HP 3PAR InForm OS 3.1.1 CIM API Programming Reference

To use this document, you must be familiar with basic object oriented development techniques and with the following: Storage Management Initiative Specification (SMI-S), Common Information Model (CIM), Hypertext Transfer Protocol (HTTP), Secure Socket Layer (SSL), CIM Operations over HTTP, and the Unified Modeling Language (UML).

HP Part Number: QL226-96591
Published: June, 2012

Acknowledgments

For the latest version of this document, go to http://www.hp.com/go/3par/, navigate to your product page, click Support for your product, and then click Manuals.

Contents

1 Concepts

About SMI-S

SMI-S enables management of Storage Area Networks (SANs) in a heterogeneous, multi-vendor environment. SMI-S uses an object-oriented model based on the Common Information Model (CIM) to define objects and services which comprise a SAN. By leveraging vendor and technology independent standards, SMI-S allows management application vendors to create applications that work across products from multiple vendors.

The SMI-S model is divided into several profiles, each of which describes a particular class of SAN entities (such as disk arrays or Fibre Channel switches). These profiles allow for differences in implementations but provide a consistent approach for clients to discover and manage SAN resources and facilitate interoperability across vendor products within the SAN.

SMI-S also defines an automated resource discovery process using Service Location Protocol version 2 (SLPv2). This allows management applications to automatically find SAN resources and then probe them to determine which of the SMI-S profiles and features they support.

For more information, refer to the Storage Management Initiative Web site at www.snia.org/smi/home.

About the WBEM Initiative

SMI-S is based on the Web-Based Enterprise Management (WBEM) Initiative, which is defined by the Distributed Management Task Force (DMTF). WBEM is a set of management and Internet standard technologies developed to unify the management of distributed computing environments.

The DMTF has developed the following core set of standards that make up WBEM:

Table 1 Core Standards for WBEM Initiative

Standard	Description
The Common Information Model (CIM) standard	The CIM standard is the data model for WBEM. CIM provides a conceptual framework for describing management data for systems, networks, applications and services, and allows for vendor extensions. SMI-S uses CIM to model those objects and relationships that comprise a SAN.
CIM-XML	CIM-XML is a method of exchanging CIM management data. CIM-XML uses an xmlCIM payload and HTTP(s) as the transport mechanism. This protocol is defined by the following specifications: • *Specification for the Representation of CIM in XML* - Defines a standard for the representation of Common Information Model (CIM) elements and messages in XML, written in Document Type Definition (DTD). • *CIM Operations over HTTP* - Defines a mapping of CIM Messages onto HTTP that allows implementations of CIM to interoperate in an open, standardized manner. It uses the CIM XML DTD that defines the XML Schema for CIM objects and messages.
WBEM Discovery using Service Location Protocol (SLP)	WBEM Discovery using SLP is a method for applications to identify WBEM-based management systems. For more information regarding WBEM and CIM, refer to the DMTF web site at http://www.dmtf.org.

2 Security

CAUTION: The CIM API is not part of the evaluated Common Criteria storage system configuration and should not be used when operating in Common Criteria mode.

TCP Ports

The CIM-API uses dedicated TCP ports for CIM-XML communications and server location information. Two ports are specified by the DMTF and registered with IANA for CIM-XML communications between management clients and any CIM Server. The following table lists the TCP Ports for the CIM-XML communication and service location protocols:

Table 2 TCP Ports for CIM-XML Communication

Protocol	TCP Port
HTTP	5988 (default value)
HTTPS	5989 (default value)
Service Location (SLP)	427

Authentication

Authentication verifies the identity of an entity.

Management clients accessing the CIM Server are authenticated using a request/challenge mechanism using HTTP Basic authentication. When a request is received from a management client, the CIM Server challenges the client to send a user name and password encoded in the HTTP Authorization header. The user names and passwords used are the same as those used by other management interfaces and are case sensitive.

NOTE: CIM does not currently support LDAP user name and password authentication; only local user names and passwords are valid. Please see the *HP 3PAR Concepts Guide* for more information on local versus LDAP user credentials.

The CIM Server uses Open SSL to support HTTPS connections. The server supports SSLv3 and TLSv1 by default and uses the default Open SSL cipher list only. For more about OpenSSL, refer to http://www.openssl.org/docs.

NOTE: Because Basic Authentication means that client user names and passwords are sent over the wire in unencrypted form, it is recommended that the authentication is carried out either over a physically secure private network, or in conjunction with HTTPS.

Authorization

Authorization determines whether an entity that has already been authenticated is allowed to perform a given operation.

The CIM Server allows any authenticated user to retrieve CIM class and instance information. However, to invoke methods on CIM classes or instances, you must either have an Edit , Super, Administrator, or User permission level. Refer to the *HP 3PAR Concepts Guide* or the *HP 3PAR InForm OS CLI Administrators Manual* for complete information on authorization levels.

NOTE: Access to certain information concerning volumes, CPGs, etc., is controlled by the InForm OS. Therefore, if a user authenticates with the CIM API and only has access to certain domains, only those objects in those domains returned by the InForm OS. In addition, operations on those objects also be constrained at the domain level.

3 Configuring and Using the CIM Server

CLI Command Overview

Table 3 CLI Commands Used to Manage the CIM Server

Command	Purpose	Required Permission Level
`setcim`	Sets properties of the CIM Server. This command can be used to enable or disable certain services.	Super
`showcim`	Displays status information for the CIM Server.	Any
`startcim`	Enables the CIM Server.	Super
`stopcim`	Disables the CIM Server.	Super

NOTE: Refer to the *HP 3PAR InForm OS Command Line Interface Reference* for additional information.

Enabling and Disabling the CIM Server

By default, the CIM Server is disabled and must be manually enabled by an administrator before it can be used.

Enabling the CIM Server

To enable the CIM Server, issue the `startcim` command:

```
# startcim
CIM server will start in about 90 seconds.
```

Disabling the CIM Server

To disable the CIM Server, issue the `stopcim` command:

```
#stopcim -f
CIM server stopped successfully.
```

If you issue this command without the `-f` option, you are prompted to confirm your intent to disable the CIM Server.

Displaying the CIM Server Status

To display the CIM Server status information, issue the `showcim` command.

```
# showcim
-Service- -State- --SLP-- SLPPort -HTTP-- HTTPPort -HTTPS- HTTPSPort PGVer CIMVer
Enabled   Active  Enabled     427 Enabled     5988 Enabled      5989 2.9.1 3.1.1
```

The `showcim` command shows the overall status of the CIM Server; the status and ports used for the HTTP, HTTPS, and SLP; the version of the internal Pegasus CIM Object Manager; and the version of the CIM Server.

Displaying the CIM Server Version

The CIM Server version can be displayed using the `showcim` command as shown previously, or by using the `showversion -a` command.

```
# showversion -a
Release version 3.1.1
Patches:  None
Component Name                  Version
CLI Server                      3.1.1
...
CIM Server                      3.1.1
```

Communicating with the CIM Server

Several methods of communicating with the CIM Server are listed as follows:

Client Software

We do not provide client software, but commercial and open source CIM client software packages are readily available. Table 4 (page 12) lists some of the available tools and packages.

Table 4 Available Client Software Tools and Packages

Tool or Package	Description	URL
CIMNavigator	CimNavigator is a graphical browser tool written in Java that can be used to manipulate CIM objects hosted on CIMOMs on remote or local computers. We use this tool for internal development and testing.	http://www.cimnavigator.com/
WBEM Services	The WBEM Services project is an open-source Java implementation of Web Based Enterprise Management. This project consists of APIs, server and client applications and tools.	http://wbemservices.sourceforge.net
Ultimate CIM	Ultimate CIM is the Eclipse plug-in that enables developers to explore CIMOM, walk associations, execute WQL queries, graphically view class hierarchy and more.	http://sourceforge.net/projects/ultimatecim/

Sending Client Requests

Management clients send requests to the CIM Server. A valid client request must include:

- A properly formed HTTP header.
- A request must be addressed to the CIM Server on an HP 3PAR Storage System.
- This address consists of an IP Address and the appropriate TCP port for either HTTP or HTTPS as shown in the table that follows:

Protocol	TCP Port
HTTP	5988 (default value)
HTTPS	5989 (default value)

- The desired operation and its required parameters.

  For information about CIM operations, please refer to *CIM Operations over HTTP*.

- The object or objects on which the operation is to be performed.

  The payload of the client request is XML. For information about XML coding for CIM, refer to *Representation of CIM in XML*.

CIM Namespaces

CIM namespaces are logical groupings of CIM classes and CIM instances that represent managed objects in a particular environment. When making requests of a CIM Server, a client application must include the namespace that contains those classes. The namespace is case insensitive.

Table 5 CIM Server Supported Namespaces

Namespace	Purpose
`root/PG_Interop`	This is the Interop Namespace as defined in SMI-S. The Interop Namespace is used to find Registered Profiles and other CIM classes related to the SMI-S Server Profile.
`root/tpd`	The root/tpd namespace (**T**hree**P**ar**D**ata) is used to uniquely identify CIM Servers and provides management of all classes related to managing the storage system.

4 CIM API SMI-S Support

The CIM API supports the Storage Management Initiative Specification. The CIM API also supports extensions to these profiles to provide features that are not available from the standard profiles.

Standard Compliance

The CIM Server has successfully completed testing and conforms to elements of the Storage Networking Industry Association's Conformance Testing Program (SNIA-CTP) for the SMI specification. For detailed information, refer to:

www.snia.org/forums/smi/tech_programs/ctp/conformingproviders

SMI-S Functionality

SMI-S defines a number of services, profiles, and subprofiles that are used to manage elements of a SAN. Each HP 3PAR Storage System comes with an embedded CIM Server that includes support for the profiles, subprofiles, and services.

Supported Discovery Service

Table 6 Supported Discovery Service

Service	Version	Purpose
Service Location Protocol (SLP)	SLPV2	Allows management applications to discover the location of CIM Servers and the supported capabilities of those servers.

Supported Profiles

Table 7 Supported Profiles

Profile	Version	Purpose
Server Profile	1.1.0	Defines the capabilities of the CIM Server, including the supported SMI-S Profiles and Subprofiles supported.
Array Profile	1.1.0	Describes RAID arrays and disk storage systems.

Supported Array Packages

Table 8 Mandatory Supported Array Packages

Package	Version	Purpose
Block Services	1.1.0	Provides management of `StorageVolumes` and `StoragePools`.
Physical	1.1.0	Models information about a storage system's physical package and optionally about internal sub-packages.

Supported Array Subprofiles

Table 9 Supported Array Subprofiles

Subprofile	Version	Purpose
Location	1.1.0	Provides information about the physical location of a disk storage system.
Multiple Computer System	1.1.0	Provides information about the multiple computer systems (controller nodes) that cooperate to present a virtual computer system (cluster).
Software	1.1.0	Provides information on installed controller software.
Masking and Mapping	1.1.0	Provides management of LUNs.
FC Target Ports	1.1.0	Provides management of front end (host facing) Fibre Channel ports.
FC Initiator Ports	1.1.0	Provides management of back end (disk enclosure) FC ports.
iSCSI Target Ports	1.1.0	Provides management of front end (host facing) iSCSI ports.
Disk Drive Lite	1.1.0	Provides management of disk drive devices.
Block Server Performance	1.1.0	Provides performance statistics information for block servers.
Copy Services	1.1.0	Provides management of physical and virtual copies (remote copy is not supported).
Job Control	1.1.0	Provides management of tasks related to Copy Services and Replication Services.
Thin Provisioning	1.4	Provides management of thinly provisioned volumes and DynamicStoragePool (aka Common Provisioning Group).
Replication Services	1.4	Provides management of physical and virtual copies of a volume or volume set (remote copy is not supported).

Supported Server Subprofile

Table 10 Supported Server Subprofile

Subprofile	Version	Purpose
Profile Registration	1.4	Describes the set of classes and associations deal with profiles supported by the ObjectManager.

