values, and hence we "
                 "can define and expand them here. "
                 "The return codes are as follows. "
                 "  Ok - Successful. "
                 "  Not Supported - This is to comply with the parent "
                 "    definition in the MOF. Never returned. "
                 "  Command Failed - Obvious! "
                 "  In Use - The SAN.FS server is already running. "
                 "  Not Found - A server with the given name is not found. "
                 "  Server Timedout - The server was launched "
                 "     successfully but failed to come online after "
             "      a maximum wait period. "
                 "  Incompatible Operation - Some other incompatible "
                 "     operation is currently running. Check the "
                 "     currently running administrative processes "
                 "     and try again. "
```

*Figure 32. MOF STC_TankService class definition (Part 2 of 4)*

```
                  "  Cannot Connect to Server - The admin server could not "
                  "     connect to the local SAN FS server. "
                  "  Cannot Persist Watchdog State - This is only a "
                  "    warning and the server is started successfully. "
                  "    The watchdog state could not be persisted for some "
                  "    reason. "
                  "  Aborted - The server has aborted instead of coming up. "
                  "  Internal Error - The SAN FS server encountered an "
                  "     internal error. Please see the message log and call "
                  "     Technical Support for resolution."),
      ValueMap{ "0", "1", "4", "5", "21", "24", "44", "61", "66",
                "78", ".."},
      Values {"OK", "Not Supported", "Command Failed", "In Use", "Not Found",
              "Server Timedout", "Incompatible Operation",
              "Cannot Connect to Server", "Cannot Persist Watchdog State",
              "Aborted", "Internal Error"}]
    uint32 StartService();

        [Override("StopService"),
         Description("Stop the StorageTank server on this node. This is the "
                     "inherited method from CIM_Service. See the description "
                     "for the CIM_Service:StopService() method. We override "
                     "the method here to provide the proper error returns. "
                     "The return values of 0 and 1 are described by the parent "
                     "and therefore, should mean the same here. Note that the "
                     "parent does not define the return values, and hence we "
                     "can define and expand them here."),
         ValueMap{ "0", "1", "4", "21", "24", "44", "61", "62", "66", ".."},
         Values {"OK", "Not Supported", "Command Failed", "Not Found",
                 "Server Timedout", "Incompatible Operation",
                 "Cannot Connect to Server", "Too Many Connections",
                 "Cannot Persist Watchdog State", "Internal Error"}]
    uint32 StopService();

        [Description("In the event of irrecoverable loss of the cluster "
                     "master server, make this subordinate server "
                     "the cluster master. A master may be lost due to "
                     "hardware failures, software failures, or partitioned "
                     "networks. If an administrator determines that a master "
                     "server is irrecoverably lost and decides to make a "
                     "subordinate server the new master, the administrator "
                     "must ensure that the previous master is down to prevent "
                     "rogue server issues. For instance, turn the power off to "
                     "the node hosting the dead master server. Once a master "
                     "is lost, the subordinate servers will be in either "
                     "Initializing or Joining states and certainly not in any "
                     "of the operational states (Online, PartlyQuiescent, "
                     "FullyQuiescent, AdminQuiescent). Note that you cannot "
                     "change the master server if the cluster master state"
                     "is not really down i.e. there is an existing master . "
                     "Note also that unlike most of the other commands, this "
                     "command must be issued on a subordinate node and not the "
                     "master node . "
```

*Figure 32. MOF STC_TankService class definition (Part 3 of 4)*

```
                    "The return codes are as follows. "
                    "  OK  - Successful "
                    "  Incompatible Operation - Some other incompatible "
                    "     operation is currently running. Check the "
                    "     currently running administrative processes "
                    "     and try again. "
                    "  Not Subordinate - The server is the master. "
                    "  Invalid Subordinate State  - The server is a "
                    "                  subordinate, but it is not in the "
                    "                  right state to become the master. "
                    " Cannot Connect to Server - The admin server could not "
                    "     connect to the local SAN FS server. "
                    " Too Many Connections - Exceeded the limit on number "
                    "     of parallel administrative operations allowed. "
                    " Cannot Become Admin Master - Once the local SAN FS "
                    "   server is made the master, the local admin server "
                    "   attempted to set itself as the master admin server "
                    "   and failed. The resons can be that the previous "
                    "   admin master thinks it is still the master, unable to "
                    "   communicate with remote admin servers, etc. This is "
                    "   only a warning in that the SAN FS server is set as "
                    "   master. "
                    " Internal Error - Any other errors. Please see the  "
                    "   SAN FS and admin server message logs for details."),
        ValueMap{ "0", "44", "58", "59", "61", "62", "67", ".."},
        Values {"OK", "Incompatible Operation", "Not Subordinate",
                "Invalid Subordinate State", "Cannot Connect to Server",
                "Too Many Connections", "Cannot Become Admin Master",
                "Internal Error"}]
    uint32 BecomeMaster();
};
```

*Figure 32. MOF STC_TankService class definition (Part 4 of 4)*

**Related topics:**

- "STC_TankService" on page 149

## MOF STC_MasterMetrics class definition

```
/*
 * STC_MasterMetrics
 */

    [Description("This class represents the StorageTank Cluster Master server"
                 "metrics. There wil be only one instance of this class."),
     provider("com.ibm.storage.storagetank.provider.xnp.MasterMetrics")]
class STC_MasterMetrics :  CIM_ServiceStatisticalInformation
{
        [Description("Total number of transactions relating to meta-data"
                     "activity for system objects. System objects include"
                     "storage pools, containers, volumes, policy sets, and"
                     "nodes. The activity includes reads, create, delete,"
                     "modify, etc., of these objects"),
         Counter]
    uint64 TotalSystemMetaActivity;

        [Description("Total number of transactions relating to metadata "
                     "updates for system objects."),
         Counter]
    uint64 TotalSystemMetaUpdateActivity;

        [Description("Current number of total buffers for system meta-data"
                     "activity. Total buffers are split into clean buffers,"
                     "dirty buffers, and free buffers.")]
    uint32 TotalSystemBuffers;

        [Description("Current number of clean buffers for system meta-data"
                     "activity. Clean buffers contain data but the buffers"
                     "are available for re-use."),
         Gauge]
    uint32 CleanSystemBuffers;

        [Description("Current number of dirty buffers for system meta-data"
                     "activity. Dirty buffers contain data awaiting I/O to"
                     " disk."),
         Gauge]
    uint32 DirtySystemBuffers;

        [Description("Current number of free buffers for system meta-data"
                     "activity. Free buffers are those that are currently"
                     "not in use."),
         Gauge]
    uint32 FreeSystemBuffers;

};
```

*Figure 33. MOF STC_MasterMetrics class definition*

**Related topics:**

- "STC_MasterMetrics" on page 109

## MOF STC_TankMetrics class definition

```
/*
 * STC_TankMetrics
 */
    [Description("This class represents the StorageTank subordinate server"
                 "metrics. There will be one instance of this class for every"
                 "server running, including the cluster master server."),
     provider("com.ibm.storage.storagetank.provider.xnp.TankMetrics")]
class STC_TankMetrics :  CIM_ServiceStatisticalInformation
{

        [Description("Total number of transactions relating to file system "
                     "metadata activity including container attach points, "
                     "creating directories, extending files, etc."),
         Counter]
    uint64 TotalUserMetaActivity;


        [Description("Total number of transactions relating to file system "
                     "updates for system objects."),
         Counter]
    uint64 TotalUserMetaUpdateActivity;


        [Description("Current number of session locks held in the Lock"
                     "Manager for this node. Session lock holds a reference"
                     "on inode. A client must acquire a session lock to do"
                     "any operation with a filename, eg. stat, lstat, opendir"
                     "etc."),
         Gauge]
    uint64 SessionLocks;


        [Description("Current number of data locks held in the Lock Manager"
                     "for this node. Clients must hold datalocks in order to"
                   "cache data pages and attributes of files and to cache"
                     "read-only attributes and contents of directories and"
                     "links." ),
         Gauge]
    uint32 DataLocks;

        [Description("Current number of byte range locks held in the Lock"
                     "Manager for this node. These locks are used to implement"
                     "POSIX, SYSV and Berkeley lock system calls, they have no"
                     "direct effect on the contents or attributes of storage"
                     "tank objects or other file system operations."),
          Gauge]
    uint32 ByteRangeLocks;


        [Description("Current number of total buffers for user meta-data"
                     "activity. Total buffers are split into clean buffers,"
                     "dirty buffers, and free buffers.")]
    uint32 TotalBuffers;
```

*Figure 34. MOF STC_TankMetrics class definition (Part 1 of 2)*

```
        [Description("Current number of clean buffers for user meta-data"
                     "activity. Clean buffers contain data but the buffers"
                     "are available for re-use."),
         Gauge]
     uint32 CleanBuffers;

        [Description("Current number of dirty buffers for user meta-data"
                     "activity. Dirty buffers contain data awaiting I/O to"
                     " disk."),
         Gauge]
     uint32 DirtyBuffers;

        [Description("Current number of free buffers for user meta-data"
                     "activity. Free buffers are those that are currently"
                     "not in use."),
         Gauge]
     uint32 FreeBuffers;

};
```

*Figure 34. MOF STC_TankMetrics class definition (Part 2 of 2)*

**Related topics:**

- "STC_TankMetrics" on page 147

## MOF STC_AdminProcess class definition

```
/*
 * STC_AdminProcess
 */
    [Description ("This class represents the long running administrative "
                  "commands in the cluster."),
     provider("com.ibm.storage.storagetank.provider.xnp.AdminProcess")]
class STC_AdminProcess : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName = "STC_Cluster";

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Propagated ("CIM_Service.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping Service's Name.")]
    string ServiceCreationClassName = "STC_MasterService";

        [Propagated ("CIM_Service.Name"),
         Key, MaxLen (256),
         Description ("The scoping Service's Name.")]
    string ServiceName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName = "STC_AdminProcess";

        [Key, Description ("The ID of the process.")]
    uint64 Id;

        [Override ("InstallDate"),
         Description("The date time when the process was started."),
         Alias("StartDateTime")]
    datetime InstallDate;

        [Description("The command that initiated this process.")]
    string Command;
};
```

*Figure 35. MOF STC_AdminProcess class definition*

**Related topics:**

- "STC_AdminProcess" on page 92

## MOF STC_RegisteredFSClients class definition