Supported SMI-S Levels by Release

Table 11 Supported SMI-S Levels by Release

InForm OS Release/ SMI-S Version		2.2.0/1.0.2	2.2.1/ 1.1.0	2.2.2/ 1.1.0	2.2.3/ 1.1.0	2.3.1/1.1.0	3.1.1/ 1.4.0
Profile	**Subprofile/Package**						
Server Profile		x	x	x	x	x	x
	Profile Registration						x
Array Profile		x	x	x	x	x	x
	Software	x	x	x	x	x	x
	Block Services		x	x	x	x	x
	Physical Package		x	x	x	x	x
	Masking & Mapping		x	x	x	x	x
	FC Target Ports		x	x	x	x	x
	FC Initiator Ports		x	x	x	x	x
	Location		x	x	x	x	x
	Mulitple Computer System		x	x	x	x	x
	Disk Drive Lite		x	x	x	x	x
	iSCSI Target Ports		x	x	x	x	x
	Block Server Performance				x	x	x
	Copy Services					x	x
	Job Control					x	x
	Thin Provisioning						x
	Replication Services						x

Discovery Service

The first step in managing an SMI-S enabled SAN is to discover its available resources (such as the CIM servers). SMI-S defines Service Location Protocol v2 (SLPv2) as the discovery mechanism.

CIM clients use SLP to query for available CIM servers. CIM servers answer with generic information about what services they provide and the URL at which these services reside.

Server Profile

A CIM Server is anything that supports the CIM-XML protocol and supports the basic read functional profile as defined by the CIM Operations over HTTP specification. After a CIM client has discovered a CIM server within a SAN, it then needs to determine the services provided by that server. The Server Profile defines the capabilities of the CIM Server. This includes the communication mechanisms the CIM Server supports, the CIM namespaces that it manages, and the SMI-S profiles and subprofiles that are supported.

For a complete discussion on the Server Profile model, please refer to SMI-S at http://www.snla.org.

Server Profile CIM Classes

Table 12 Server Profile CIM Key Classes

CIM Class	Description
TPD_CIMXMLCommunicationMechanism	Defines the communication protocols supported by the CIM server.
TPD_Namespace	Defines the CIM namespaces supported by the CIM Server.
TPD_ObjectManager	The CIM Object Manager service of the CIM Server.
TPD_RegisteredProfile	One instance of this class exists for each SMI-S Profile that is registered with the CIM Server.
TPD_RegisteredSubprofile	One instance of this class exists for each SMI-S Subprofile that is registered with the CIM Server.
TPD_StorageSystem	The storage server that is hosting the CIM Server.

Supported Methods

Table 13 Method for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

For the Server Profile Instance Diagram, refer to the SMI-S, v1.4 at http://www.snia.org.

Profile Registration Profile

Profile Registration Profile is mandatory for Server Profile. This profile models the profiles registered in the object manager and the associations between registration classes and the domain classes implementing the profile.

CIM Classes

Table 14 Key Classes

Class	Description
TPD_RegisteredProfile	One instance of this class will exist for each SMI-S Profile that is registered with the CIM Server. One instance exists for the SMI-S Profile.
TPD_RegisteredSubprofile	One instance of this class will exist for each SMI-S Subprofile that is registered with the CIM Server.

UML Diagram

For the Server Profile Instance Diagram, refer to the SMI-S, v1.4 at http://www.snia.org.

Registered Version

The RegisteredVersion property of the different RegisteredProfile/RegisteredSubprofile instances have different values; these values are the same as those published in the Registered Name and Version subclause of the profiles in SMI-S 1.4.

Note that the overall version of SMI-S supported by the provider are expressed using the SMI-S RegisteredProfile.

The following list the profile/subprofile corresponding RegisteredVersion:

- Array-1.3.0
- Server-1.4.0
- Disk Drive Lite-1.4.0
- Location-1.4.0
- Masking and Mapping-1.4.0
- Copy Services-1.4.0
- Job Control-1.3.0
- Multiple Computer System-1.2.0
- FC Target Ports-1.2.0
- FC Initiator Ports-1.3.0
- Software-1.4.0
- iSCSI Target Ports-1.2.0
- Block Server Performance-1.4.0
- Thin Provisioning-1.4.0
- Profile Registration-1.4.0
- Replication Services-1.4.0

SMI-S Profile

Each RegisteredProfile instance representing a profile is associated to a RegisteredProfile instance holding the SMI-S version number. The version number (RegisteredVersion) of SMI-S profiles may or may not be the same as the version number of the SMI-S RegisteredProfile. The RegisteredProfile instances are associated using ElementConformsToProfile, where the RegisteredProfile representing SMI storage profiles (e.g., Array, Server) is referenced from the ManagedElement role of the association. Also, this SMI-S profile is not associated to TPD_ObjMgrSoftwareIdentity, unlike the other RegisteredProfiles and RegisteredSubprofiles. It is also not advertised in SLP.

Note that the overall version of SMI-S supported by the provider are expressed using the SMI-S RegisteredProfile.

Table 15 Property Values for the SMI-S Profile

Properties	Description
InstanceID	SNIA:SMI-S
RegisteredOrganization	11 (SNIA)
RegisteredName	SMI-S

Table 15 Property Values for the SMI-S Profile *(continued)*

Properties	Description
RegisteredVersion	1.4.0 (version of the SMI-S the provider conforms to)
AdvertiseTypes	2 (Not Advertised)

Array Profile

The Array profile models the physical and logical aspects of disk storage systems. The basic Array profile provides a high level, read-only view of a storage system. The mandatory Block Services package and the various optional subprofiles extend this description and provide additional management capabilities.

For detailed information regarding the Array profile, refer to SMI-S at http://www.snla.org.

Array Profile CIM Classes

Table 16 Array Profile CIM Class

Class	Description
TPD_StorageSystem	Represents the storage array as a whole. The Dedicated property indicates this system operates as a storage array.

Supported Methods

Table 17 Method for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

Block Services Package

The Block Services package defines standards for viewing/managing storage capacity. This includes managing `StoragePools` and allocating capacity from `StoragePools` to be used by external consumers. Implementation of this package provides the ability to manage storage volumes on the HP 3PAR Storage System.

For detailed information regarding the Block Services package, refer to SMI-S at www.snla.org.

Block Services Package CIM Classes

Table 18 Block Services Package CIM Classes

Class	Description
TPD_StorageSystem	Represents the storage array as a whole.
TPD_StorageVolume	Storage volumes that are equivalent to SCSI Logical Units visible to consumers.
TPD_StoragePool	Allocatable space on the array.
TPD_StorageSetting	Used by clients as input to the `StorageConfigurationService`, `StorageSetting` describes the desired configuration of Storage Volumes.
TPD_StorageCapabilities	Provides information about the possible configuration parameters for elements created from a given `StoragePool`.

Table 18 Block Services Package CIM Classes *(continued)*

Class	Description
TPD_StorageConfigurationCapabilities	Provides information about what storage configuration functionality is provided by `StorageConfigurationService`.
TPD_StorageConfigurationService	Allows clients to configure Storage Volumes.
TPD_DynamicStoragePool	Storage Pool representing a Common Provisioning Group (CPG).

Supported Methods

Table 19 (page 20) - Table 22 (page 20) shows the supported methods for the Block Services Package:

Table 19 Methods for TPD_StorageCapabilities

Method	Description
CreateSetting()	Provides the ability to create `StorageSetting` objects for use in creating `StorageVolumes`.
GetSupportedStripeLengthRange()	Returns the possible stripe lengths for this capability.
GetSupportedStripeDepths()	Returns the possible stripe depths for this capability.

Table 20 Methods for TPD_StorageSetting

Method	Description
modifyInstance()	Provides the ability to modify a `StorageSetting` for use in creating `StorageVolumes`.
deleteInstance()	Provides the ability to delete a `StorageSetting`.

Table 21 Methods for TPD_StorageConfigurationService

Method	Description
CreateOrModifyElementFromStoragePool()	Provides the ability to create or modify `StorageVolumes` using a specific `StorageSetting` from a source `StoragePool`.
ReturnToStoragePool()	Provides the ability to remove `StorageVolumes`, returning the capacity to the original `StoragePool`.
CreateOrModifyStoragePool()	Provides the ability to create or modify `DynamicStoragePool` or `DeltaReplicaStoragePool` using a specific `StorageSetting` from a source `StoragePool`.
DeleteStoragePool()	Provides the ability to remove `DynamicStoragePool`, returning the capacity to the parent `StoragePool`.

Table 22 Method for TPD_StoragePool

Method	Description
GetSupportedSizeRanges()	Returns the supported size ranges for elements allocated from this Pool.

StorageSetting

There are eight types of `StorageSetting`:

1. Fixed `StorageSetting` for creating `Legacy Fully Provisioned Volume`.

   There are four of these, one for each RAID type, and each is associated to `3PAR:Capabilities:RAIDxStorageCapabilities`. Their `InstanceIDs` are `3PAR:StorageSetting:RAIDx`. A client can use `StorageSetting` to create a legacy, fully provisioned virtual volume. Most Logical Disk (LD) and virtual volume-specific properties are populated.

2. Fixed `StorageSetting` for creating `DynamicStoragePool`.

   There are four of these, one for each RAID type, and each is associated to `3PAR:Capabilities:RAIDxStorageCapabilities`. Their `InstanceIDs` are `3PAR:DynamicStoragePoolSetting:RAIDx`. A client can use this `StorageSetting` to create a `DynamicStoragePool`. Most LD properties are populated, but not those for virtual volumes.

3. Fixed `StorageSetting` for creating `Volume From DynamicStoragePool`.

   There are two of these settings per DynamicStoragePool, and each is associated to the 3PAR:Capabilities for DynamicStoragePool <poolname> StorageCapabilities of the DynamicStoragePool. One has an InstanceID of 3PAR:ThinProvisionedVolumeSetting from DynamicStoragePool <poolname> that a client can use for creating a TP volume from the DynamicStoragePool; the other has an instance of 3PAR:FullyProvisionedVolumeSetting from DynamicStoragePool <poolname> that a client can use for creating a fully provisioned volume from the DynamicStoragePool. Only volume-related properties will be populated.

4. `TPD_VolumeSetting`.

   This StorageSetting contains the current configuration of a volume. This is associated to the `StorageVolume` via `TPD_ElementSettingData`. There is no association to any `TPD_StorageCapabilities`.

5. `StorageSetting` generated from of`"CreateSetting 3PAR:Capabilities:RAIDx"`capabilities.

   `StorageSetting` created this way is generic enough that it encompasses LD, virtual volume and snap space specific properties. A client has to tailor the properties carefully to create the element (DSP, legacy volume or new-style volume) it desires. This setting has a `StorageSettingGeneratedFromCapabilities` association to the “3PAR:Capabilities:RAIDx” capabilities.

6. `StorageSetting` generated from `CreateSetting` of `"3PAR:Capabilities for DynamicStoragePool" Capabilities`.

   `StorageSetting` created this way can be used by a client for user space or snapshot space of a virtual volume. By default the `ThinlyProvisioned` property is set to **FALSE**. A client has to modify this to **TRUE** if a TPVV is desired. This setting has a `StorageSettingGeneratedFromCapabilities` association to the `"3PAR:Capabilities for DynamicStoragePool:<poolname>" Capabilities`.

7. `StorageSetting` created by `CreateInstance`.

   `StorageSetting` contains values that are totally determined by the client who created it. The most likely usage for this is if a client clones an existing fixed `StorageSetting` and then modifies some of the property values before creating it. This Setting has a `StorageSettingGeneratedFromCapabilities` association to the appropriate `"3PAR:Capabilities:RAIDx" Capabilities` according to the RAID type of the `StorageSetting`.The default is RAID10.

8. Persistent StorageSetting/Template.

   StorageSetting (templates) that are stored in Persistent Repository. The TemplateType and TemplateForVolumeType properties of the StorageSetting restrict what kind of element creation this setting can be applied to. This Setting has a StorageSettingGeneratedFromCapabilities association to the appropriate "3PAR:Capabilities:RAIDx" Capabilities according to the RAID type of the StorageSetting. The default is RAID10.

Table 23 Relevant Properties of StorageSetting

Properties	Description
Uint64 **AllocationUnit**	Grow increment in bytes. When a TP StorageVolume or a DynamicStoragePool grows on demand, the system allocates this number of bytes on each allocation.
Uint64 **AllocationUnitMax**	Maximum allocation unit in bytes.
Uint64 **AllocationUnitMin**	Minimum allocation unit in bytes.
Uint64 **SpaceLimit**	Snap space limit in bytes. If SpaceLimit (in bytes) is greater than zero, the space consumed by the storage element shall not exceed the value of SpaceLimit. For TP StorageVolume, if SpaceLimit is zero, allocation limit is not enforced (e.g. the volume can grow until the system runs out of space).
Uint16 **SpaceLimitWarningThreshold**	Snap space limit warning in percentage of SpaceLimit. When actual space allocated to the TP StorageVolume or DynamicStoragePool hits this threshold, alerts are generated. For example, if SpaceLimitWarningThreshold is 90% and SpaceLimit is 100GB, alerts are generated when allocated space is 90GB or more.
Uint64 **UserSpaceLimit**	User space limit in bytes. If UserSpaceLimit (in bytes) is greater than zero, the space consumed by the storage element shall not exceed the value of UserSpaceLimit. For TP StorageVolume , if UserSpaceLimit is zero, allocation limit is not enforced (e.g. the volume can grow until the system runs out of space).
Uint16 **UserSpaceLimitWarningThreshold**	User space limit warning in percentage of UserSpaceLimit. When the actual space allocated to the TP StorageVolume or DynamicStoragePool hits this threshold, alerts are generated. For example, if UserSpaceLimitWarningThreshold is 90% and UserSpaceLimit is 100GB, alerts are generated when allocated space is 90GB or more.
Uint32 **UserSpaceAllocationWarning**	TP volume user space allocation warning in percentage of volume size. If this is set, UserSpaceLimitWarningThreshold is ignored. UserSpaceLimitWarningThreshold calculation requires UserSpaceLimit , which may not be specified. UserSpaceAllocationWarning thus allows a percentage to be specified without requiring UserSpaceLimit to be specified (in CLI, one can specify –usr_aw option without specifying –usr_al option). Applies only to TPVVs.
Uint32 **SnapSpaceAllocationWarning**	Volume snap space allocation warning in percentage of volume size. If this is set, SpaceLimitWarningThreshold is ignored. SpaceLimitWarningThreshold calculation requires SpaceLimit, which may not be specified. SnapSpaceAllocationWarning thus allows a percentage to be specified without requiring SpaceLimit to be specified (in CLI, one can specify –snp_aw option without specifying –snp_al option).

Table 23 Relevant Properties of StorageSetting *(continued)*

Properties	Description
Uint64 **DSPSnapSpaceGrowWarning**	Warning alert is issued if snap space of a CPG exceeds `DSPSnapSpaceGrowWarning`. 0 means no warning limit is enforced. This value is in bytes. Applies to `DynamicStoragePool` only.
Uint16 **IntendedUsage**	Describes how the elements are used. For example, if a new `TP StorageVolume` is used for snap shot, this value is 7. The default value is 0 ("not specialized").
Uint16 **LowSpaceWarningThreshold**	Generates a low space indication and applies only when creating a `DynamicStoragePool`. If non-zero, a low space indication is generated when `RemainingManagedSpace` of `DynamicStoragePool` <= (`TotalManagedSpace of DynamicStoragePool * LowSpaceWarningThreshold`)/100. If it is zero, no indication is generated. Unit is in percentage. This corresponds to the `-aw` option of the CLI `createcpg` command.
Boolean **ThinlyProvisioned**	True if the corresponding volume is thinly provisioned.
Uint16 **TemplateType**	Type of template to be created (VV or CPG).
Uint16 **TemplateForVolumeType**	Type of VV template to be created (TPVV, CPVV or all).
Uint16 **ThinProvisionedPoolType**	The type of thin provisioned pool used when StorageSetting is used a goal for creating a thin provisioned pool. In other contexts, this property is undefined. The possible values match the appropriate values in `StorageConfigrationCapabilities.SupportedStorageElementTypes`.
Uint16 **StoragePoolInitialUsage**	The Usage value to be used when creating a new StoragePool.

Table 24 Matrix of Populated Properties for Different Types of StorageSetting

Properties	Default Values for Each StorageSetting Type					
	Legacy Volume Setting	DynamicStoragePool Setting	Thin Provisioned Volume Setting	Full Provisioned Volume Setting	Generated from "3PAR:Capabilities: RAIDx" CreateSetting	Generated from "3PAR:Capabilities for DynamicStoragePool" CreateSetting
ThinlyProvisioned	F	NULL	T	F	F	F
AllocationUnit	NULL	Default	NULL	NULL	Default	NULL
AllocationUnitMax	NULL	Max	NULL	NULL	Max if goal setting. Default if default setting.	NULL
AllocationUnitMin	NULL	Min	NULL	NULL	Min if goal setting. Default if default setting.	NULL
SpaceLimit	NULL	0	NULL	NULL	0	0

Table 24 Matrix of Populated Properties for Different Types of StorageSetting *(continued)*

Properties	Default Values for Each StorageSetting Type					
SpaceLimitWarningThreshold	NULL	0	NULL	NULL	0	0
UserSpaceLimit	NULL	NULL	NULL	NULL	NULL	NULL
UserSpaceLimitWarningThreshold	NULL	NULL	NULL	NULL	NULL	NULL
UserSpaceAllocationWarning	NULL	NULL	NULL	NULL	NULL	NULL
SnapSpaceAllocationWarning	NULL	NULL	NULL	NULL	NULL	NULL
LowSpaceWarningThreshold	NULL	0	NULL	NULL	NULL	NULL
DSPSnapSpaceGrowWarning	NULL	0	NULL	NULL	NULL	NULL
LD Parameters (e.g., rs, ss, ha, pattern etc.)	Default	Default	NULL	NULL	Default	NULL
Volume Parameters (e.g., spt, hpc, policy etc.)	Default	NULL	Default	Default	Default	Default
TemplateType	Volume	CPG	Volume	Volume	Volume	Volume
TemplateForVolumeType	Any	NULL	TPVV	TPVV	NULL	Any
ThinlyProvisionedPoolType	NULL	ThinlyProvisionedLimitlessStoragePool (9)	NULL	NULL	NULL	NULL

Creating a DynamicStoragePool (CPG) via SMI-S

Creating a `DynamicStoragePool` is a two-step process:

1. Choose or create a `StorageSetting` instance using `CreateSetting()`.
   - If choosing a pre-created one, a client should use a `StorageSetting` with InstanceID="3PAR:DynamicStoragePoolSetting:RAIDx".
   - If generating a `StorageSetting`, a client should invoke the `CreateSetting` method on one of the `StorageCapabilties` with InstanceID="3PAR:Capabilities:RAIDx".
   - By default, `StorageSetting.ThinProvisionedPoolType` is set to `ThinlyProvisionedLimitlessStoragePool(9)`, which creates a DSP with limitless capacity. The size parameter to `CreateOrModifyStoragePool` is ignored.
   - A client can also set `ThinProvisionedPoolType`to `ThinlyProvisionedQuotaStoragePool(8)`, which means there is an upper limit to how big the DSP can grow. For this, the `Size` parameter has to be specified and non-zero.
   - `ThinProvisionedPoolType` cannot be set to `ThinlyProvisionedAllocatedStoragePool(7)` since this is not supported.
   - `StorageSetting.StoragePoolInitialUsage` must either be `NULL` or `Unrestricted(2)`.

2. Invoking the StorageConfigurationService's CreateOrModifyStoragePool method: a client passes in the StorageSetting obtained in step 1 and other relevant input parameters to create a DynamicStoragePool.
   - The input StorageSetting can be deleted by StorageSetting.DeleteInstance, or it can be made persistent via ModifyInstance of the StorageSetting.ChangeableType property to "Persistent" and StorageSetting.TemplateType property to "DynamicStoragePool".
   - The method CreateOrModifyStoragePool (existing method) of StorageConfigurationService is used to create a DynamicStoragePool (DSP). Only DynamicStoragePool can be created; concrete pool cannot be created.

   The method signature is described below:

```
uint32 CreateOrModifyStoragePool
[In] String ElementName,
[Out] CIM_ConcreteJob ref Job,
[In] StorageSetting ref Goal,
[In, Out] Uint64 Size,
[In] String InPools[],
[In] String InExtents[],
[In, Out] DynamicStoragePool ref Pool);
```

   - ElementName - The end user's input name of the DynamicStoragePool (CPG) to be created.
   - Job - Is always NULL.
   - Goal - This is an instance of StorageSetting describing characteristics of the desired DynamicStoragePool. If it is NULL, the provider picks the default setting. The ThinProvisionedPoolType property specifies what type of DSP to create: (ThinlyProvisionedQuotaStoragePool(8) for a DSP with an upper limit, or ThinlyProvisionedLimitlessStoragePool(9) for a DSP with no upper bound).
   - Size - The virtual size of the DynamicStoragePool. On initial creation, the system does not allocate any space to the pool. The minimum unit (the grow increment) is allocated for this pool only after a volume is created off this pool. If the DSP has a size limit, then ThinProvisionedPoolType property in Goal has to be set to ThinlyProvisionedQuotaStoragePool(8). To create a "limitless" DynamicStoragePool (e.g., creating a CPG without specifying the size), the ThinProvisionedPoolType property in Goal has to be set to ThinlyProvisionedLimitlessStoragePool(9), and this parameter is ignored.
   - InPools - This shall contain an array of StoragePool where the new pool is to be created from. However, only support creation from *"3PAR:all-FC"*, *"3PAR:all-NL"*, or *"3PAR:all-SSD"* pools is supported. Therefore, this array should contain only one element. If any other values are passed in, an "Invalid Parameter" error is returned.
   - InExtents - This should be a null array because creating DynamicStoragePool from StorageExtents is not supported. Non-null values are ignored.
   - Pool - It should be null as an input to create a new DynamicStoragePool. Upon method completion, it contains the object path of the newly created DynamicStoragePool. If it is not null, it refers to an existing DynamicStoragePool whose properties are being modified based on the input Goal (StorageSetting).

Table 25 DynamicStoragePool Possible Return Values

ValueMap	Values	Explanation/Notes

Table 25 DynamicStoragePool Possible Return Values *(continued)*

ValueMap	Values	Explanation/Notes
0	Job Completed with No Error	
1	Not supported	
2	Unknown	
3	Timeout	
4	Failed	
5	Invalid parameter	
6	In Use	
4097	Size Not Supported	

Relevant Properties of DynamicStoragePool

Table 26 Relevant Properties of DynamicStoragePool

Properties	Description
string **PoolID**	A unique ID within the storage array. The user's defined name is used as an identifier. **NOTE:** DSP does not have a WWN because it is not considered a physical entity.
Uint64 **TotalManagedSpace**	This is the actual allocated size to the DSP (not the logical size). Logical size is denoted by `SpaceLimit`.
Uint16 **LowSpaceWarningThreshold** (percentage)	The unit is a percentage. If it is zero, no alert is generated. If it is non-zero, an indication is generated when `RemainingManagedSpace <= (TotalManagedSpacet*LowSpaceWarningThreshold)/100`.
Unit64 **AllocationUnit** (bytes)	This is the grow increment. When a TP `StorageVolume` or a `DynamicStoragePool` grows on demand, the system allocates this number of bytes on each allocation.
Uint16 **SpaceLimitDetermination**	Defines the approach associated with the pool for determining capacity information for the pool. Allocated (2) means that the advertised capacity is the same as the actual capacity. Quota (3) means that there is an administratively defined limit (SpaceLimit) on the capacity that may be used to create or expand child elements. Limitless (4) means that there is no defined limit on the capacity for creating or expanding children.
Uint64 **SpaceLimit**	The capacity of the storage allocated to the pool when `SpaceLimitDetermination` has the value 3 (Quota) or 4 (Limitless) or is set to the value of `TotalManagedSpace` if `SpaceLimitDetermination` has the value 2 (Allocated). In the case of Limitless, `SpaceLimit` is set to 1PB, the maximum allowed by the HP 3PAR Storage System.
Uint64 **ThinProvisionMetaDataSpace** (bytes)	The size of metadata consumed by this storage pool. This is only defined if the pool is thin provisioned.
Uint64 **CurrentSpaceConsumed**	This is the current number of bytes being used by all TP `StorageVolumes` (RAID overhead already taken into

Table 26 Relevant Properties of DynamicStoragePool *(continued)*

Properties	Description
(bytes)	account) allocated from this pool. For example, a RAID-1 CPG with 50 GB being allocated to TPVVs. The `CurrentSpaceConsumed` is 50 GB.
Uint64 **ConsumedSnapDataSpace** (bytes)	This is the nominal number of bytes allocated on disk for Snapshot data (SD) of this pool, not including RAID overhead.
Uint64 **ConsumedSnapDataSpacePlusMetaData** (bytes)	This is the actual number of bytes allocated on disk for Snapshot data (SD) space of this pool, including RAID overhead.
Uint64 **ConsumedSnapAdminSpace** (bytes)	This is the nominal number of bytes allocated on disk for Snapshot admin (SA) of this pool, not including RAID overhead.
Uint64 **ConsumedSnapAdminSpacePlusMetaData** (bytes)	This is the actual number of bytes allocated on disk for SA space of this pool, including RAID overhead.
Uint64 **RemainingManagedSpace** (bytes)	The remaining usable capacity after the allocation of TP `StorageVolumes`. It is calculated as follows: `RemainingManagedSpace=SpaceLimit(virtual size)- CurrentSpaceConsumed`.
Uint32 **GetSupportedSizeRange()**	This method returns the maximum and minimum size, in bytes, of a `StorageVolume` that can be created off this DSP. Only the `StorageVolume` and `Thin Provisioned Volume` values are accepted for the `ElementType` parameter.
Uint32 **GetSupportedSizes()**	This method is not supported.
Uint64 **SnapDataSpaceWarningLimit** (bytes)	Issue warning alert when space allocation exceeds this amount. A size of 0 means no warning limit is enforced. Default is 0.