```
/*
 * STC_RegisteredFSClients
 */
    [Description ("This class lists all the registered File System clients of "
                  "a MDS server. Every client-server registration pair is "
                  "unique. The same client that registered to two servers will "
                  "appear as two instances. "),
     provider("com.ibm.storage.storagetank.provider.xnp.RegisteredFSClients")]
class STC_RegisteredFSClients : CIM_LogicalElement
{
        [Propagated ("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping System's CreationClassName. ")]
    string SystemCreationClassName;

        [Propagated ("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.")]
    string SystemName;

        [Propagated ("CIM_Service.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping Service's CreationClassName. ")]
    string ServiceCreationClassName;

        [Propagated ("CIM_Service.Name"),
         Key, MaxLen (256),
         Description ("The scoping Service's Name.")]
    string ServiceName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class "
                      "or the subclass used in the creation of an instance. "
                      "When used with the other key properties of this class, "
                      "this property allows all instances of this class and "
                      "its subclasses to be uniquely identified.")]
    string CreationClassName;

        [Override ("Name"),
         MaxLen (256),
         Description ("The name of the client.")]
    string Name;

        [Key, Description ("The ID of the client.")]
    uint64 Id;

        [Description ("The IP network address of the client.")]
    string IPAddress;

        [Description ("The IP network port address of the client.")]
    uint32 IPPort;

        [Description ("The OS platform of the client.")]
    string Platform;
```

*Figure 36. MOF STC_RegisteredFSClients class definition (Part 1 of 2)*

```
        [Description ("The File System(STP) version of the client.")]
    string Version;

        [Description ("Total number of times the client has renewed lease. "
                      "This indicates a measure of how long the client "
                      "was active."),
         Counter]
    uint64 LeaseRenewals;

        [Description ("State of the lease for this client."),
         ValueMap { "0", "1" },
         Values   { "Expired Lease", "Valid Lease" }]
    uint16 State;

        [Description("Boolean indicating if this SAN FS client "
                     "has Root/Administrator privileges to the SAN FS "
                     "File System Name-Space.")
        ]
    boolean IsPrivileged;

        [Description("The timestamp at which the server has issued/extended "
                     "the lease.")
        ]
    datetime LastLeaseTimeStamp;

        [Description ("This is a count-down timer indicating how long the "
                      "current lease will last. The lease "
                      "time is 2 * lease renewal interval, a configurable "
                      "parameter, beginning from the LastLeaseTimeStamp."),
         Units("Seconds")]
    uint32 ResidualLeaseTime;

        [Description ("Total number of transactions started by this client."),
         Counter]
    uint64 Transactions;

        [Description ("Total number of transactions completed."),
         Counter]
    uint64 CompletedTransactions;

        [Description ("Current number of session locks this client is holding."),
         Gauge]
    uint32 SessionLocks;

        [Description ("Current number of data locks this client is holding."),
         Gauge]
    uint32 DataLocks;

        [Description ("Current number of byte range locks this client is "
                      "holding."),
         Gauge]
    uint32 ByteRangeLocks;
};
```

*Figure 36. MOF STC_RegisteredFSClients class definition (Part 2 of 2)*

**Related topics:**

- "STC_RegisteredFSClients" on page 135

## MOF STC_NodeFan class definition

```
/*
 * STC_NodeFan
 *
 * ASM Node Fan status. Currently, we get the speeds of the fans as a
 * percentage of the optimal speed of 100.
 */
    [Description ("The STC_NodeFans"),
provider("com.ibm.storage.storagetank.provider.std.NodeFan")]
class STC_NodeFan : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ( "The scoping System's CreationClassName.") ]
   string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
   string SystemName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
   string CreationClassName;

        [Key, MaxLen (64),
         Description ("An address or other identifying information to uniquely "
                      "name the LogicalDevice.") ]
   string DeviceID;

        [Description("Speed of the fan in percentage."),
         Units("Percent")]
   uint32 Speed;
};
```

*Figure 37. MOF STC_NodeFan class definition*

**Related topics:**

- "STC_NodeFan" on page 123

## MOF STC_MessageLog class definition

```
/*
 * STC_MessageLog
 *
 */
    [Abstract,
     Description("The STC_MessageLog class represents all Log files"
                 "that are present in the Storage Tank subsystem, "
                 "its characteristics and provides methods to retrieve "
                 "log records from this file. We support and enhance the "
                 "iterator methods of traversing the log file as defined "
                 "in CIM_MessageLog class in the following ways. "
                 "1. We support PositionToLastRecord() method in addition to "
                 "   PositionToFirstRecord() method. "
                 "2. We support GetNextRecords() and GetPreviousRecords() "
                 "   methods that sets the direction of traversal from a "
                 "   given iterator identifier. These methods can return more "
                 "   than one log record as follows. These methods have a set "
                 "   of array properties as out parameters. All the property "
                 "   values at a given index into this array constitute a log "
                 "   record.")]
class STC_MessageLog : CIM_MessageLog
{

        [Description("The Absolute path and name of the Log File. This is "
                     "consistent across all Nodes in the Cluster.")]
    string LogFileName;

        [Override("ClearLog"),
         Description(
              "Requests that the MessageLog be cleared of all entries. "
              "See the description of the property in the parent class "
              "for detailed description. The method in the parent is "
              "overridden to define proper return values. "
              "The return codes are as follows. "
              "  OK - Successful. "
              "  Not Supported  - The Parent class mandates this return "
              "       code. We do not support clearing the Event log. "
              "  Command Failed - The clear failed on the SAN FS server. "
              "  Partial Data - The log was cleared partially and is not "
              "     complete. "
              "  Cannot Connect to Server - The admin server could not "
              "     connect to the local SAN FS server. "
              "  Too Many Connections - Exceeded the limit on number "
              "     of parallel administrative operations allowed. "
              "  Internal Error - The SAN FS server encountered an "
              "     internal error. Please see the message log and call "
              "     Technical Support for resolution."),
         ValueMap{"0", "1", "4", "41", "61", "62", ".."},
         Values  {"OK", "Not Supported", "Command Failed", "Partial Data",
                  "Cannot Connect to Server", "Too Many Connections",
                  "Internal Error"}
        ]
    uint32 ClearLog();
```

*Figure 38. MOF STC_MessageLog class definition (Part 1 of 6)*

```
    [Override("PositionToFirstRecord"),
     Description (
        "Requests that an iteration of the MessageLog be established "
        "and that the iterator be set to the first entry in the Log. "
        "An identifier for the iterator is returned as an output "
        "parameter of the method."),
     ValueMap{"0", "1", "39", "40", "41", ".."},
     Values  {"OK", "Not Supported", "File Not Found",
              "Cannot Read File", "Partial Data",
              "Internal Error"},
    ExecuteRole("Monitor")]
uint32 PositionToFirstRecord ([IN(false), OUT] string IterationIdentifier);

    [Description (
        "Requests that an iteration of the MessageLog be established "
        "and that the iterator be set to the last entry in the Log. "
        "An identifier for the iterator is returned as an output "
        "parameter of the method."),
     ValueMap{"0", "1", "39", "40", "41", ".."},
     Values  {"OK", "Not Supported", "File Not Found",
              "Cannot Read File", "Partial Data",
              "Internal Error"},
    ExecuteRole("Monitor")]
uint32 PositionToLastRecord ([IN(false), OUT] string IterationIdentifier);


    [Description (
        "Requests that a next number of records be retrieved from the "
        "MessageLog, starting from the record indicated by the "
        "IterationIdentifier. The number of records to be retrieved "
        "is given by the NumberOfEntries parameter. The "
        "IterationIdentifier is advanced to the record after the last "
        "record returned. If the traversal reaches the last record in the "
        "file, an \"End of Iteration\" code is returned. Subsequent calls "
        "to this method may return new records if they have been written "
        "to the log. The NumberOfEntries parameter returns the actual "
        "number of log records that were retrieved."),
     ValueMap{"0", "1", "10", "37", "38", "39", "40", "41", ".."},
     Values  {"OK", "Not Supported", "Invalid Parameter",
              "End of Iteration", "Invalid IterationIdentifier",
              "File Not Found", "Cannot Read File", "Partial Data",
              "Internal Error"},
    ExecuteRole("Monitor")]
```

*Figure 38. MOF STC_MessageLog class definition (Part 2 of 6)*

```
    uint32 GetNextRecords(
           [IN, OUT]
       string IterationIdentifier,
           [IN, OUT, MaxValue(100)]
       uint32 NumberOfEntries,
           [IN(false), OUT]
       datetime MessageTimestamp [],
           [IN(false), OUT]
       string MessageID [],
           [IN(false), OUT,
            ValueMap{"1", "2", "3", "4"},
            Values  {"Normal", "Event", "Audit", "Trace"}
           ]
       uint8 MessageType [],
           [IN(false), OUT]
       string SourceNode [],
           [IN(false), OUT,
            ValueMap{"0", "1", "2", "3"},
            Values  {"Information", "Warning", "Error", "Severe"}
           ]
       uint8 Severity [],
           [IN(false), OUT]
       string MessageString []);


       [Description (
           "Requests that the previous number of records be retrieved from "
           "the MessageLog, ending with the record indicated by the "
           "IterationIdentifier. The number of records to be returned is "
           "given in the NumberOfEntries parameter. The IterationIdentifier "
           "is positioned to the record before the first record returned. "
           "If the traversal reaches the first record in the file, an \"End "
           "of Iteration\" code is returned. Subsequent calls will have no "
           "effect. The NumberOfEntries parameter returns the actual "
           "number of log records that were retrieved."),
        ValueMap{"0", "1", "10", "37", "38", "39", "40", "41", ".."},
        Values  {"OK", "Not Supported", "Invalid Parameter",
                 "End of Iteration", "Invalid IterationIdentifier",
                 "File Not Found", "Cannot Read File", "Partial Data",
                 "Internal Error"},
       ExecuteRole("Monitor")]
```

*Figure 38. MOF STC_MessageLog class definition (Part 3 of 6)*

```
uint32 GetPreviousRecords(
        [IN, OUT]
    string IterationIdentifier,
        [IN, OUT, MaxValue(100)]
    uint32 NumberOfEntries,
        [IN(false), OUT]
    datetime MessageTimestamp [],
        [IN(false), OUT]
    string MessageID [],
        [IN(false), OUT,
         ValueMap{"1", "2", "3", "4"},
         Values  {"Normal", "Event", "Audit", "Trace"}
        ]
    uint8 MessageType [],
        [IN(false), OUT]
    string SourceNode [],
        [IN(false), OUT,
         ValueMap{"0", "1", "2", "3"},
         Values  {"Information", "Warning", "Error", "Severe"}
        ]
    uint8 Severity [],
        [IN(false), OUT]
    string MessageString []);


    [Override("GetRecord"),
     Description(
         "Requests that the record indicated by the IterationIdentifier "
         "be retrieved from the MessageLog. After retrieval, the Iteration "
         "Identifier may be advanced to the next record by setting the "
         "PositionToNext input parameter to TRUE. The method in the parent "
         "is overridden to explicitly qualify the OUT parameters as "
         "IN (false).")
    ]
uint32 GetRecord (
    [IN, OUT] string IterationIdentifier,
    [IN, Description ("Boolean indicating that the Iteration"
         "Identifier should be advanced to the next record, after "
         "retrieving the current Log entry.") ]
     boolean PositionToNext,
    [IN (false), OUT] uint64 RecordNumber,
    [IN (false), OUT] uint8 RecordData[]);
```

*Figure 38. MOF STC_MessageLog class definition (Part 4 of 6)*

```
    [Override("DeleteRecord"),
     Description (
          "Requests that the record indicated by the IterationIdentifier "
          "be deleted from the MessageLog. After deletion, the Iteration"
          "Identifier may be advanced to the next record by setting the "
          "PositionToNext input parameter to TRUE. If set to FALSE, then "
          "the IterationIdentifier will be positioned at the previous "
          "record. The method in the parent is overridden to explicitly "
          "qualify the OUT parameters as IN (false).")
    ]
uint32 DeleteRecord (
    [IN, OUT] string IterationIdentifier,
    [IN, Description ("Boolean that when set to TRUE requests the "
          "IterationIdentifier to be advanced to the next record, "
          "after the current entry is deleted. If set to FALSE, "
          "IterationIdentifier is set to the previous record.") ]
     boolean PositionToNext,
    [IN (false), OUT] uint64 RecordNumber,
    [IN (false), OUT] uint8 RecordData[]);


    [Override("WriteRecord"),
     Description (
          "Requests that a record be inserted at the Log position "
          "indicated by the IterationIdentifier. The entry's data is "
          "provided in the RecordData input parameter. After insertion, "
          "the IterationIdentifier may be advanced to the next record "
          "by setting the PositionToNext input parameter to TRUE. "
          "The output parameter, RecordNumber, returns the current "
          "record number addressed via the IterationIdentifier. The method "
          "in the parent is overridden to explicitly qualify the OUT "
          "parameters as IN (false).")
    ]
uint32 WriteRecord (
    [IN, OUT] string IterationIdentifier,
    [IN, Description ("Boolean indicating that the Iteration"
          "Identifier should be advanced to the next record, after "
          "writing the Log entry.") ] boolean PositionToNext,
    [IN] uint8 RecordData[],
    [IN (false), OUT] uint64 RecordNumber);


    [Override("FlagRecordForOverwrite"),
     Description (
          "Requests that the record indicated by the IterationIdentifier "
          "be flagged as overwriteable. After updating the entry, the "
          "Iteration Identifier may be advanced to the next record by "
          "setting the PositionToNext input parameter to TRUE. One output "
          "parameter is defined for the method RecordNumber. It returns the "
          "current record number addressed via the Iteration Identifier. "
          "The method in the parent is overridden to explicitly qualify the "
          "OUT parameters as IN (false).")
    ]
```

*Figure 38. MOF STC_MessageLog class definition (Part 5 of 6)*

```
   uint32 FlagRecordForOverwrite (
       [IN, OUT] string IterationIdentifier,
       [IN, Description ("Boolean indicating that the Iteration"
            "Identifier should be advanced to the next record, after "
            "updating the current Log entry.") ]
        boolean PositionToNext,
       [IN (false), OUT] uint64 RecordNumber);

};
```

*Figure 38. MOF STC_MessageLog class definition (Part 6 of 6)*

**Related topics:**

- "STC_MessageLog" on page 117

## MOF STC_AdminMessageLog class definition

```
/*
 * STC_AdminMessageLog
 *
 */
 [Description("The STC_AdminMessageLog class represents the Admin Server "
              "Message Log file."),
 provider("com.ibm.storage.storagetank.provider.std.AdminMessageLog")]
class STC_AdminMessageLog : STC_MessageLog
{

};
```

*Figure 39. MOF STC_AdminMessageLog class definition*

**Related topics:**

- "STC_AdminMessageLog" on page 92

## MOF STC_AdminSecurityLog class definition

```
/*
 * STC_AdminSecurityLog
 *
 */
 [Description("The STC_AdminSecurityLog class represents the Admin Server "
              "Security Log file."),
 provider("com.ibm.storage.storagetank.provider.std.AdminSecurityLog")]
class STC_AdminSecurityLog : STC_MessageLog
{

};
```

*Figure 40. MOF STC_AdminSecurityLog class definition*

**Related topics:**

- "STC_AdminSecurityLog" on page 92

## MOF STC_MDSMessageLog class definition

```
*
 * STC_MDSMessageLog
 *
 */
 [Description("The STC_MDSMessageLog class represents the Meta-data Server "
              "Message Log file."),
 provider("com.ibm.storage.storagetank.provider.std.MDSMessageLog")]
class STC_MDSMessageLog : STC_MessageLog
{
       [Description("The Absolute path and name of the Backup Log File. "
                    "This is consistent across all Nodes in the Cluster.")]
    string BackupLogFileName;

};
```

*Figure 41. MOF STC_MDSMessageLog class definition*

**Related topics:**

- "STC_MDSMessageLog" on page 117

## MOF STC_MDSAuditLog class definition

```
//*
 * STC_MDSAuditLog
 *
 */
 [Description("The STC_MDSAuditLog class represents the Meta-data Server "
              "Audit Log file."),
 provider("com.ibm.storage.storagetank.provider.std.MDSAuditLog")]
class STC_MDSAuditLog : STC_MessageLog
{
        [Description("The Absolute path and name of the Backup Log File. "
                     "This is consistent across all Nodes in the Cluster.")]
    string BackupLogFileName;
};
```

*Figure 42. MOF STC_MDSAuditLog class definition*

**Related topics:**

- "STC_MDSAuditLog" on page 116

## MOF STC_MDSEventLog class definition

```
/*
 * STC_MDSEventLog
 *
 */
 [Description("The STC_MDSEventLog class represents the Meta-data Server "
              "Event Log file. Though this is represented as a seperate "
              "File, the Event Log is primarly a filter that groups all Event "
              "Records from the STC_MDSMessageLog"),
 provider("com.ibm.storage.storagetank.provider.std.MDSEventLog")]
class STC_MDSEventLog : STC_MessageLog
{
        [Description("The Absolute path and name of the Backup Log File. "
                     "This is consistent across all Nodes in the Cluster.")]
    string BackupLogFileName;
};
```

*Figure 43. STC_MDSEventLog class definition*

**Related topics:**

- "STC_MDSEventLog" on page 117

## MOF STC_SystemMDRAid class definition

```
/*
 * STC_SystemMDRAid
 *
 */
    [Description("This class is an aid to System Meta-data disaster recovery "
                 "preparation. "
                 "It provides a mechanism to extract SAN FS's System "
                 "Meta-Data information into a dump file on the local disk "
                 "of the system (not on the SAN).  Multiple dump files "
                 "can be extracted to save the state of the system meta-data "
                 "at various points in time. From the dump file, "
                 "administrative commands to re-create system meta-data can "
                 "be generated using the GenerateCommandFiles method. The "
                 "following set of command files are generated. "
                 " TankSysCLI.auto - This file contains commands to recreate   "
                 "                   storage pools, containers, and policies. "
                 "                   In case of disaster, this file can be run "
                 "                   without manual intervention. "
                 " TankSysCLI.volume - This file contains commands to recreate "
                 "                   volumes. This file cannot be run without "
                 "                   manual verification and editing. "
                 " TankSysCLI.attachpoint - This file contains commands to "
                 "                   recreate container attachpoints. This file "
                 "                   cannot be run without manual verification, "
                 "                   editing, and intervention. "
                 "These command files are needed only in case of recovery, "
                 "and the generation can be postponed till when needed. Also "
                 "the SAN FS server need not be up and running to generate "
                 "these files. Once the system meta-data is recreated, the "
                 "dump can be taken again and compared with the original "
                 "dump to verify the recovery."),
     provider("com.ibm.storage.storagetank.provider.std.SystemMDRAid")]
class STC_SystemMDRAid : CIM_ManagedSystemElement
{
        [Propagated ("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping System's CreationClassName. ")]
    string SystemCreationClassName;
```

*Figure 44. MOF STC_SystemMDRAid class definition (Part 1 of 3)*

```
    [Propagated ("CIM_System.Name"),
     Key, MaxLen (256),
     Description ("The scoping System's Name.")]
string SystemName;

    [Key, MaxLen (256),
     Description ("CreationClassName indicates the name of the class "
                  "or the subclass used in the creation of an instance. "
                  "When used with the other key properties of this class, "
                  "this property allows all instances of this class and "
                  "its subclasses to be uniquely identified.")]
string CreationClassName;

    [Override ("Name"),
     Key, MaxLen (256),
     Description ("The Name of the extracted system meta-data dump as "
                  "follows. "
                  " <name>.dump")]

string Name;

    [Description("The system meta-data server local disk directory name "
                 "where the dump files and the generated command files "
                 "are stored."),
     MaxLen(256)]
string LocalDirectoryName;

    [Description("The script used to generate command files from the "
                 "meta-data dump file."),
     MaxLen(256)]
string CLIGeneratorName;

    [Override ("InstallDate"),
     Description("The date when the dump was created."),
     Alias("CreationDate")]
datetime InstallDate;

    [Description("The size of the meta-data dump file "),
     Units("KiloBytes")]
uint64 Size;

    [Description("Create a new meta-data recovery dump file. "
                 "Constructor. If IsForce is true, overwrite any existing "
                 "dump file."),
     Static,
     ValueMap{ "0", "7", "9", "18", "22", "30", "44", "61", "62", ".."},
     Values {"OK", "Insufficient Space", "Invalid Name", "Name Exists",
             "Not the Admin Master Server",  "Transaction Failed",
             "Incompatible Operation", "Cannot Connect to Server",
             "Too Many Connections", "Internal Error"}]
uint32 Create([IN, MaxLen(250)] string Name, [IN] boolean IsForce);
```

*Figure 44. MOF STC_SystemMDRAid class definition (Part 2 of 3)*