Modifiable StorageSetting Properties for DSP

When creating or modifying a DSP, a client can change the following `StorageSetting` properties to a non-default value. The provider ignores all other properties.

Table 27 Summary of Modifiable StorageSetting Properties for DSP

StorageSetting Property	CLI Equivalence	Comment
SpaceLimit	-sdgl	
SpaceLimitWarningThreshold/ DSPSnapSpaceGrowWarning	-sdgw	If both the `DSPSnapSpaceGrowWarning` and `SpaceLimitWarningThreshold` properties are specified, `DSPSnapSpaceGrowWarning` takes precedence. This is because `SpaceLimitWarningThreshold` is in percentage of `SpaceLimit` (whereas `DSPSnapSpaceGrowWarning` is in

Table 27 Summary of Modifiable StorageSetting Properties for DSP *(continued)*

StorageSetting Property	CLI Equivalence	Comment
		bytes), and for the HP 3PAR Storage System, we set the absolute limit, not in percentage, hence the need for proprietary property `DSPSnapSpaceGrowWarning`. If `DSPSnapSpaceGrowWarning` is NULL and BOTH `SpaceLimitWarningThreshold` and `SpaceLimit` are specified, the numbers are converted to bytes for internal usage.
LowSpaceWarningThreshold	-aw	
AllocationUnit	-sdgs	
Domain	-domain	Can only be specified during creation; is ignored for modification operation.
DataRedundancyGoal	-ssz	Modifiable only for RAID-10.
ParitySetSize	-ssz	Modifiable only for RAID-50 and RAID-60.
ExtentStripeLength	-rs	Row size.
UserDataStripeDepth	-sz	Step size.
HighAvailability	-ha	
ChunkletLocationPreference	-ch	
Disk pattern properties	-p	e.g. `DiskPrimPathNodes`, `DiskPrimRPM`, etc.
Corresponding Template properties	-templ	Only if using persistent `StorageSetting`, i.e., template.
ThinProvisionedPoolType		Only Quota (if sdgl is specified) and Limitless (if no sdgl is specified) are supported.

StorageCapabilities of DynamicStoragePool

After a DSP is created, an instance of `StorageCapabilities` is created, and the `StorageSetting` properties that are used to create the DSP are copied into this `StorageCapabilities`, which is then bound to the DSP via a `CapabilitiesOfStoragePool` association.

StoragePoolConfigurationCapabilities of DynamicStoragePool

After a DSP is created, an instance of `StoragePoolConfigurationCapabilities` is created and is associated to the DSP via a `CapabilitiesOfStoragePoolConfiguration` association.

As a matter of fact, each StoragePool in the system (`TPD_StoragePool`, either `primordial` or `concrete`, `TPD_DynamicStoragePool` and `TPD_DeltaReplicaStoragePool`) has an associated `StoragePoolConfigurationCapabilities` instance.

There are four instances per system, one for each kind of StoragePool: `PrimordialPool`, `ConcretePool`, `DynamicStoragePool` and `DeltaReplicaStoragePool`.

Table 28 Instances for StoragePoolConfigurationCapabilities

	StoragePoolConfigurationCapabilities			
	PrimordialPool	**ConcretePool**	**DynamicStoragePool**	**DeltaReplicaStoragePool**
SupportedSynchronousActions	NULL	Storage Pool Creation (2), Storage Pool Deletion (3), Storage Pool Modification (4).	Storage Element Creation (5), Storage Element Return (6), Storage Element Modification (7).	NULL
SupportedStoragePoolFeatures	NULL	Single InPool (3)	NULL	NULL
SupportedStorageElementFeatures	NULL	StorageVolume Creation (3), StorageVolume Modification (5), Single InPool (6).	StorageVolume Creation (3), StorageVolume Modification (5), Single InPool (6).	NULL
SupportedStorageElementTypes	NULL	StorageVolume (2), ThinlyProvisionedQuotaStoragePool (8), ThinlyProvisionedLimitlessStoragePool (9).	StorageVolume (2), ThinlyProvisionedStorageVolume (5).	StorageVolume(2)

Creating a Volume Through SMI-S

There are two extrinsic methods required to create a volume. The `CreateSetting` method in `TPD_StorageCapabilities` class generates the settings that are used to create the Volume, and the `CreateOrModifyElementFromStoragePool` method in `TPD_StorageConfigurationService` class creates, modifies or grows the volume with the `StorageSetting` instance created by the `CreateSetting` method. The `ReturnToStoragePool` method is used to delete a volume.

Additionally, you can use the intrinsic method with the `TPD_StorageSetting` class. The `ModifyInstance` method is used to modify the `StorageSetting` properties that are returned by the `CreateSetting` method.

A new method `TPD_CreateOrModifyElementFromStoragePools` can also be used to create, modify, and grow virtual volumes. This method is similar to `CreateOrModifyElementFromStoragePool`, but instead of the `InPool` parameter, there is an `InPools` parameter which accepts an array of references to StoragePool, with the first element containing reference to the parent pool from which the volume allocate its user space, and the second element containing reference to the parent pool from which the volume allocate its snapshot space.

Supported Method For Creating a Storage Volume

Table 29 Method for TPD_StorageConfigurationService

Method	**Description**
CreateOrModifyElementFromStoragePool()	Allows clients to create a `StorageVolume`.

Method Signature

The method signature is described as follows:

```
uint32 CreateOrModifyElementFromStoragePool (
     [In] String ElementName,
     [In,
        Values {"StorageVolume", "StorageExtent", "LogicalDisk",
  "ThinlyProvisionedStorageVolume",
  "ThinlyProvisionedLogicalDisk",
  "ThinlyProvisionedAllocatedStoragePool",
```

```
  "ThinlyProvisionedQuotaStoragePool",
  "ThinlyProvisionedLimitlessStoragePool"},
  ValueMap{"2","3","4","5","6","7","8","9"}]
Uint16 ElementType,
[Out] CIM_ConcreteJob ref Job,
[In] CIM_StorageSetting ref Goal,
[In, Out] Uint64 Size,
[In] CIM_StoragePool ref InPool,
[In, Out] CIM_LogicalElement ref TheElement);
```

- **ElementName**: The name of the volume to be created or expanded.
- **ElementType**: This enumeration specifies what type of object to create.
  - `StorageVolume(2)` and `ThinlyProvisionedStorageVolume(5)` are supported. If the `InPool` parameter contains a reference to a concrete pool, this can only be `StorageVolume(2)`.
- **Job**: If a Job was created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter. Job control is not supported. This output parameter is always NULL.
- **Goal**: This is the Service Level that the `StorageVolume` option is expected to provide. The Setting is a subset of the Capabilities available from the parent `StoragePool`. The Goal can be a null value, in which case the default Setting for the Pool is used. The Goal can refer to one of the preexisting `StorageSetting` instances representing RAID types, or the `StorageSetting` generated by a `CreateSetting` method, or from a `StorageSetting` option that is associated with another existing `StorageVolume` option.
- **Size**: As an input, this is the desired logical size in bytes of the `StorageVolume`. If it is not possible to create a volume of the desired size, a return code of `Size not supported` is returned with the size set to the nearest supported size. If it is a growing volume request, the size is the new size of the volume. As an output, size is the actual size allocated for the volume. It might be a little larger than the input size because the allocation unit is based on chunklet size (256 MB).
- **InPool**: This option contains the reference to the source StoragePool.
  - This can either be a reference to `Concrete` pool or `DynamicStoragePool`. If this contains a reference to `Concrete` pool, then only a fully provisioned volume can be created and the ElementType has to be set to `StorageVolume(2)`. If this contains a reference to `DynamicStoragePool`, then the volume that is created is a fully provisioned volume, if the ElementType is set to `StorageVolume(2)`, or `TPVV`, if ElementType is set to `ThinlyProvisionedStorageVolume(5)`. It is not valid to have a combination of `ThinlyProvisionedStorageVolume(5)` for the ElementType and a reference to the `Concrete` pool for `InPool` parameter. If this parameter is NULL and ElementType is set to `StorageVolume(2)`, then by default `StorageVolume` is created from FC pool.
- **TheElement**: As an input, if it is not null, the method is called to modify (either set or grow) the existing `StorageVolume`. If it is null (on input), the method is called to create a new `StorageVolume`. As an output, it contains the newly created `StorageVolume` options.

Table 30 Return Values for CreateOrModifyElementFromStoragePool()

ValueMap	Values	Explanation/Notes
0	Job Completed with No Error	
4	Failed	

Table 30 Return Values for CreateOrModifyElementFromStoragePool() *(continued)*

ValueMap	Values	Explanation/Notes
5	Invalid Parameter	
4097	Size Not Supported	A volume with the specified size cannot be created or grown.

Creating Data and Snapshot Space from Two Storage Pools

Table 31 (page 31) shows the supported method for creating data and snapshot space from two storage pools:

Table 31 Method for TPD_StorageConfigurationService

Method	Description
TPD_CreateOrModifyElementFromStoragePools	Supports data and snapshot spaces, allocating space from two different `DynamicStoragePools`. This method can also be used to modify existing TP `StorageVolume`, for example, increasing its size or changing other properties.

This method is useful for creating a default volume that has snap space without the need for specifying a Goal parameter.

This method can also be used to create fully-provisioned volumes that draw user space from a DSP.

NOTE: This method does not support creating a legacy volume where user space is allocated from a concrete "all" pool, although `CreateOrModifyElementFromStoragePool` still supports this.

Method Signature for CreateOrModifyElementFromStoragePools

The method signature for `CreateOrModifyElementFromStoragePools` is described as follows:

```
uint32 TPD_CreateOrModifyElementFromStoragePools (
 [In] String ElementName,
 [In,
Values {"StorageVolume","ThinlyProvisionedStorageVolume"},
ValueMap{"2", "5"}]
Uint16 ElementType,
[Out] CIM_ConcreteJob ref Job,
 [In] CIM_StorageSetting ref Goal,
 [In] CIM_StoragePool ref InPools[],
 [In, Out] Uint64 Size,
 [In, Out] CIM_LogicalElement ref TheElement);
```

- **ElementName**: The name of the `TP StorageVolume` (TPVV) or `StorageVolume` (CPVV) to be created.
- **ElementType**: This enumeration specifies what type of object to create. `StorageVolume(2)` and `ThinlyProvisionedStorageVolume(5)` are supported. If other values are passed in, the provider should throw an "operation not supported" exception.
- **Job**: If a Job was created as a side-effect of the execution of the method, a reference to that Job is returned through this parameter. Job control is not supported. This output parameter is always NULL.
- **InPool**: This array contains string references (object path) of `DynamicStoragePool` where the volume draws space. The first element contains `DynamicStoragePool` for user data. The second element contains a `DynamicStoragePool` for snapshot space. The snapshot space properties (allocation warning and space limit threshold, etc …) from each pool are

described in the corresponding Goals array. For TPVV or CPVV, the array size may be two, even if both its user and snap space are drawn from the same DSP. For a fully-provisioned volume or a TPVV that is not a CPVV, the array size must be one, corresponding to only the user space. InPools cannot contain anything other than `DynamicStoragePool`.