```
        [Description("Delete an existing meta-data recovery file."),
         ValueMap{ "0", "22", ".."},
         Values {"OK", "Not the Admin Master Server", "Internal Error"}]
    uint32 Delete();

        [Description("Generate commands for recreating meta-data from recovery "
                     "dump file.  Any existing command files will be "
                     "over-written "),
         ValueMap{ "0", "7", "22", ".."},
         Values {"OK", "Insufficient Space", "Not the Admin Master Server",
                 "Internal Error"}]
    uint32 GenerateCommandFiles();
};
```

*Figure 44. MOF STC_SystemMDRAid class definition (Part 3 of 3)*

**Related topics:**


- "STC_SystemMDRAid" on page 142

## MOF STC_NodeTemperature class definition

```
/*
 * STC_NodeTemperature
 */
        [Description ("This class represents the temperature state of "
                      "different  hardware components of a node, as reported "
                      "by the Advanced System Management (ASM) processor. "
                      "There is an instance of this class for each temperature "
                      "sensor available on every node of a cluster."),
         provider("com.ibm.storage.storagetank.provider.std.NodeTemperature")]
class STC_NodeTemperature : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName;

        [Key, MaxLen (64),
         Description ("An address or other identifying information to uniquely "
                      "name the LogicalDevice.") ]
    string DeviceID;

        [Description("The current temperature value of the device"),
         Units("Degrees C")]
    real32 Value;

        [Description("Whether thresholds are available for this device.")]
    boolean HasThresholds;
```

*Figure 45. MOF STC_NodeTemperature (Part 1 of 2)*

```
        [Description("Temperature Warning Reset threshold value. If the "
                     "temperature was above the Warning threshold and then "
                     "dropped below this value, any active temperature events "
                     "are cleared. A value of zero means that this threshold "
                     "is disabled.")]
    real32 WarningReset;

        [Description("Temperature warning threshold value. If the temperature "
                     "reaches this value, a warning event is generated. A "
                     "value of zero means that this threshold is disabled.")]
    real32 Warning;

        [Description("Soft Shutdown Temperature threshold value. If the "
                     "temperature reaches this value, a critical event is "
                     "generated and the server is powered off after the O/S "
                     "is shut down. A value of zero means that this threshold "
                     "is disabled.")]
    real32 SoftShutdown;

        [Description("Hard Shutdown Temperature threshold value. If the "
                     "temperature reaches this value, a critical event is "
                     "generated and the server is powered off immediately. "
                     "A value of zero means that this threshold is disabled.")]
    real32 HardShutdown;
};
```

*Figure 45. MOF STC_NodeTemperature (Part 2 of 2)*

**Related topics:**

- "STC_NodeTemperature" on page 123

## MOF STC_NodeVoltage class definition

```
/*
 * STC_NodeVoltage
 */
        [Description ("This class represents the voltage state of "
                      "different  voltage sources of a node, as reported "
                      "by the Advanced System Management (ASM) processor. "
                      "There is an instance of this class for each voltage "
                      "source available on every node of a cluster."),
provider("com.ibm.storage.storagetank.provider.std.NodeVoltage")]
class STC_NodeVoltage : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName;

        [Key, MaxLen (64),
         Description ("An address or other identifying information to uniquely "
                      "name the LogicalDevice.") ]
    string DeviceID;
```

*Figure 46. MOF STC_NodeVoltage class definition (Part 1 of 3)*

```
    [Description("The current voltage of this line")]
real32 CurrentVoltage;

    [Description("The default Voltage of this line")]
real32 DefaultVoltage;

    [Description("Whether thresholds are available for this device")]
boolean HasThresholds;

    [Description("Warning Reset Low value for this voltage line. If the "
                 "voltage reading was outside the WarningLow and "
                 "WarningHigh threshold range, and then changed to a "
                 "value within this WarningResetLow and WarningResetHigh "
                 "range, any active voltage events are cleared. "
                 "A value of -99.99 means that this threshold is disabled.")
    ]
real32 WarningResetLow;

    [Description("Warning Reset High value for this voltage line. If the "
                 "voltage reading was outside the WarningLow and "
                 "WarningHigh threshold range, and then changed to a "
                 "value within this WarningResetLow and WarningResetHigh "
                 "range, any active voltage events are cleared. "
                 "A value of -99.99 means that this threshold is disabled.")
    ]
real32 WarningResetHigh;

    [Description("Warning low value for this voltage line. If the voltage "
                 "drops below this value, a warning event is generated. "
                 "A value of -99.99 means that this threshold is disabled.")
    ]
real32 WarningLow;

    [Description("Warning high value for this voltage line. If the voltage "
                 "rises above this value, a warning event is generated. "
                 "A value of -99.99 means that this threshold is disabled.")
    ]
real32 WarningHigh;

    [Description("Soft Shutdown low value for this voltage line. If the "
                 "voltage drops below this value, a critical event is "
                 "generated and the server is powered off after the O/S "
                 "is shut down. A value of -99.99 means that this "
                 "threshold is disabled.")]
real32 SoftShutdownLow;
```

*Figure 46. MOF STC_NodeVoltage class definition (Part 2 of 3)*

```
        [Description("Soft Shutdown high value for this voltage line. If the "
                     "voltage rises above this value, a critical event is "
                     "generated and the server is powered off after the O/S "
                     "is shut down. A value of -99.99 means that this "
                     "threshold is disabled.")]
    real32 SoftShutdownHigh;

        [Description("Hard Shutdown value for this voltage line. If the "
                     "voltage drops below this value, a critical event is "
                     "generated and the server is powered off immediately. "
                     "A value of -99.99 means that this threshold is disabled.")
        ]
    real32 HardShutdownLow;

        [Description("Hard Shutdown value for this voltage line. If the "
                     "voltage rises above this value, a critical event is "
                     "generated and the server is powered off immediately. "
                     "A value of -99.99 means that this threshold is disabled.")
        ]
    real32 HardShutdownHigh;
};
```

*Figure 46. MOF STC_NodeVoltage class definition (Part 3 of 3)*

**Related topics:**

- "STC_NodeVoltage" on page 125

## MOF STC_NodeWatchdog class definition

```
/*
 * STC_NodeWatchdog
 */
    [Description ("This class represents an Advanced Systems Management "
                  "(ASM) service processor watchdog settings. There is an "
                  "instance of this class for each node in the cluster."),
     provider("com.ibm.storage.storagetank.provider.std.NodeWatchdog")]
class STC_NodeWatchdog : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName;

        [Key, MaxLen (256),
         Override ("Name"),
         Description ("An address or other identifying information to uniquely "
                      "name the WatchDog.") ]
    string Name;

        [Description("The Power-On Self Test (POST) watchdog is active "
                     "once when the power is coming up. "
                     "This property indicates the POST watchdog timeout "
                     "value. If the node fails to complete POST within the "
                     "time indicated by the TimeoutValue, the ASM subsystem "
                     "will generate a POST timeout alert and automatically "
                     "restart the system once. Once the system is restarted, "
                     "the POST watchdog is automatically disabled until the OS "
                     "is shutdown and the server is power cycled. A value of "
                     "zero indicates that this watchdog is disabled."),
         Units("Seconds")]
    uint32 POSTTimeout;
```

*Figure 47. MOF STC_NodeWatchdog class definition (Part 1 of 2)*

```
        [Description("The OS watchdog checks the state of the OS at "
                     "periodic intervals of time. This value indicates how "
                     "often the ASM will check that the OS is running "
                     "properly. A value of zero indicates that this watchdog "
                     "is disabled."),
         Units("Seconds")]
    uint32 OSMonitorInterval;

        [Description("This property indicates the  OS watchdog timeout "
                     "value. If the OS fails to respond to these checks "
                     "within this timeout value, the ASM substystem "
                      "will generate an OS Timeout alert and automatically "
                      "restart the system with the OS Watchdog automatically "
                      "disabled until the OS is shutdown and the server is "
                      "power cycled."),
         Units("Seconds")]
    uint32 OSTimeout;

        [Description("This vlaue indicates the timeout value of the "
                     "OS Boot Process or Loader watchdog. The timeout value "
                     "indicates the amount of time the ASM subsystem will wait "
                     "between the completion of POST and the end of loading the"
                     "OS. If the interval is exceeded, the ASM subsystem will "
                     "generate a Loader Timeout alert and automatically "
                     "restart the system once with the Loader Timeout "
                     "disabled until the OS is shutdown and the server is "
                     "power cycled. A value of zero indicates that this "
                     "watchdog is disabled." ),
         Units("Seconds")]
    uint32 LoaderTimeout;

        [Description("This property indicates the amount of time the "
                      "the ASM subsystem will wait for the OS to shutdown "
                      "before powering off the system."),
         Units("Seconds")]
    uint32 PowerOffDelay;
};
```

*Figure 47. MOF STC_NodeWatchdog class definition (Part 2 of 2)*

**Related topics:**

- "STC_NodeWatchdog" on page 126

## MOF STC_NodeVitalProductData class definition

```
/*
 * STC_NodeVitalProductData
 */
    [Description ("Vital Product Data of the components of the node."),
     Experimental,
     provider("com.ibm.storage.storagetank.provider.std.NodeVitalProductData")]
class STC_NodeVitalProductData: CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName;

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName;

        [Key, MaxLen (64),
         Description ("An address or other identifying information to uniquely "
                      "name the LogicalDevice.") ]
    string DeviceID;

        [Description("Logical Device's host Machine's Model Identifier")]
    string MachineModel;

        [Description("Logical Device's host Machine's Serial Number")]
    string SerialNumber;

        [Description("Logical Device's Firmware Revision")]
    string Revision;

        [Description("Logical Device's Firmware Revision Date")]
    dateTime RevisionDate;

        [Description("Logical Device's Firmware File Name")]
    string FirmwareFileName;

        [Description("Logical Device's Firmware Build ID")]
    string FirmwareBuildID;
};
```

*Figure 48. MOF STC_NodeVitalProductData class definition*

**Related topics:**

- "STC_NodeVitalProductData" on page 124

## MOF STC_Setting class definition

```
/*
 * STC_Setting
 */
    [Abstract,
     Description("Base class for StorageTank cluster and server configuration "
                 "parameters.")]
class STC_Setting :  CIM_Setting
{
        [Propagated ("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.")]
    string SystemCreationClassName;

        [Propagated ("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.")]
    string SystemName;

        [Propagated ("CIM_Service.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping Service's Name.")]
    string ServiceCreationClassName;

        [Propagated ("CIM_Service.Name"),
         Key, MaxLen (256),
         Description ("The scoping Service's Name.")]
    string ServiceName;
};
```

*Figure 49. MOF STC_Setting class definition*

**Related topics:**

- "STC_Setting" on page 136

## MOF STC_MasterDisruptiveSetting class definition

```
/*
 * STC_MasterDisruptiveSetting
 */
    [Description("Master (Cluster) configuration parameters that require "
                 "a cluster restart for any updates to be effective. This "
                 "class also contains parameters that are set only during "
                 "installation and therefore are read-only here."),
     provider("com.ibm.storage.storagetank.provider.xnp.MasterDisruptiveSetting")]
class STC_MasterDisruptiveSetting : STC_Setting
{
        [Description("A unique Cluster ID Number that is set only at install "
                     "time. If not specified at install, a default value "
                     "will be computed by taking the lower 16-bit value of "
                     "the system time during installation.")]
    uint32 ClusterID = 0;

        [Description("A unique Cluster name that is settable only at install "
                     "time.  This cluster name determines the name of the root "
                     "directory in the file system namespace. Besides this "
                     "usage, the cluster name is used for various other "
                     "purposes."),
        MaxLen(32)]
    string ClusterName;

        [Description("Amount of time to wait between attempted message "
                     "sends from server to client."),
         Units("MilliSeconds"), MinValue(200), MaxValue(1000)
        ]
    uint32 ClientTimeoutInterval = 500;

        [Description("Amount of time to wait between attempted message "
                     "sends from server to server."),
         Units("MilliSeconds"), MinValue(200), MaxValue(1000)
        ]
    uint32 ServerTimeoutInterval = 500;

        [Description("Interval between heartbeats written to disk."),
         Units("MilliSeconds"), MinValue(200), MaxValue(10000)
        ]
    uint32 DiskHeartbeatInterval = 500;

        [Description("Logical partition size."),
         Units("MegaBytes")]
    uint32 LogicalPartitionSize = 16;

        [Description("Interval between the heartbeats over the network."),
         Units("MilliSeconds"), MinValue(200), MaxValue(10000),
         Write]
    uint32 NWHeartbeatInterval = 500;

        [Description("Maximum number of heartbeats that can be missed before "
                     "a server is ejected from the cluster (network)."),
         MinValue(1), MaxValue(100),
         Write]
    uint32 NWMaxMissedHeartbeats = 3;
```

*Figure 50. MOF STC_MasterDisruptiveSetting (Part 1 of 2)*

```
        [Description("Maximum number of heartbeats that can be missed before "
                     "a server is ejected from the cluster (disk)."),
         MinValue(1), MaxValue(100),
         Write]
    uint32 DiskMaxMissedHeartbeats = 4;

        [Description("Amount of time a lock is leased to a client when the . "
                     "server grants a lock. The server applies a multiplier "
                     "before actually expiring the lease. "
                     " (see LockGracePeriodMultiplier)."),
         Units("Seconds"), MinValue(10), MaxValue(120),
         Write]
    uint32 LockLeasePeriod = 20;

        [Description("The server would wait the lock lease period times this "
                     "multiplier value before actually expiring a lease of a "
                     "lock to a client. The server must receive a lock "
                     "renewal request from the client in the meantime to keep "
                     "the lease active."),
         MinValue(0), MaxValue(4),
         Write]
    uint32 LockGracePeriodMultiplier = 2;

        [Description("Timeout for intra-cluster communications."),
         Units("MicroSeconds"), MinValue(500000), MaxValue(10000000),
         Write]
    uint32 ClusterTimeout = 1000000;

        [Description("Number of times a send from the server to the client "
                     "will be attempted before declaring the client dead."),
         MinValue(1), MaxValue(100),
         Write]
    uint32 RetriesToClient = 5;
};
```

*Figure 50. MOF STC_MasterDisruptiveSetting (Part 2 of 2)*

**Related topics:**

- "STC_MasterDisruptiveSetting" on page 106

## MOF STC_MasterDynamicSetting class definition

```
/*
 * STC_MasterDynamicSetting
 */
    [Description("Master (Cluster) configuration parameters that can be "
                 "dynamically updated, i.e., any updates do not require "
                 "a cluster restart. These parameters are saved and persisted "
                 "across cluster restarts."),
     provider("com.ibm.storage.storagetank.provider.xnp.MasterDynamicSetting")]
class STC_MasterDynamicSetting : STC_Setting
{
        [Description("Buffer cache size for master DB space in 4 KiloByte "
                "pages."),

         MinValue(2048), MaxValue(8192),
         Write]
    uint32 MasterBufferSize = 2048;

        [Description("Buffer cache size for subordinate DB space in 4 KiloByte "
                "pages."),
         MinValue(30000), MaxValue(250000),
         Write]
    uint32 SubordinateBufferSize = 200000;

        [Description("The interval that the space reclamation thread waits "
                     "between runs; 0 = Off."),
         Units("Minutes"), MinValue(0), MaxValue(1440),
         Write]
    uint32 SpaceReclaimDelay = 60;

        [Description("The comma-separated list of client names for whom root "
                     "privileges are granted in the file system namespace."),
         Write]
    string PrivilegedFSClients;
```

*Figure 51. MOF STC_MasterDynamicSetting class definition (Part 1 of 2)*

```
        [Description("This is a filter that decides if an SNMP trap is to be "
                     "generated if a significant event occurs in a server. "
                     "The usage is as follows. You can choose the severity "
                     "of the events that generate an SNMP trap by setting "
                     "the corresponding bit in this property.             "
                     "If this value is set to zero (no bits set), then SNMP "
                     "trap generation is disabled. "
                     "If all the bits are set to one, all event messages "
                     "generate SNMP traps."),
         Write,
         BitMap {"0", "1", "2", "3"},
         BitValues {"Information", "Warning", "Error", "Severe" } ]

    uint16 SNMPEvents;

        [Description("The comma-separated list of destination IP addresses "
                     "(Dotted decimal) of the SNMP Managers. If an SNMP trap "
                     "is generated, the trap is sent to this list of managers."),
         Write]
    string SNMPManagers;

        [Description("Number of threads for administrative operations. This "
                     "value can only be increased and not decreased. "),
         MinValue(1), MaxValue(10),
         Write]
    uint32 NumAdminThreads = 4;

        [Description("Number of threads for general operations. This value "
                     "can only be increased and not decreased."),
         MinValue(10), MaxValue(50),
         Write]
    uint32 NumWorkerThreads = 10;
};
```

*Figure 51. MOF STC_MasterDynamicSetting class definition (Part 2 of 2)*

**Related topics:**

- "STC_MasterDynamicSetting" on page 107

## MOF STC_TankDisruptiveSetting class definition

```
/*
 * STC_TankDisruptiveSetting
 */
    [Description("Server specific configuration parameters that needs a "
                 "server restart for an update to take into effect. For now, "
                 "these parameters are read-only and are specified on "
                 "the command line when the server is started."),
     provider("com.ibm.storage.storagetank.provider.xnp.TankDisruptiveSetting")]
class STC_TankDisruptiveSetting : STC_Setting
{
        [Description("A unique Server name that is settable only at install "
                     "time."),
        MaxLen(32)]
    string ServerName;

    [Description("Client-server and server-server communication protocol "
                 "type."),
     ValueMap {"0",   "1" },
     Values   {"UDP", "TCP"}]
    uint32 ProtocolType = 0;

    [Description("Client-server communication protocol type."),
     ValueMap {"0",   "1" },
     Values   {"UDP", "TCP"}]
    uint32 ClientNetworkProtocol = 1;
```

*Figure 52. MOF STC_TankDisruptiveSetting class definition (Part 1 of 2)*

```
    [Description("Server-server communication protocol type."),
     ValueMap {"0",   "1" },
     Values   {"UDP", "TCP"}]
    uint32 ServerNetworkProtocol = 0;

        [Description("Number of threads for garbage collection of deleted "
                     "files."),
         MinValue(1), MaxValue(4)]
    uint32 NumDeleteThreads = 1;

        [Description("The IP address of the ethernet interface for "
                     "the server.")]
    string PrimaryIP;

        [Description("Cluster port used by internal group services "
                     "infrastructure communication. This port must be free on "
                     "the interface when the server node is started. "
                     "Default value is 1737."),
         MinValue(1024), MaxValue(65535)]
    uint32 ClusterPort = 1737;

        [Description("Heartbeat port used by internal group services "
                     "infrastructure communication to receive heartbeats "
                     "on. This port must be free on the interface when "
                     "the server node is started. Default value is 1738."),
         MinValue(1024), MaxValue(65535)]
    uint32 HeartbeatPort = 1738;

        [Description("Storage Tank Protocol (STP) port used for communication "
                     "with file system clients.This port must be free on "
                     "the interface when the server node is started. "
                     "Default value is 1700."),
         MinValue(1024), MaxValue(65535)]
    uint32 STPPort = 1700;

        [Description("The port to receive administrative requests. This port "
                     "must be free on the interface when the server node is "
                     "started."),
         MinValue(1024), MaxValue(65535)]
    uint32 AdminPort = 1800;
};
```

*Figure 52. MOF STC_TankDisruptiveSetting class definition (Part 2 of 2)*

**Related topics:**

- "STC_TankDisruptiveSetting" on page 144

## MOF STC_TankTransientSetting class definition

```
/*
 * STC_TankTransientSetting
 */
    [Description("Server (Tank) configuration parameters that were "
                 "set on the command line but are not stored persistently."),
     provider("com.ibm.storage.storagetank.provider.xnp.TankTransientSetting")]
class STC_TankTransientSetting : STC_Setting
{
};
```

*Figure 53. MOF STC_TankTransientSetting class definition*

**Related topics:**

- "STC_TankTransientSetting" on page 152

## MOF STC_TankWatchdog class definition