- **Goal**: This contains object path of `StorageSetting`, describing quality of service the new `StorageVolume` must provide.
- **Size**: As an input, this is the desired logical size in bytes of the TP `StorageVolume` or the requested size of the `StorageVolume`. If it is a modification request, "Size" is the new size which must be greater than the existing size (e.g. volume can grow, but can not shrink). For TP `StorageVolume`, the system actually allocates only minimum disk space to back up the volume. More space is allocated on demand as more write data occur. The allocation unit for each space allocation is the "growth increment" size. The system continues to allocate more space until either the system runs out of disk space or maximum allocation of the volume (SpaceLimit) is reached.
- **TheElement**: Should be null as an input to create a new `TP StorageVolume` or `StorageVolume`. Upon method completion, it contains the object path of the newly created TP StorageVolume. If it is not null, it refers to the TP `StorageVolume` to be modified (e.g., changing size, allocation warning, etc.).

Table 32 Return Values for TPD_CreateOrModifyElementFromStoragePools ()

ValueMap	Values	Explanation/Notes
0	Job Completed with No Error	
1	Not supported	
2	Unknown	
3	Timeout	
4	Failed	
5	Invalid Parameter	
6	In Use	
4097	Size Not Supported	

CreateSetting

Table 33 Method for TPD_StorageCapabilities.CreateSetting

Method	Description
CreateSetting()	Allows clients to create a new `StorageSetting` instance based on existing instances.

This method returns a new instance to the client with default property values. There are four instances of `StorageCapabilities` (one for each RAID type). The type of `StorageSetting` the client wants to create is determined by the referenced `StorageCapabilities`. For example, if the client invokes `CreateSetting()` method on a RAID-0 `StorageCapabilities` instance, then a RAID-0 `StorageSetting` instance is returned. The client can use intrinsic method `ModifyInstance()` to customize the properties as needed.

```
Uint32 CreateSetting (
 [in] uint16 SettingType,
 [Out] CIM_StorageSetting REF NewSetting);
```

- **SettingType (in)**: If Default (2) is passed, the Max, Goal, and Min setting attributes are set to the Default values of the parent `StorageCapabilities` option when the instance is created. If they are set to Goal (3), the new `StorageSetting` attributes are set to the related attributes of the parent `StorageCapabilities` options, such as Min to Min, Goal to Default, and Max to Max.
- **NewSetting (out)**: Reference to a new `StorageSetting` instance based on the parent `StorageCapabilities`. For example, if the parent `StorageCapabilities` option is the RAID-0 instance, then this instance is a copy of the pre-created RAID-0 `StorageSetting` instance.

Table 34 Return Values for CreateSetting ()

ValueMap	Values	Explanation/Notes
0	Method Completed OK	
4	Failed	
5	Invalid Parameters	

Creating a Volume with a Default Setting

A volume can be created using one of the fixed `StorageSetting` from the four available RAID types. The volume is then created with the default values associated with each RAID type. If no `StorageSetting` reference is passed into `CreateOrModifyElementFromStoragePool`, then a volume with default RAID-10 properties is created. To create a legacy volume with FC drives, use reference to FC `StoragePool` as the `InPool` parameter. To create a legacy volume with SSDs, use reference to SSD `StoragePool` as the `InPool` parameter. To create a legacy volume with NL drives, use reference to NL `StoragePool` as the `InPool` parameter. To create a thinly provisioned volume or a fully provisioned volume that allocates from a `DynamicStoragepool`, use reference to `DynamicStoragePool` as the `InPool` parameter; for TPVV, the `ElementType` parameter should be set to `ThinlyProvisionedStorageVolume(5)`. For fully provisioned volume, the `ElementType` parameter should be set to `StorageVolume(2)`. If the `InPool` parameter is NULL, by default FC drives are used to create the volume.

The fixed `TPD_StorageSetting` instances can be found by walking the `StorageSettingAssociatedToCapabilities` association from `TPD_StorageCapabilities`.

Creating a Volume with a Non-Default Setting

The following are the steps necessary for an SMI-S client to create a volume with non-default settings:

1. Invoke the `CreateSetting` method to get at a reference to the `TPD_StorageSetting`. The SMI-S provider creates, if one does not exist, an instance of the default or goal `TPD_StorageSetting`, depending on the value in the `SettingType` option. If one already exists, the provider simply returns a reference to that instance. This instance is transient only and does not persist across reboots. The client then performs a `GetInstance` to obtain the instance of the `TPD_StorageSetting` in order to examine the properties. If the values are OK, then it should be used as the Goal parameter for the `CreateOrModifyElementFromStoragePool` method. If not, continue with the next step.
2. If you would like to change the value of some or all of the properties in the `TPD_StorageSetting` instance from step 1, then you should call the `ModifyInstance` method to modify them.

3. Call the `CreateOrModifyElementFromStoragePool` method with the reference to the `TPD_StorageSetting` from step 1 or 2 as the Goal parameter.
   - To create a legacy volume with FC drives, use a reference to the FC `StoragePool` as the `InPool` parameter.
   - To create a legacy volume with SSDs, use reference to SSD `StoragePool` as the `InPool` parameter.
   - To create a legacy volume with NL drives, use a reference to the NL `StoragePool` as the `InPool` parameter.
   - To create a thinly provisioned volume or a fully provisioned volume that allocates from a `DynamicStoragepool`, use reference to `DynamicStoragePool` as the `InPool` parameter. For a TPVV, the `ElementType` parameter should also be set to `ThinlyProvisionedStorageVolume (5).`
   - For a fully provisioned volume, the `ElementType` parameter should be set to `StorageVolume(2).`
   - If the `InPool` parameter is NULL, by default, the FC drives are used to create the volume.
   - The parameter `TheElement` has to be NULL.
4. You can optionally call `DeleteInstance` on the `TPD_StorageSetting` instance from step 1 or 2 to delete it.

Table 35 Parameters for the CreateOrModifyElementFromStoragePool Method

Operations/Parameters	Create Volume	Grow/Set Volume
ElementName	Name of Volume	New name of volume; NULL means no change in volume name.
ElementType	2 (Storage Volume) 5 (Thinly Provisioned Storage Volume)	2 5 (Thinly Provisioned Storage Volume).
Goal	Reference to either fixed `TPD_StorageSetting` or one created through the `CreateSetting` option; defaults to fixed RAID-10 if NULL	Reference to a `TPD_StorageSetting` instance containing new parameter values; NULL means no change in `StorageSetting` parameters.
Size (bytes)	Minimum 268435456 (256 MB)	Must be larger than the current volume size; NULL means the volume is not grown.
InPool	Reference to `all-FC`, `all-NL`, `all-SSD` pool or `DynamicStoragePool`; defaults to `all-FC` pool if NULL	NULL.
TheElement	NULL	Reference to an existing TPD_StorageVolume.

Creating a Volume with Disk Pattern Filtering

Disk pattern filtering can be achieved using the 3PAR extension properties to the `TPD_StorageSetting` class (refer to the following table). Each of these properties are defined as an array, with each array set representing a set of patterns, so for example if a client wants to provide for two sets of patterns, the array size of these properties must all be 2. Properties that are not applicable must be set to either empty string for string type or –1 for integer type.

The format for the string is the same as that for the CLI as described in the Table 36 (page 35):

Table 36 Disk Pattern CIM Properties

Disk Pattern CIM Property Name (from `TPD_StorageSetting` class)	CIM Property Type	CLI Equivalent	Format
DiskPrimPathNodes	String array	-nd	<item>
DiskPrimPathPCISlots	String array	-st	<item>
DiskPrimPathPorts	String array	-pt	<item>
DiskCages	String array	-cg	<item>
DiskMagazines	String array	-mg	<item>
DiskMagPositions	String array	-pn	<item>
DiskIDs	String array	-dk	<item>
DiskTotalChunkletGT	Signed Integer array	-tc_gt	an integer
DiskTotalChunkletLT	Signed Integer array	-tc_lt	an integer
DiskFreeChunkletGT	Signed Integer array	-fc_gt	an integer
DiskFreeChunkletLT	Signed Integer array	-fc_lt	an integer
DiskPackageModels	String array	-devid	<id>,<id>,..
DiskRPM	Signed Integer array	-rpm	an integer

Where an item is:

`<nb>` – `nb` is an integer.

`<item>,<item>` – an item is a list of items.

`<nb>-<nb>` – an item is a range of integers

Summary of Different Types of Volumes That Can Be Created

Table 37 (page 35) outlines the types of volumes that are created based on the input `InPool` and `Goal` parameter to `CreateOrModifyElementFromStoragePool()` or `InPools` and `Goal` parameter to `TPD_CreateOrModifyElementFromStoragePools()`.

Table 37 Volumes Created Based on Parameters to CreateOrModifyElementFromStoragePool()

Type of Volume Created	CreateOrModifyElementFromStoragePool	TPD_CreateOrModifyElementFromStoragePools
Legacy volume, where user space draws from "3PAR:all-FC" (default), "3PAR:all-NL", or "3PAR:all-SSD" concrete pool.	1. ElementType == NULL or StorageVolume (2). 2. InPool == NULL or concrete pool. 3. Goal == NULL or DSPName property not set. Neither `Goal.SnapVolumeSize` nor `Goal.SnapVolumePercentage` can be set. Goal, if present, contains also all the LD parameters (e.g., set size, row size, etc.).	Not supported.
Legacy volume with static snap space, where user and snap space both draw from the same "3PAR:all-FC" (default), "3PAR:all-NL", or "3PAR:all-SSD" concrete pool.	1. ElementType == NULL or StorageVolume (2). 2. InPool == NULL or concrete pool. 3. Either `Goal.SnapVolumeSize` or `Goal.SnapVolumePercentage` is set (both cannot be set).	Not supported.

Table 37 Volumes Created Based on Parameters to CreateOrModifyElementFromStoragePool() *(continued)*

Type of Volume Created	CreateOrModifyElementFromStoragePool	TPD_CreateOrModifyElementFromStoragePools
	`Goal.DSPName` cannot be set. `Goal`, if present, contains all the LD parameters (e.g., set size, row size, etc.).	
Legacy CPVV, where user space draws from "3PAR:all-FC" (default), "3PAR:all-NL", or "3PAR:all-SSD" pool. Snapshot space draws from the DSP.	1. ElementType == NULL or StorageVolume (2). 2. InPool == NULL or concrete pool. 3. `Goal.DSPName` property set. `Goal.SnapVolumeSize` and `Goal.SnapVolumePercentage` can be set. `Goal` also contains all the LD parameters (e.g., set size, row size, etc.). `Goal.SpaceLimit` and `Goal.SpaceLimitWarningThreshold` correspond to allocation limit and warning respectively of the snap space.	Not supported.
Legacy TPVV, where user and snap space both draw from the same DSP.	Not supported.	Not supported.
Fully provisioned volume, where user space draws from DSP.	1. ElementType = NULL or StorageVolume (2). 2. InPool = DSP 3. `Goal` = NULL or DSPName property not set. If the `Goal` parameter is present, all LD related parameters in `Goal` would be ignored.	1. ElementType = NULL or StorageVolume (2). 2. nPools[0] = DSP, InPools[1] = NULL. 3. Goal can be NULL. If the `Goal` parameter is present, all LD related parameters in `Goal` would be ignored.
Fully-provisioned CPVV, where user space and snap space draw from the same DSP or two different DSPs.	1. ElementType = NULL or `StorageVolume (2).` 2. InPool = DSP. 3. `Goal.DSPName` property set. If the `Goal` parameter is present, all LD related parameters in Goal are ignored. `Goal.SpaceLimit` and `Goal.SpaceLimitWarningThreshold` corresponds to allocation limit and warning respectively of the snap space.	1. ElementType = NULL or StorageVolume (2). 2. InPools[0] = DSP1, InPools[1] = DSP2. 3. `Goal` can be NULL. If it is present, all LD related parameters in `Goal` are ignored. `Goal.SpaceLimit` and `Goal.SpaceLimitWarningThreshold` corresponds to allocation limit and warning respectively of the snap space.