```
/*
 * STC_TankWatchdog
 */
    [Description ("Tank Watchdog is an administrative service that keeps track "
                  "of the liveness of a storage tank server. If enabled, the "
                  "watchdog will restart the server in case it detects "
                  "positively that the server is not live and not present. "
                  "This class represents the Storage Tank server watchdog "
                  "operations. There is an instance of this class for each "
                  "server in the cluster."),
     provider("com.ibm.storage.storagetank.provider.std.TankWatchdog")]
class STC_TankWatchdog : CIM_LogicalElement
{
        [Propagated("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.") ]
    string SystemCreationClassName = "STC_Cluster";

        [Propagated("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.") ]
    string SystemName;

        [Propagated ("CIM_Service.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping Service's CreationClassName. ")]
    string ServiceCreationClassName = "STC_TankService";

        [Propagated ("CIM_Service.Name"),
         Key, MaxLen (256),
         Description ("The scoping Service's Name.")]
    string ServiceName;

        [Key, MaxLen (256),
         Description ("CreationClassName indicates the name of the class or "
                      "the subclass used in the creation of an instance. When "
                      "used with the other key properties of this class, this "
                      "property allows all instances of this class and its "
                      "subclasses to be uniquely identified.") ]
    string CreationClassName = "STC_TankWatchdog";

        [Key, MaxLen (256),
         Override ("Name"),
         Description ("An address or other identifying information to uniquely "
                      "name the WatchDog.") ]
    string Name;

        [Description("State of the watchdog. It can be the following. "
                     "  Off     -   The watchdog is manually turned off. "
                     "  On      -   The watchdog is manually turned on.  "
                     "  Standby -   The watchdog is in a passive standby "
                     "              mode as the server it is probing has been "
                     "              manually shut down. The watchdog will "
                     "              automatically turn itself on when the "
                     "              server it is watching is re-started. "
```

*Figure 54. MOF STC_TankWatchdog class definition (Part 1 of 4)*

```
                  "  Aborted -   The watchdog detected a non-live server "
                  "              too many times in a row. "
                  "              An administrator must manually turn the "
                  "              watchdog on again to continue. "
                  "  Unknown -   The watchdog is in an indeterminate state "
                  "              as the watchdog server could not be "
                  "              reached."
                  "The state of the watchdog is persisted."),
     ValueMap {"0", "1", "2", "3", "4"},
     Values   {"Off", "On", "Standby", "Aborted", "Unknown"}]
uint32 State = 0;

    [Description("This indicates the probed server status as found in the "
                 "current probe cycle. "
                 " NotProbed  - Not started the probe as the watchdog is "
                 "              either off, or in a standby state, or in a "
                 "              aborted state. "
                 " Probing    - Watchdog has started the probe. "
                 " ServerLive - Detected that the server is alive. There "
                 "              is no need to restart the server. "
                 " ServerAbsent - Positively detected that the server is "
                 "              absent. The watchdog will attempt to "
                 "              restart the server only in this case. "
                 " Unknown    - The liveness test failed and the "
                 "              absence test failed. "
                 "              The watchdog will not attempt to restart "
                 "              the server."),
     ValueMap {"0", "1", "2", "3", "4"},
     Values   {"NotProbed", "Probing", "ServerLive", "ServerAbsent",
               "Unknown"}]
uint32 ProbeState = 0;

    [Description("The watchdog will periodically start a probe at this "
                 "interval"),
     Units("Seconds"),
     MinValue(10), MaxValue(60)]
uint32 ProbeInterval = 10;

    [Description("Maximum time to wait for the server to respond to a "
                 "liveness test request before deciding that the server "
                 "is not live."),
     Units("Seconds"),
     MinValue(1), MaxValue(10)]
uint32 LiveTestTimeoutInterval = 2;
```

*Figure 54. MOF STC_TankWatchdog class definition (Part 2 of 4)*

```
    [Description("Number of times to try detecting liveness of the server "
                 "in case the server is declared not live. When this retry "
                 "limit is reached, the watchdog will be turned off. Just "
                 "to make it clear, this is the number of tries and not "
                 "actually the number of retries (tries - 1). For example, "
                 "if this value is 3, try sending the probe 3 times "
                 "(retry 2 times)."),
     MinValue(1), MaxValue(10)]
uint32 RetryLimit = 3;

/*
 * Watchdog Stats
 * These stats are initialized whenever watchdog becomes active from the .
 * off mode.
 */
    [Description("The date and time when the watchdog was started.")]
datetime StartTimeStamp;

    [Description("The date and time when the last probe was started.")]
datetime LastProbeTimeStamp;

    [Description("Total number of probes done so far."),
     Counter]
uint64 TotalProbes;

    [Description("The total number of times the liveness test has taken "
                 "longer than the test timeout interval and hence the "
                 "timeout error occurred."),
     Counter]
uint64 LiveTestTimeouts;

/* Retries statistics */
    [Description("Total number of retries done so far."),
     Counter]
uint64 TotalRetries;

    [Description("Number of retries in the current probe cycle. The "
                 "value of this variable will range from 0 to "
                 "one less than the limit as set by the RetryLimit "
                 "property."),
     Gauge, ModelCorrespondence {"RetryLimit"}]
uint32 CurrentRetries;

    [Description("The lowest number of retries reached so far.")]
uint32 RetriesLWM;

    [Description("The highest number of retries reached so far.")]
uint32 RetriesHWM;
```

*Figure 54. MOF STC_TankWatchdog class definition (Part 3 of 4)*

```
    /*
     * Liveness Test Stats
     */
        [Description("Time taken by the last liveness test.")]
    uint32 LastLiveTestTime;

        [Description("Low watermark for time taken by the liveness test."),
         Units("MilliSeconds")]
    uint32 LiveTestTimeLWM;

        [Description("High watermark for time taken by the liveness test."),
         Units("MilliSeconds")]
    uint32 LiveTestTimeHWM;

    /*
     * Absence test stats
     */
        [Description("Total number of time absence test was started."),
         Counter]
    uint64 TotalAbsenceTests;

        [Description("Time taken by the last absence test."),
         Units("MilliSeconds")]
    uint32 LastAbsenceTestTime;

        [Description("Low watermark Time taken by the last absence test."),
         Units("MilliSeconds")]
    uint32 AbsenceTestTimeLWM;

        [Description("High watermark for time taken by the last absence test."),
         Units("MilliSeconds")]
    uint32 AbsenceTestTimeHWM;

    /*
     * Methods
     */
        [Description("Enable the watchdog timer."),
         ValueMap { "0", "61", "63", "66", ".." },
         Values   { "OK", "Cannot Connect to Server", "Already Enabled",
                    "Cannot Persist Watchdog State", "Internal Error" }]
    uint32 Enable();

        [Description("Disable the watchdog timer."),
         ValueMap { "0", "61", "64", "66", ".." },
         Values   { "OK", "Cannot Connect to Server", "Already Disabled",
                    "Cannot Persist Watchdog State", "Internal Error" }]
    uint32 Disable();
};
```

*Figure 54. MOF STC_TankWatchdog class definition (Part 4 of 4)*

**Related topics:**

- "STC_TankWatchdog" on page 153

## MOF STC_TankEvents class definition

```
/*
 * STC_TankEvents
 */
    [Description("This class enumerates the possible events that can be "
                 "generated by a server. A master is capable of generating "
                 "any of these events whereas a subordinate server is capable "
                 "of generating only a  subset set of these events. "
                 "Note well that the instances enumerated by this class is "
                 "NOT the list of event that occurred in a server."),
     provider("com.ibm.storage.storagetank.provider.xnp.TankEvents")]
class STC_TankEvents : CIM_LogicalElement
{
        [Propagated ("CIM_System.CreationClassName"),
         Key, MaxLen (256),
         Description ("The scoping System's CreationClassName.")]
    string SystemCreationClassName;

        [Propagated ("CIM_System.Name"),
         Key, MaxLen (256),
         Description ("The scoping System's Name.")]
    string SystemName;

        [Propagated ("CIM_Service.CreationClassName"),
         Key, MaxLen (256),
         Description("The scoping Service's Name.")]
    string ServiceCreationClassName;

        [Propagated ("CIM_Service.Name"),
         Key, MaxLen (256),
         Description ("The scoping Service's Name.")]
    string ServiceName;

        [Description("The ID associated with the message that will be logged "
                     "in the server log when this event occurs."),
         Key]
    string MessageID;

        [Description("Severity of the event."),
         ValueMap{"0", "1", "2", "3"},
         Values  {"Information", "Warning", "Error", "Severe"}]
    uint8 Severity;
```

*Figure 55. MOF STC_TankEvents class definition (Part 1 of 2)*

```
       [Description("The message format the server will use to log a "
                    "message  when this event occurs in the server. If "
                    "there are any parameter format specifications "
                    "in this string, these will be replaced with actual "
                    "values when this message is logged.")]
    string Message;

        [Description("The name of the StorageTank SNMP Trap generated by this "
                    "event.")]
    string SNMPTrap;

        [Description("A boolean indicating if this event occurs, whether the "
                    "SNMPEvents configuration parameter filter will allow the "
                    "generation of the SNMP Trap.")]
    boolean IsSNMPTrapEnabled;

        [Description("Generate a test event. This method can be used to check "
                    "that an SNMP Manager can receive a trap. This event "
                    "causes a trap with severity information. Make sure "
                    "that the SNMPEvents configuration parameter is set to "
                    "allow event with severity Information. Make sure "
                    "that the SNMPManagers configuration parameter is also "
                    "set properly. Also make sure that the specified "
                    "SNMP manager(s) is configured properly and active to "
                    "receive traps."),
         Static,
         ValueMap{ "0", "22", ".."},
         Values {"OK", "Not the Admin Master Server", "Internal Error"}
         ]
    uint32 Test();
};
```

*Figure 55. MOF STC_TankEvents class definition (Part 2 of 2)*

**Related topics:**

- "STC_TankEvents" on page 145

## MOF STC_RemoteServiceAccessPoint class definition

```
/*
 * STC_RemoteServiceAccessPoint
 */

    [Description("This class provides information for how to access the admin "
                 "gui for the SAN.FS device."),
     provider("com.ibm.storage.storagetank.provider.std.RemoteServiceAccessPoint")]
class STC_RemoteServiceAccessPoint: CIM_RemoteServiceAccessPoint
{
};
```

*Figure 56. MOF STC_RemoteServiceAccessPoint class definition*

**Related topics:**

- "STC_RemoteServiceAccessPoint" on page 136

# Appendix A. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully.

**Features:**

These are the major accessibility features in SAN File System:

- You can use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen.

  **Note:** The SAN File System Information Center and its related publications are accessibility-enabled for the IBM Home Page Reader.
- You can operate all features using the keyboard instead of the mouse.

**Navigating by keyboard:**

You can use keys or key combinations to perform operations and initiate many menu actions that can also be done through mouse actions. You can navigate the SAN File System console and help system from the keyboard by using the following key combinations:

- To traverse to the next link, button or topic, press Tab inside a frame (page).
- To expand or collapse a tree node, press Right or Left arrows, respectively.
- To move to the next topic node, press Down arrow or Tab.
- To move to the previous topic node, press Up arrow or Shift+Tab.
- To scroll all the way up or down, press Home or End, respectively.
- To go back, press Alt+Left arrow
- To go forward, press Alt+Right arrow.
- To go to the next frame, press Ctrl+Tab. There are quite a number of frames in the help system.
- To move to previous frame, press Shift+Ctrl+Tab.
- To print the current page or active frame, press Ctrl+P.

# Appendix B. SNMP Trap MIB

The SAN File System SNMP MIB defines the structure and content of the following traps:

- tankGenericTrap
- tankClusterStateChangeTrap
- tankServerStateChangeTrap
- tankStoragePoolSpaceTrap
- tankContainerQuotaTrap
- tankLogRotateTrap

As defined in the MIB, the traps capture information about SAN File System resources at the time of an event.