Table 37 Volumes Created Based on Parameters to CreateOrModifyElementFromStoragePool() *(continued)*

Type of Volume Created	CreateOrModifyElementFromStoragePool	TPD_CreateOrModifyElementFromStoragePools
Thinly provisioned volume, where user space draws from DSP.	1. ElementType = `ThinlyProvisionedStorageVolume(5)` 2. InPool = DSP. 3. `.Goal` == NULL or DSPName property not set. If it is present, all LD related parameters in Goal are ignored.	1. ElementType = `ThinlyProvisionedStorageVolume(5)`. 2. InPool = DSP. 3. Goal can be NULL. If it is present, all LD related parameters in Goal are ignored.
Thinly provisioned CPVV, where user space and snap space draw from the same DSP or two different DSPs.	1. ElementType = `ThinlyProvisionedStorageVolume(5)`. 2. InPool = DSP. 3. `Goal.DSPName` property set. If the Goal parameter is present, all LD related parameters in Goal are ignored. `Goal.SpaceLimit` and `Goal.SpaceLimitWarningThreshold` corresponds to allocation limit and warning respectively of the snap space.	1. ElementType = `ThinlyProvisionedStorageVolume(5)`. 2. InPools[0] = DSP1, InPools[1] = DSP2. 3. Goal can be NULL. If it is present, all LD related parameters in Goal would be ignored. `Goal.SpaceLimit` and `Goal.SpaceLimitWarningThreshold` corresponds to allocation limit and warning respectively of the snap space.

NOTE: Consult "Modifiable StorageSetting Properties for StorageVolume" (page 37) for the list of modifiable properties of `StorageSetting`.

Modifiable StorageSetting Properties for StorageVolume

When creating or modifying a thin-provisioned or full-provisioned `StorageVolume`, a client can change the following properties of a `StorageSetting` meant for the user space of the volume to a non-default value. The provider ignores all other properties.

Table 38 Summary of Modifiable StorageSetting Properties for StorageVolume

StorageSetting Property	CLI Equivalence	Comment
Policy	-pol	
GeometrySectorsPerTrack	-spt	
GeometryHeadsPerCylinder	-hpc	
ThinlyProvisioned	-tpvv	
SpaceLimit	-snp_al	"-snp_al" uses percentage, but SpaceLimit is in number of bytes.
SpaceLimitWarningThreshold	-snp_aw	snap space limit warning in percentage of `SpaceLimit`.
UserSpaceLimit	-usr_al	`-usr_al` uses percentage, but `UserSpaceLimit` is in number of bytes; applicable only for TPVV; the provider ignores this for a full-provisioned volume, i.e., if `ThinlyProvisioned` == FALSE.
UserSpaceLimitWarningThreshold	-usr_aw	user space limit warning in percentage of `UserSpaceLimit`, applicable only for TPVV; the provider ignores this for a full-provisioned volume, i.e., if `ThinlyProvisioned` == FALSE.

Table 38 Summary of Modifiable StorageSetting Properties for StorageVolume *(continued)*

StorageSetting Property	CLI Equivalence	Comment
ExpirationTime	-exp	Time when the volume expires. (In CLI, the value is the delta, whereas here it is the actual date and time.)
RetentionTime	-retain	Time when the volume retention ends. (In CLI, the value is the delta, whereas here it is the actual date and time.)

Calculating Capacity for StoragePool and StorageVolume

Calculate capacity for `StoragePool` and `StorageVolume` through one of the following methods.

Primordial Storage Pool

The primordial pool represents the raw capacity of the array. Its `TotalManagedSpace` therefore represents the capacity of all the disk drives in the array, whether admitted or not, minus capacity lost due to chunklet formatting. The `RemainingManagedSpace` option represents the capacity of the disks that are in a New state, namely those that have not been admitted. `StoragePool.SpaceLimit` has the same value as `TotalManagedSpace`.

Concrete Storage Pool

`TPD_AllocatedFromStoragePool.SpaceConsumed` between a primordial and a concrete pool (3PAR:all-FC, 3PAR:all-NL or 3PAR:all-SSD) represents the raw capacity taken by the concrete pool. The capacity used for metadata in the creation of the concrete pool can be calculated by (`TPD_AllocatedFromStoragePool.SpaceConsumed – ConcreteStoragePool.TotalManagedSpace`).

`TotalManagedSpace` for a concrete storage pool represents the raw capacity that can be used to create storage volumes, dynamic storage pool, or delta replica storage pool. `RemainingManagedSpace` for a concrete storage pool represents how much raw capacity is available for creating a new volume, dynamic storage pool, or delta replica storage pool. After a volume, dynamic storage pool or delta replica storage pool is created, the raw size of the new volume or the actual raw size allocated to the dynamic storage pool or delta replica storage pool is subtracted from the `RemainingManagedSpace` of the parent concrete pool. This volume raw size is also represented in `TPD_AllocatedFromStoragePool.SpaceConsumed` between the parent concrete pool and the storage volume, dynamic storage pool, or delta replica storage pool.

`TotalManagedSpace` for a concrete storage pool is equal to `RemainingManagedSpace` for the concrete storage pool value plus `TPD_AllocatedFromStoragePool.SpaceConsumed` option from all the storage volumes, dynamic storage pool, or delta replica storage pool allocated from the pool.

`StoragePool.SpaceLimit` has the same value as `TotalManagedSpace`.

Dynamic Storage Pool (CPG)

`TPD_AllocatedFromStoragePool.SpaceConsumed` between a concrete pool and a dynamic storage pool (DSP) represents the raw capacity taken by the DSP. Capacity used for metadata in the creation of the pool can be calculated by (`TPD_AllocatedFromStoragePool.SpaceConsumed – DynamicStoragePool.TotalManagedSpace`).

`TotalManagedSpace` for a DSP represents the logical capacity that can be used to create storage volumes or delta replica storage pool. `RemainingManagedSpace` for a DSP represents what logical capacity is available for creating a new volume or delta replica storage pool. After a volume or delta replica storage pool is created, the logical size of the new volume or the actual logical size allocated to delta replica storage pool is subtracted from the

RemainingManagedSpace of the parent DSP. This size is also represented in TPD_AllocatedFromStoragePool.SpaceConsumed between the DSP and the storage volume or delta replica storage pool.

For a DynamicStoragePool, the formula used in capacity calculation is different from that of a concrete pool (3PAR:all-FC, 3PAR:all-NL or 3PAR:all-SSD). TotalManagedSpace is the actual allocated size to the pool and is not involved in the calculation anymore. Instead, SpaceLimit is used:

- SpaceLimit - RemainingManagedSpace plus total SpaceConsumed.
- TotalManagedSpace - Actual space allocated.
- SpaceLimit - Virtual size. If the pool is limitless, i.e., -sdgl is not specified, SpaceLimit is 1PB which represents the maximum size the system allows.
- RemainingManagedSpace - SpaceLimit minus the sum of SpaceConsumed.

Legacy Storage Volume

Since a concrete pool (3PAR:all-FC, 3PAR:all-NL, and 3PAR:all-SSD) is not RAID-specific, raw capacity is used in calculating its ManagedSpace as well as SpaceConsumed with its direct parent primordial pool and child pools/volumes. All DeltaReplicaStoragePools that are directly allocated from concrete pools also use raw capacity in calculating TotalManagedSpace and RemaniningManagedSpace.

TPD_AllocatedFromStoragePool.SpaceConsumed between the parent concrete pool and the storage volume represents the raw size of the volume.

The meaning of StorageVolume.NumberOfBlocksis the logical usable volume block size. A new property, NumberOfRawBlocks, is introduced to the StorageVolume class to represent the raw volume block size. Hence, TPD_AllocatedFromStoragePool.SpaceConsumed is equal to TPD_StorageVolume.NumberOfRawBlocks multiplied by TPD_StorageVolume.BlockSize. TPD_StorageVolume.ConsumableBlocks remains the same and is now equal to TPD_StorageVolume.NumberOfBlocks.

Capacity used for metadata in the creation of the storage volume can be calculated by [(TPD_StorageVolume.NumberOfRawBlocks minus TPD_StorageVolume.NumberOfBlocks) multiplied by the TPD_StorageVolume.BlockSize].

Initial State Example

Suppose an array consisting of all Fibre Channel disks has a total disk capacity of 1090060445184 bytes (~1039563 MB) of which 923699114496 bytes (~880908 MB) worth of disks have been admitted. The rest of the disks, totaling 166361330688 bytes, have not been admitted and are therefore in a new state. The admitted disks are used to create a concrete pool (all-FC) consisting of all FC disks.

Pimordial TPD_StoragePool.TotalManagedSpace = 1090060445184 (raw capacity of the array, not including capacity lost due to chunklet formatting).

Primordial TPD_StoragePool.RemainingManagedSpace = 166361330688 (raw capacity of new disks; if there are no new disks this value is 0)

TPD_AllocatedFromStoragePool.SpaceConsumed between primordial and concrete "all-FC" pool = 1090060445184 – 166361330688 = 923699114496 (raw space consumed in the creation of "all-FC" concrete pool)

These values for the primordial pool do change no matter how many volumes are created. RemainingManagedSpace and SpaceConsumed only change if the new disks are admitted.

Before Volume Creation

Now the total raw capacity of the all-FC concrete pool is 813896302592 bytes (776192 MB). The Overhead/Metadata used in the creation of the concrete pool (for example, TOC) can be inferred by the difference between this value and `TPD_AllocatedFromStoragePool.SpaceConsumed` between the primordial and concrete pools. Because at this point there are no volumes in the array, the capacity that is free for creating volume is the same as the total raw capacity of the concrete pool.

Concrete TPD_StoragePool.TotalManagedSpace = 813896302592 (raw capacity of the concrete pool)

Metadata used in creating "all-FC" concrete pool = TPD_AllocatedFromStoragePool.SpaceConsumed – Concrete TPD_StoragePool.TotalManagedSpace = 923699114496 - 813896302592 = 109802811904 (~104716 MB)

Concrete TPD_StoragePool.RemainingManagedSpace = 813896302592 (raw capacity that can be used to create volume)

After Volume Creation

Suppose a RAID-10 volume of logical size 10240 MB is created from all-FC pool. This value is represented in the `TPD_StorageVolume` class as `ConsumableBlocks`. In InForm OS release 2.3.1, the `NumOfBlocks` also represents the logical size of the volume. Because this is a RAID-10 volume, its raw size is actually double that of the logical size, for example, 20480 MB. This value is represented in the `TPD_StorageVolume` class as `NumberOfBlocks` (InForm OS 2.2.4 and prior) and `NumberOfRawBlocks` (InForm OS 2.3.1) the raw capacity consumed from the concrete pool in creating this volume is the same as the raw size of the volume, and this capacity is deducted from the `RemainingManagedSpace` of the concrete pool.

BlockSize = 512

- `TPD_StorageVolume.NumberOfBlocks` = `TPD_StorageVolume.ConsumableBlocks` = (10240 * 1024 * 1024) / BlockSize = 20971520 (logical size of the volume).
- `TPD_StorageVolume.NumberOfRawBlocks` = (20480 * 1024 * 1024) / BlockSize = 41943040 (raw size of the volume).
- `TPD_AllocatedFromStoragePool.SpaceConsumed` between concrete "all-FC" pool and volume = `TPD_StorageVolume.NumberOfRawBlocks` * BlockSize = 21474836480 (raw space consumed from concrete pool in the creation of the volume).
- `Concrete TPD_StoragePool.RemainingManagedSpace` = previous `RemaningManagedSpace value` – volume raw size = 813896302592 – 21474836480 = 792421466112.

The `TotalManagedSpace` for the concrete pool remains unchanged.

Fully Provisioned Storage Volume

For a `DynamicStoragePool`, since it is a RAID-specific storage pool, logical user capacity is used in calculating its `ManagedSpace` as well as `SpaceConsumed` with its child pools/volumes. All `DeltaReplicaStoragePools` that are allocated from a DSP, use logical user capacity in calculating `TotalManagedSpace` and `RemainingManagedSpace`. RAID and other types of overhead are hidden in the `SpaceConsumed` between concrete pool and the DSP.

`TPD_AllocatedFromStoragePool.SpaceConsumed` between the parent dynamic storage pool and the storage volume represents the actual size allocated to the volume, and is equal to `TPD_StorageVolume.NumberOfBlocks` multiplied by `TPD_StorageVolume.BlockSize`. This is also the usable size of the volume in blocks. Raw size of the volume is represented by `TPD_StorageVolume.NumberOfRawBlocks`. Unlike the case for legacy storage volume, `SpaceConsumed` between `DynamicStoragePool` and `StorageVolume` is not the raw size

allocated to the volume, since RAID overhead is now accounted for in DynamicStoragePool. The capacity used for the metadata in the creation of the storage volume can be calculated by:

[(TPD_StorageVolume.NumberOfRawBlocks - TPD_StorageVolume.NumberOfBlocks)* TPD_StorageVolume.BlockSize].

TPD_StorageVolume.ConsumableBlocks is set to be the same as TPD_StorageVolume.NumberOfBlocks.