```
IBM-STORAGETANK-MIB DEFINITIONS ::= BEGIN
IMPORTS
 OBJECT-TYPE, NOTIFICATION-TYPE, MODULE-IDENTITY,
 enterprises, Integer32
          FROM SNMPv2-SMI

        TEXTUAL-CONVENTION,
        DateAndTime
          FROM SNMPv2-TC

        MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
          FROM SNMPv2-CONF

        SnmpAdminString
          FROM SNMP-FRAMEWORK-MIB;         -- RFC2571

ibmStorageTankModule MODULE-IDENTITY
 LAST-UPDATED "200304070000Z" -- 07 April 2003 00:00 GMT
 ORGANIZATION "IBM SSG"
 CONTACT-INFO
  "
  postal: SSG Department
          IBM Beaverton Labs
   Beaverton
   Oregon
   USA"
```

*Figure 57. SNMP Trap MIB (Part 1 of 7)*

```
 DESCRIPTION
  "This file defines the private IBM MIB extensions for
                TotalStorage StorageTank product."
 REVISION "200304070000Z"
 DESCRIPTION
  "Initial version of IBM StorageTank MIB"
 ::= { ibmProd 195 }

ibm              OBJECT IDENTIFIER ::= { enterprises 2 }
ibmProd     OBJECT IDENTIFIER ::= { ibm 6 }

--An OID value of zero (0) is used below to be compatible to
--SNMPv1 TRAP-TYPE definitions.
ibmTankTraps OBJECT IDENTIFIER  ::= { ibmStorageTankModule 0 }
ibmStorageTankObjects OBJECT IDENTIFIER ::= { ibmStorageTankModule 2 }
ibmStorageTankConformance OBJECT IDENTIFIER ::= { ibmStorageTankModule 3 }

tankGenericTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageText,
                tankMessageId
 }
```

*Figure 57. SNMP Trap MIB (Part 2 of 7)*

```
 STATUS current
      DESCRIPTION
          "This is the StorageTank generic trap that uses SNMP Trap mechanism
           as an asynchronous transport for notifying managers about any
           critical log messages."
      ::= { ibmTankTraps 1 }

tankClusterStateChangeTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageId,
                tankClusterOldState,
                tankClusterCurrentState
 }
 STATUS current
      DESCRIPTION
          "This trap is generated when the StorageTank cluster state changes
           as defined by the ClusterState object. Both operational and
           non-operational state changes cause this trap. Also, the severity
           of this trap will depend on whether the cluster is becoming less
           accessible or more accessible. This trap is generated only by the
           server who is the cluster master."
      ::= { ibmTankTraps 2 }

tankServerStateChangeTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageId,
                tankServerOldState,
                tankServerCurrentState
 }
 STATUS current
      DESCRIPTION
          "This trap is generated when a StorageTank server changes state."
      ::= { ibmTankTraps 3 }

tankStoragePoolSpaceTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageId,
                tankStoragePoolName,
                tankAlertPercentage,
                tankAllocPercentage
 }
```

*Figure 57. SNMP Trap MIB (Part 3 of 7)*

```
      STATUS current
      DESCRIPTION
          "This trap is generated when a StorageTank StoragePool space
           allocated percentage exceeds the alert percentage set. The severity
           will be a warning if allocation percentage is still less than
           100%, indicating that the current allocation that caused this
           trap has succeeded. Otherwise, the severity will be set to error."
      ::= { ibmTankTraps 4 }

tankContainerQuotaTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageId,
                tankContainerName,
                tankAlertPercentage,
                tankAllocPercentage
 }
      STATUS current
      DESCRIPTION
          "This trap is generated when a container storage space
           allocated percentage exceeds the alert percentage set. The severity
           will be a warning if allocation percentage is still less than
           100% or soft quota is in effect, indicating that the current
           allocation that caused this trap has succeeded. Otherwise, the
           severity will be set to error."
      ::= { ibmTankTraps 5 }

tankLogRotateTrap NOTIFICATION-TYPE
 OBJECTS {
  tankClusterName,
  tankServerName,
  tankTimeStamp,
                tankSeverity,
                tankMessageId,
                tankLogType,
                tankLogName
 }
      STATUS current
      DESCRIPTION
          "This trap is generated when the current log file of a StorageTank
           log is full. If a log file becomes full, the current log
           file becomes the backup log and the current backup log becomes
           the current log. This trap is to remind that the backup log will
           need to be backed up.  The severity will be set to informational."
      ::= { ibmTankTraps 6 }

tankClusterName OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE (1..32))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The originating cluster name."
 ::= { ibmStorageTankObjects 1 }
```

*Figure 57. SNMP Trap MIB (Part 4 of 7)*

```
tankServerName OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE (1..32))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The originating node name."
 ::= { ibmStorageTankObjects 2 }

tankTimeStamp OBJECT-TYPE
 SYNTAX DateAndTime
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The date and time when this event occurred."
 ::= { ibmStorageTankObjects 3 }

tankSeverity OBJECT-TYPE
 SYNTAX INTEGER { information(1), warning(2), error(3), severe(4) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The severity of this event."
 ::= { ibmStorageTankObjects 4 }

tankMessageId OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE(10))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The message ID for this event as looged in the log file.."
 ::= { ibmStorageTankObjects 5 }

tankMessageText OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE(1..1024))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The description of the event that caused this trap."
 ::= { ibmStorageTankObjects 6 }


tankStoragePoolName OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE(1..256))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The name of the StoragePool."
 ::= { ibmStorageTankObjects 7 }

tankAlertPercentage OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The alert percentage value from 0 to 100."
 ::= { ibmStorageTankObjects 8 }
```

*Figure 57. SNMP Trap MIB (Part 5 of 7)*

```
tankAllocPercentage OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The space allocated so far from 0 to 100."
 ::= { ibmStorageTankObjects 9 }

tankClusterOldState OBJECT-TYPE
 SYNTAX INTEGER { down(0), online(1), partlyQuiescent(2),
                         fullyQuiescent(3), administrative(4), forming(5),
                         notMaster(6) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The current state of the cluster."
 ::= { ibmStorageTankObjects 10 }

tankClusterCurrentState OBJECT-TYPE
 SYNTAX INTEGER { down(0), online(1), partlyQuiescent(2),
                         fullyQuiescent(3), administrative(4), forming(5),
                         notMaster(6) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The current state of the cluster."
 ::= { ibmStorageTankObjects 11 }

tankServerOldState OBJECT-TYPE
 SYNTAX INTEGER { down(0), online(1), partlyQuiescent(2),
                         fullyQuiescent(3), administrative(4),
                         initializing(5), failedInit(6), unCommissioned(7),
                         joining(8) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The current state of the cluster."
 ::= { ibmStorageTankObjects 12 }

tankServerCurrentState OBJECT-TYPE
 SYNTAX INTEGER { down(0), online(1), partlyQuiescent(2),
                         fullyQuiescent(3), administrative(4),
                         initializing(5), failedInit(6), unCommissioned(7),
                         joining(8) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The current state of the cluster."
 ::= { ibmStorageTankObjects 13 }

tankContainerName OBJECT-TYPE
 SYNTAX SnmpAdminString (SIZE(1..256))
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The name of the Container."
 ::= { ibmStorageTankObjects 14 }
```

*Figure 57. SNMP Trap MIB (Part 6 of 7)*

```
tankLogType OBJECT-TYPE
 SYNTAX INTEGER { normal(1), audit(2) }
 MAX-ACCESS accessible-for-notify
 STATUS current
 DESCRIPTION
  "The type of log."
 ::= { ibmStorageTankObjects 15 }

tankLogName OBJECT-TYPE
        SYNTAX SnmpAdminString (SIZE(1..256))
        MAX-ACCESS accessible-for-notify
        STATUS current
        DESCRIPTION
                "The name of the Log File."
        ::= { ibmStorageTankObjects 16 }

tankCompliances OBJECT IDENTIFIER ::= { ibmStorageTankConformance 1 }
tankGroups      OBJECT IDENTIFIER ::= { ibmStorageTankConformance 2 }

tankCompliance MODULE-COMPLIANCE
      STATUS  current
      DESCRIPTION
              "The compliance statement for the TANK-MIB."
      MODULE  -- this module
          MANDATORY-GROUPS {
                              tankRequiredObjectsGroup,
                              tankNotifGroup
                            }
      ::= { tankCompliances 1 }

-- MIB groupings

tankRequiredObjectsGroup OBJECT-GROUP
     OBJECTS {
                tankClusterName,
                tankServerName,
                tankTimeStamp,
                tankMessageId,
                tankMessageText
              }
    STATUS current
    DESCRIPTION
        "The objects defined in this MIB module that MUST
         be implemented by a compliant implementation."
    ::= { tankGroups 1 }

tankNotifGroup NOTIFICATION-GROUP
     NOTIFICATIONS {
                    tankClusterStateChangeTrap,
                    tankServerStateChangeTrap,
                    tankStoragePoolSpaceTrap,
                    tankContainerQuotaTrap,
                    tankLogRotateTrap,
                    tankGenericTrap
                   }
     STATUS  current
     DESCRIPTION
        "All notifications defined in this MIB module MUST
         be implemented by a compliant implementation."
     ::= { tankGroups 2 }

END
```

*Figure 57. SNMP Trap MIB (Part 7 of 7)*

**Related topics:**

- "SNMP" on page 20
- "STC_TankEvents" on page 145

# Appendix C. Notices

This information was developed for products and services offered in the U.S.A.

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
Armonk, NY 10504-1785
U.S.A.

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
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 Corporation
MW9A/050
5600 Cottle Road
San Jose, CA 95193
U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

The following terms are trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the United States or other countries or both:

AIX
Enterprise Storage Server
IBM
IBM logo
FlashCopy
StorageTank
TotalStorage
WebSphere
xSeries

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

# Glossary

This glossary includes 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). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ANSI/EIA Standard - 440A: Fiber Optic Terminology*, copyright 1989 by the Electronics Industries Association (EIA). Copies can be purchased from the Electronics Industries Association, 2001 Pennsylvania Avenue N.W., Washington, D.C. 20006. Definitions are identified by the symbol (E) 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 of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The *Storage Networking Dictionary*, available online at the Storage Networking Industry Association (SNIA) Web site:

  www.snia.org/education/dictionary/
- The Distributed Management Task Force (www.dmtf.org), copyright 2003 by the Distributed Management Task Force, Inc., 225 SE Main Street Portland, OR 97214. Definitions derived from this book have the symbol (D) after the definition.

This glossary uses the following cross-reference forms:

**See** This 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** This refers the reader to one or more related terms.

**ACLI.** See *Administrative command-line interface (ACLI)*.

**Administrative command-line interface (ACLI).** A command-line interface used to administer all aspects of the SAN File System. The ACLI runs on all engines that host Metadata servers and the Administrative server.

**administrative log.** A log that maintains a history of messages created by the Administrative server.

**Administrative server.** For SAN File System, a set of servlets running within a customized instance of WebSphere Application Server that handles all SAN File System administrative requests from the SAN File System console. See also *SAN File System console*.

**alert.** A message or other indication that identifies a problem or an impending problem.

**audit log.** A log that maintains the history of all commands issued by any administrator for all Metadata servers in the cluster.

**CIM.** See *Common Information Model*.

**CIM client application.** A storage management program that initiates CIM requests to the Administrative agent for the device.

**CIM namespace.** The scope within which a CIM schema applies.

**CIM object manager (CIMOM).** The common conceptual framework for data management that receives, validates, and authenticates the CIM requests from the client application and then directs the requests to the appropriate component or device provider.

**CIMOM.** See *CIM object manager*.

**client.** For SAN File System, a client is a system that can access the SAN File System. These clients act as servers to a broader clientele, providing Network File System or Common Internet File System access to the global namespace or hosting applications (such as database servers or Web-hosting services that use multiple servers).

**class.** The definition of an object within a specific hierarchy. An object class can have properties and methods and serve as the target of an association.

**CLI.** See *Administrative command-line interface*.

**client state manager (CSM).** A component of the client kernel that provides protocol support for the client.

**cluster.** A group of engines that is managed as a set and presents a single point of control for configuration and service activity.

**cluster log.** A log that maintains a history of messages created by all Metadata servers in the cluster.

**cluster state.** A status condition of the cluster. Cluster states can be inactive (Not running or Forming), active (Online, Offline, Partly quiescent, or Fully quiescent) or unknown. See also *Forming, Fully quiescent, Not running, Offline, Online,* and *Partly quiescent*.

**Common Information Model (CIM).** A set of standards from the Distributed Management Task Force Inc. (DMTF). CIM provides a conceptual framework for storage management and an open approach to the design and implementation of storage systems, applications, databases, networks, and devices.

**coordinated universal time (UTC).** The time scale, based on the System International (SI) second, as defined and recommended by the Comitb Consultatif International de la Radio (CCIR) and maintained (using an atomic clock) by the Bureau International des Poids et Mesures (BIPM).

**CSM.** See *client state manager*.

**default user storage pool.** A storage pool that stores file data that SAN File System has not assigned (using the active policy) to a user storage pool, as well as file data that is assigned directly to this storage pool. There is only one default user storage pool; however, you can assign any user storage pool as the default storage pool. See also *user storage pool*

**engine.** The hardware unit that hosts the software for the Metadata server.

**event log.** (1) A log that maintains a history of event messages issued by all Metadata servers in the cluster. (2) IBM Term: A log that contains information about events for a particular system or group, for a particular metric, or for all the events that are associated with a specific monitor.

**file-placement rule.** A rule that controls in what pool SAN File System places files in the global namespace. See also *rule* and *global namespace*.

**fileset.** A hierarchical grouping of files managed as a unit for balancing workload across a cluster.

**FlashCopy image.** A space-efficient image of the contents of part of the SAN File System at a particular moment.

**Forming.** A status condition where the cluster has a master and is in the process of forming. This state is always the initial one whenever a cluster is newly formed.

**Fully quiescent.** A status condition that cuts off all client communication with the cluster.

**global namespace.** A single file system that provides complete, shared access to both Windows and UNIX clients in the same environment.

**ID.** See *identifier*.

**identifier (ID).** A sequence of bits or characters that identifies a user, program, device, or system to another user, program, device, or system.

**Initializing.** A status condition during which a Metadata server or the entire cluster is set up for the first time.

**key.** A property that is used to provide a unique identifier for an instance of a class. Key properties are marked with the Key qualifier. (D)

**lease.** The amount of time that a client can hold a lock.

**lock.** A restriction that allows clients to have exclusive access to files. Types of locks include *data locks*, *session locks*, and *range locks*.

**logical unit (LU).** In open systems, a logical disk drive.

**logical unit number (LUN).** In the small computer system interface (SCSI) protocol, a unique number used on a SCSI bus to enable it to differentiate between up to sixteen separate devices per SCSI ID address, each of which is a logical unit.

**LU.** See *logical unit*.

**LUN.** See *logical unit number*.

**managed object format (MOF).** A compiled language for defining classes and instances. A MOF compiler offers a textual means of adding data to the CIM Object Manager repository. MOF eliminates the need to write code, thus providing a simple and fast technique for modifying the CIM Object Manager repository. (D)

**master Metadata server.** In SAN File System, the Metadata server in a cluster that is responsible for physical-space allocation.

**metadata.** Data that describes the characteristics of stored data; descriptive data.

**Metadata server.** In SAN File System, a server that offloads the metadata processing from the data-storage environment to improve SAN performance. An instance

of the SAN File System runs on each engine, and together the Metadata servers form a cluster. See also *cluster*.

**method.** A way to implement a function on a class.

**MOF.** See *managed object format*.

**Not running.** A status condition where one or more servers in the cluster are not added and therefore the cluster cannot perform any functions.

**object name.** An object that consists of a CIM namespace path and a model path. The namespace path provides access to the CIM implementation managed by the CIM agent, and the model path provides navigation within the implementation.

**Offline.** A status condition during which clients are not being serviced and the cluster is responding only to administrative requests.

**Online.** A status condition that indicates the normal operational state for the cluster.

**Partly Quiesced.** A state in which the cluster or server is in a "quiet" client communications mode to allow other operations to occur.

**Partly quiescent.** A status condition that allows existing metadata activity and client communication to continue on the cluster, but prohibits new communication.

**policy.** A list of file-placement rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy is active at one time. See also *file-placement rule* and *service-class rule*.

**pool.** See *storage pool*.

**property.** An attribute that is used to characterize instances of a class.

**qualifier.** A value that provides additional information about a class, association, indication, method, method parameter, instance, property, or reference.

**quota.** A limit on the amount of disk space a user can use.

**rule.** The lines within a policy that specify which actions will occur when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups) and the fileset name associated with the object.

**SAN File System console.** A Web user interface used to monitor and control the SAN File System remotely by using any standard Web browser.

**schema.** A group of object classes defined for and applicable to a single namespace. Within the CIM agent, the supported schemas are loaded through the managed object format (MOF) compiler.

**security log.** A log that maintains a history of Administrative server login activity.

**service location protocol.** A directory service that the CIM client application calls to locate the CIMOM.

**Shutdown.** A status condition that describes when the cluster is shut down as intended.

**SLP.** See *service location protocol*.

**Starting.** A status condition when a Metadata server is starting as designed but is not ready to accept connections from clients.

**storage pool.** A named set of storage volumes that is the destination for storing client data.

**symbolic link.** A type of file that contains the path name of and acts as a pointer to another file or directory.

**system storage pool.** A storage pool that contains the system metadata (system and file attributes, configuration information, and Metadata server state) that is accessible to all Metadata servers in the cluster. There is only one system storage pool. See also *Metadata server*.

**user storage pool.** An optional storage pool that contains blocks of data that compose the files that are created by SAN File System clients. See also *storage pool* and *default user storage pool*.

**volume.** A labeled logical unit, which can be a physical device or a logical device. For SAN File System, there is a one to one relationship between volumes and LUNs. See also *logical unit number*.

**UTC.** See *coordinated universal time*

# Index

# M



# N



# O



# P



# Q



# R

## S



## T



## U



## V



## W

# Readers' Comments — We'd Like to Hear from You

**IBM TotalStorage™ SAN File System**
**(based on IBM Storage Tank™ technology)**
**System Management API Guide and Reference**
**Version 1 Release 1**

**Publication No. GA27-4315-00**

**Overall, how satisfied are you with the information in this book?**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Overall satisfaction | ☐ | ☐ | ☐ | ☐ | ☐ |

**How satisfied are you that the information in this book is:**

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Accurate | ☐ | ☐ | ☐ | ☐ | ☐ |
| Complete | ☐ | ☐ | ☐ | ☐ | ☐ |
| Easy to find | ☐ | ☐ | ☐ | ☐ | ☐ |
| Easy to understand | ☐ | ☐ | ☐ | ☐ | ☐ |
| Well organized | ☐ | ☐ | ☐ | ☐ | ☐ |
| Applicable to your tasks | ☐ | ☐ | ☐ | ☐ | ☐ |

**Please tell us how we can improve this book:**

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

**Readers' Comments — We'd Like to Hear from You**
GA27-4315-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corp
Dept. CGFA
PO Box 12195
Research Triangle Park, NC 27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

GA27-4315-00

Cut or Fold
Along Line

Printed in U.S.A.

GA27-4315-00