Example:

- Initial State

A dynamic storage pool has a virtual capacity of 100GB (as set in the –sdgl option). This value is reflected in the TPD_DynamicStoragePool.SpaceLimit property and is what is available for volume allocation. Let us assume that there are no volumes or DeltaReplicaStoragePool allocated from this pool initially. The TotalManagedSpace property reflects the actual capacity allocated to the pool.

- Before Volume Creation
- TPD_DynamicStoragePool.TotalManagedSpace = actual capacity allocated to the pool.
- TPD_DynamicStoragePool.SpaceLimit = 100GB
- TPD_DynamicStoragePool.RemainingManagedSpace = 100GB (capacity that can be used to create volume)
- After Volume Creation

Suppose a RAID-10 volume of logical size 10G is created from the dynamic storage pool. This value is represented in the TPD_StorageVolume class as NumberOfBlocks (ConsumableBlocks also has the same value). Since this is a RAID-10 volume, its raw size is actually double that of the logical size, i.e., 20 GB. This value is represented in the TPD_StorageVolume class as NumberOfRawBlocks. AllocatedFromStoragePool.SpaceConsumed between StorageVolume and the parent DynamicStoragePool is the same as the logical size of the volume; RAID overhead is accounted for in the DynamicStoragePool itself and does not carry over to the child volume.

- BlockSize = 512
- TPD_StorageVolume.NumberOfBlocks = TPD_StorageVolume.ConsumableBlocks = (10240 * 1024 * 1024) / BlockSize = 20971520 (logical block size of the volume)
- TPD_StorageVolume.NumberOfRawBlocks = (20480 * 1024 * 1024) / BlockSize = 41943040 (raw block size of the volume)
- TPD_AllocatedFromStoragePool.SpaceConsumed between dynamic storage pool and volume = TPD_StorageVolume.NumberOfBlocks * BlockSize = 10737418240
- TPD_DynamicStoragePool.RemainingManagedSpace = TPD_DynamicStoragePool.SpaceLimit – volume logical size = 100G – 10G = 90G
- TotalManagedSpace for the dynamic storage pool remains unchanged.

Thinly Provisioned Storage Volume

TPD_AllocatedFromStoragePool.SpaceConsumed between the parent dynamic storage pool and the thinly provisioned storage volume represents the actual size allocated to the volume. This value is also represented in StorageVolume.ProvisionedConsumableBlocks, i.e., SpaceConsumed equals ProvisionedConsumableBlocks.

As is the case with Section (page 40), logical capacity is used in calculating SpaceConsumed with child volumes/delta replica pool.

StorageVolume.NumberOfBlocks is the virtual size of the TPVV.

TPD_StorageVolume.ConsumableBlocks is set to be the same as TPD_StorageVolume.NumberOfBlocks.

Example:

- Initial State

A dynamic storage pool has a virtual capacity of 100GB (as set in the –sdgl option). This value is reflected in TPD_DynamicStoragePool.SpaceLimit property and is what is available for volume allocation. Let us assume that there are no volumes or DeltaReplicaStoragePool allocated from this pool initially. The TotalManagedSpace property reflects the actual capacity allocated to the pool.

- Before Volume Creation
- TPD_DynamicStoragePool.TotalManagedSpace = actual capacity allocated to the pool
- TPD_DynamicStoragePool.SpaceLimit = 100GB
- TPD_DynamicStoragePool.RemainingManagedSpace = 100 GB (capacity that can be used to create volume)
- After Volume Creation

Suppose a RAID-10 volume of logical size 10G is created from the dynamic storage pool. This value is represented in the TPD_StorageVolume class as NumberOfBlocks (ConsumableBlocks also has the same value). Since this is a thinly provisioned volume, what are actually allocated to the volume may not be the full 10G (let us assume it is 4G). The actual allocated value is represented in the TPD_StorageVolume class as ProvisionedConsumableBlocks. AllocatedFromStoragePool.SpaceConsumed between StorageVolume and the parent DynamicStoragePool is the capacity that is actually allocated to the volume, i.e., AllocatedFromStoragePool.SpaceConsumed is equal to StorageVolume.ProvisionedConsumableBlocks.

- BlockSize = 512
- TPD_StorageVolume.NumberOfBlocks = TPD_StorageVolume.ConsumableBlocks = (10240 * 1024 * 1024) / BlockSize = 20971520 (logical block size of the volume)
- TPD_StorageVolume.ProvisionedConsumableBlocks = actual allocated usable capacity to the volume in blocks = 4G / BlockSize = 8388608
- TPD_StorageVolume.NumberOfRawBlocks = actual allocated raw capacity to the volume in blocks = 4 * 2 = 8G / BlockSize = 16777216
- TPD_AllocatedFromStoragePool.SpaceConsumed between dynamic storage pool and volume = TPD_StorageVolume.ProvisionedConsumableBlocks * BlockSize = 4294967296 bytes
- TPD_DynamicStoragePool.RemainingManagedSpace = TPD_DynamicStoragePool.SpaceLimit – capacity actually allocated to the volume = 100G – 4G = 96G
- TotalManagedSpace for the dynamic storage pool remains unchanged.

Table 39 Properties Relevant to TPVV

Properties	Description
Uint64 **NumberOfBlocks**	The logical size of the TPVV, in blocks.
Uint64 **NumberOfRawBlocks**	Total number of logically contiguous blocks including overhead and metadata, of size Block Size, which form this Extent. The total size of the Extent can be calculated

Table 39 Properties Relevant to TPVV *(continued)*

Properties	Description
	by multiplying `BlockSize` by `NumberOfRawBlocks`. If the `BlockSize` is 1, this property is the total size of the Extent.
Uint64 **ConsumableBlocks**	The logical size of the TPVV, in blocks, same as `NumberOfBlocks`.
Uint64 **ProvisionedConsumableBlocks**	The number blocks that are currently allocated and visible to host. For example, a RAID-10 volume with `ConsumableBlocks` = 100 GB, but only 10 GB is actually used, then `ProvisionedConsumableBlocks` is 10 GB. Applicable only to a thin-provisioned volume.
Uint32 **SnapAllocationLimit** (percentage)	The SD space of the volume is prevented from growing beyond the specified percentage of the volume size.
Uint32 **SnapAllocationWarning** (percentage)	Generate a warning alert when SD space of the volume exceeds the specified percentage of the volume size.
Uint32 **UserAllocationLimit** (percentage)	The user space of the volume is prevented from growing beyond the specified percentage of the volume size. Applicable only to a thin-provisioned volume.
Uint32 **UserAllocationWarning** (percentage)	Generate a warning alert when user space of the volume exceeds the specified percentage of the volume size. Applicable only to a thin-provisioned volume.
Boolean **ThinlyProvisioned**	If TRUE, the volume is thinly provisioned.
String **SnapDSPName**	The `DynamicStoragePool` (DSP) the snapshot data (SD) are provisioned from.
String **UserDSPName**	The `DynamicStoragePool` (DSP) the user space is provisioned from.

Indications

Following alert indications relating to thin provisioning are supported:

1. Capacity Warning - This is an alert message indicating that the actual capacity of a volume or pool is nearing a limit (e.g., actual usage of containing pool is nearing SpaceLimit). The related standard message is `"Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity in use nearing available limit"`. The indications that fall into this category include:
   a. TPD_StoragePoolGrowWarningAlert - Indication specifying that a storage pool has reached its allocation warning threshold. The following lists the indication property and the value:
      - OwningEntity: "SNIA"
      - MessageID: "DRM28"
      - MessageArguments: "Pool", <PoolID>
      - AlertType: DeviceAlert(5)
      - PerceivedSeverity: Minor(4)
      - ProbableCause: ThresholdCrossed(52)
      - ProbableCauseDescription: "Pool at low space warning threshold: RemainingManagedSpace/TotalManagedSpace"
   b. TPD_StorageVolumeAllocationWarningAlert - Indication specifying that a TPVV has reached its allocation warning threshold.
      - Same as 1a, above but MessageArguments: "Volume", <Volume DeviceID> and ProbableCauseDescription: "Volume at low space warning threshold: RemainingManagedSpace/TotalManagedSpace"
   c. TPD_DeltaReplicaStoragePoolGrowWarningAlert - Indication specifying that a delta replica storage pool has reached its allocation warning threshold.
      - Same as 1a, above
2. Capacity Critical - This is an alert message indicating that the actual capacity of a volume or pool has reached a limit (e.g., actual usage of containing pool is equal to SpaceLimit). Write commands from hosts to the volume or pool are failing. The related standard message is `"Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity in use exceeded available limit"`. The indications that fall into this category include :
   a. TPD_StoragePoolGrowLimitAlert - Indication specifying that a storage pool has reached its allocation limit.
      - OwningEntity: "SNIA"
      - MessageID: "DRM29"
      - MessageArguments: "Pool", <PoolID>
      - AlertType: DeviceAlert(5)
      - PerceivedSeverity: Major(5)
      - ProbableCause: Out of Memory(33)
      - ProbableCauseDescription: "No remaining space in storage pool"

b. TPD_StoragePoolGrowFailureAlert - Indication specifying that a storage pool has failed to grow.
   - Same as 2a, above

c. TPD_StorageVolumeAllocationFailureAlert - Indication specifying that a storage volume has failed to allocate more space.
   - Same as 2a, above but MessageArguments: "Volume", <Volume DeviceID> and ProbableCauseDescription: "No remaining space in storage volume"

d. TPD_StorageVolumeAllocationLimitAlert - Indication specifying that a storage volume has reached its its allocation limit threshold.
   - Same as 2c, above

e. TPD_DeltaReplicaStoragePoolGrowLimitAlert - Indication specifying that a delta replica storage pool has reached its allocation limit threshold.
   - Same as 2a, above

f. TPD_DeltaReplicaStoragePoolGrowFailureAlert - Indication specifying that a delta replica storage pool has failed to grow.
   - Same as 2a, above

3. Capacity OK - This is an alert message indicating that a previous alert condition regarding capacity of a pool or volume has been cleared. The related standard message is `"Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity condition cleared"`. The indications that fall into this category include:
   a. TPD_StoragePoolCapacityClearAlert - Indication indicating that the previous alert condition regarding a dynamic storage pool has cleared.
      - OwningEntity: "SNIA"
      - MessageID:"DRM30"
      - MessageArguments:"Pool", <Pool ID>

   b. TPD_StorageVolumeCapacityClearAlert - Indication indicating that the previous alert condition regarding a storage volume has cleared.
      - Same as 3a, above but MessageArguments: "Volume", <Volume DeviceID>

   c. TPD_DeltaReplicaStoragePoolCapacityClearAlert - Indication indicating that the previous alert condition regarding a delta replica storage pool has cleared.
      - Same as 3a, above.

Copy Services Subprofile

The subprofile defines a management interface for local snapshot management and clone management. Remote copy is not supported in our implementation of this subprofile.

For detailed information regarding the Copy Services subprofile,refer to SMI-S at www.snia.org.

CIM Classes

The model diagrams and the CIM classes involved in implementing Copy Services. The classes are listed in alphabetical order.

Figure 1 Service and Capabilities Class Diagram

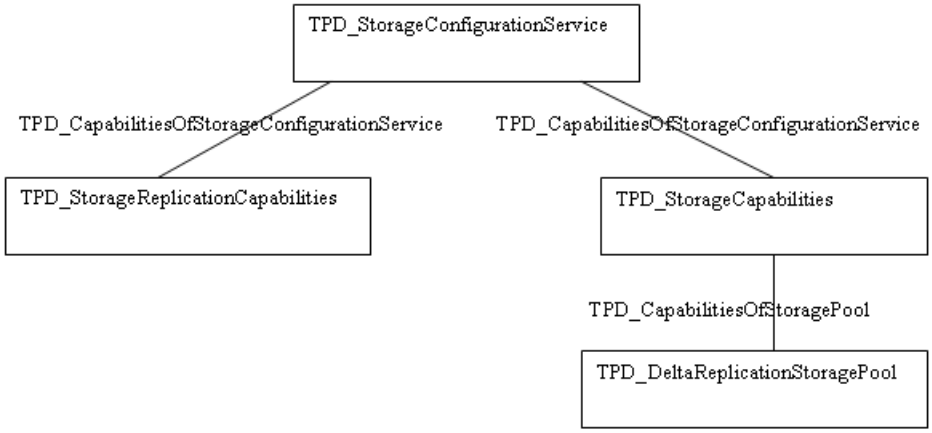


Figure 2 Virtual Copy Class Diagram for a Legacy Volume

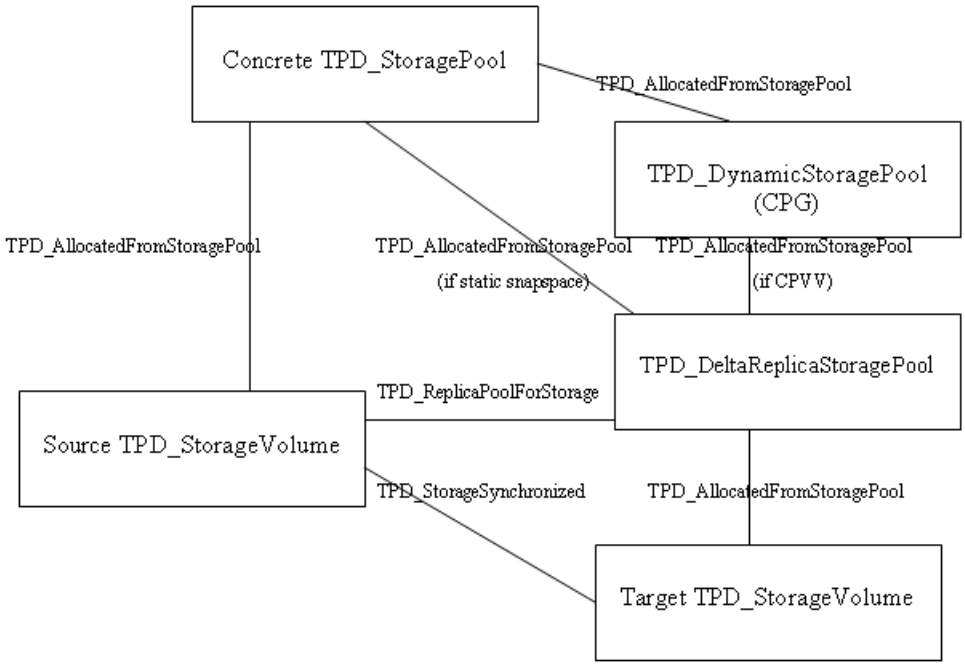
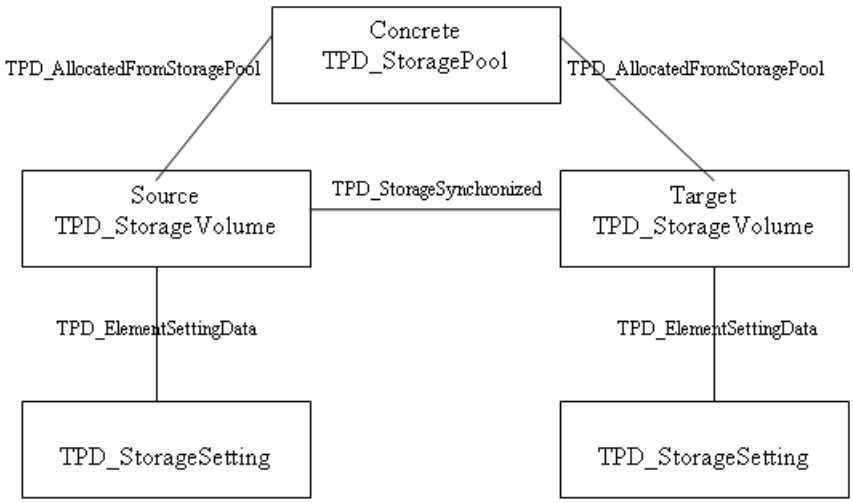
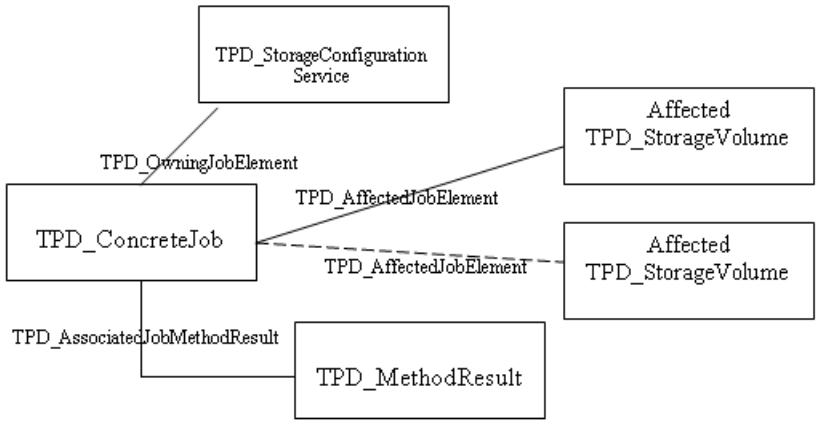


Figure 3 Physical Copy Class Diagram for a Legacy Volume



Note: Physical copy that is associated with the source volume requires a snapshot volume, which is not shown here.

Figure 4 Job Control Class Diagram



TPD_AffectedJobElement

Association between `TPD_ConcreteJob` and the replica `TPD_StorageVolume`. Refer to Section (page 48) for more details.

TPD_AllocatedFromStoragePool

Virtual copy snapshot volume is associated to the `TPD_DeltaReplicaStoragePool` via `TPD_AllocatedFromStoragePool`. `SpaceConsumed` property in this case is the space consumed by the snapshots. For a fully-provisioned volume with static `snapspace`, the `TPD_AllocatedFromStoragePool` association also links the `TPD_DeltaReplicaStoragePool` with the concrete `StoragePool`; whether this is all-FC, all-NL, or all-SSD depends on the device type of the source volume.

For CPVV and TPVV, `TPD_DeltaReplicaStoragePool` is associated to the `DynamicStoragePool` (DSP) from which the SA and SD spaces are allocated via TPD_AllocatedFromStoragePool.

TPD_AssociatedJobMethodResult

Association between `TPD_ConcreteJob` and `TPD_MethodResult`. Refer to Section (page 48) for more details.

TPD_CapabilitiesOfStorageConfigurationService

Association is added between `TPD_StorageConfigurationService` and each instance of TPD_StorageReplicationCapabilities.

TPD_CapabilitiesOfStoragePool

Association between `TPD_DeltaReplicaStoragePool` if it exists and the corresponding TPD_StorageCapabilities.

TPD_ConcreteJob

This is the class that describes the job itself. Refer to Section (page 48) for a more details.

TPD_ConcreteJobInstModification

This is a indication class, sub-classed from `CIM_InstModification`, which indicates completion of a job.

TPD_DeltaReplicaStoragePool

This `StoragePool` class is instantiated upon allocation of SA and SD space of a volume. The SA and SD spaces are modeled as `TPD_DeltaReplicaStoragePool` from which virtual copy snapshots (delta replica) can be allocated and created. This class is a subclass of `CIM_StoragePool`.

`CIM_StoragePool.GetSupportedSizeRange()` method, which is supported in `TPD_StoragePool`, is not supported for `TPD_DeltaReplicaStoragePool`.

TPD_ElementSettingData

Association between `TPD_VolumeSetting` and the replica volume.

TPD_HostedStoragePool

Association is added between `TPD_StorageSystem` and each `TPD_DeltaReplicaStoragePool`.

TPD_MethodResult

This is the class that describes the result of a job. Refer to Section (page 48) for more details.

TPD_OwningJobElement

Association between `TPD_StorageConfigurationService` and `TPD_ConcreteJob`. Refer to section Section (page 48) for more details.

TPD_ReplicaPoolForStorage

Association between the `TPD_DeltaReplicaStoragePool` and the source `StorageVolume` from which the SA and SD spaces are created.

TPD_StorageCapabilities

If `TPD_DeltaReplicaStoragePool` exists, then a `TPD_CapabilitiesOfStoragePool` association is created between it and the `TPD_StorageCapabilities`. For example, if the source volume is of RAID-10, then the `RAID-10 TPD_StorageCapabilities` are associated to the `TPD_DeltaReplicaStoragePool`.

TPD_StorageConfigurationCapabilities

Relevant properties are:

1. `SupportedAsynchronousActions[]` – This property has been deprecated and replaced by `StorageReplicationCapabilities.SupportedAsynchronousActions[]`, but SMI-S decrees that a provider must still specify the values here for backward compatibility with 1.0 clients. 1.1 clients ignore these values and consult those from `TPD_StorageReplicationCapabilities` instead. The following are added to the array:
   a. Replica Attachment – create a physical copy; corresponds to "Local Replica Attachment" in `StorageReplicationCapabilities.SupportedAsynchronousActions`.
   b. Replica Modification – resync a physical copy; corresponds to "Local Replica Modification" in `StorageReplicationCapabilities.SupportedAsynchronousActions`.
2. `SupportedSynchronousActions[]` – This property has been deprecated and replaced by `StorageReplicationCapabilities.SupportedSynchronousActions []`, but new values are added to the array for backward compatibility with 1.0 clients:
   - Replica Creation – create a virtual copy; corresponds to "Local Replica Creation" in `StorageReplicationCapabilities.SupportedSynchronousActions`.
3. `SupportedCopyTypes[]` - describes the replication capabilities supported by the associated `StorageConfigurationServices`. This array contains the following 2 values:
   a. `UnSyncAssoc` – corresponds to virtual copy or physical copy with resync capability
   b. `UnSyncUnAssoc` – corresponds to creating an independent physical copy
4. SupportedStorageElementUsage[] - indicates the intended usage or any restrictions that may have been imposed on the usage of StorageVolumes. This array contains the following values:
   a. Unrestricted - applies to base volume
   b. Reserved for Computer System - applies to admin volume
   c. Reserved by Replication Services - applies to temporary and resync snapshot volume
   d. Delta Replica Target - applies to snapshot volume
5. SupportedStoragePoolUsage[] - indicates the intended usage or any restrictions that may have been imposed on the usage of storage pool. This array contains the following values:
   a. Unrestricted - applies to concrete StoragePool
   b. Reserved for Computer System - applies to primordial StoragePool
   c. Reserved as a Delta Replica Container - applies to DeltaReplicaStoragePool

TPD_StorageConfigurationService

The following new SMI-S methods are implemented:

1. `CreateReplica()` – used for creating virtual copy
2. `AttachOrModifyReplica()` – used for creating physical copy to an existing volume
3. `ModifySynchronization()` – used for promotion of physical and virtual copies, resynchronization of physical copy and halting physical copy or snapshot promotion. The input parameter Operation specifies the kind of operation to perform. Valid values for Operation are:
   a. Resync – Causes a snapshot to be restarted as a new PIT (Point-in-Time) image (resync physical copy)
   b. Restore – Copies a snapshot to the source element (promote virtual copy)
   c. Detach – Removes the association between the source and replica elements (promote physical copy)
   d. Start Copy – Starts a background copy operation for a replica (start copy on an existing physical copy)
   e. Stop Copy – Stops a background copy operation (halting physical copy)

   In addition, a new method is defined that mirrors the CLI `updatesnapspace` command:
4. `UpdateDeltaSnapshotSpace()` – used to start calculation of SA/SD space used by a virtual copy snapshot volume.

TPD_StorageReplicationCapabilities

This describes the replication capability of the CIM provider. There are three fixed instances of this class, one for each type of `SupportedSynchronizationType` that we support. Some of the major properties in this class are:

1. `SupportedSynchronizationType` – supported values are:
   a. `UnSyncAssoc-Full` - create a full size snapshot (physical copy with 'Save snapshot for later resync' option turned on).
   b. `UnSyncAssoc-Delta` - create a delta snapshot (virtual copy).
   c. `UnSyncUnAssoc` - create a full size, independent replica (physical copy with 'Save snapshot for later resync' option turned off).
2. `SupportedAsynchronousActions[]` - Enumeration indicating what operations are executed as asynchronous jobs, supported values are:
   a. `Local Replica Attachment` – physical copy creation is asynchronous (Attachment means attaching a replica to an existing volume).
   b. `Local Replica Modification` – resync physical copy is also asynchronous.
3. `SupportedSynchronousActions[]` - Enumeration indicating what operations are executed the without creation of a job, supported values are:
   - `Local Replica Creation` – virtual copy creation is synchronous.
4. `SupportedSpecializedElements[]` - Enumeration indicating which specialized storage element types are supported by this instance of StorageReplicationCapabilities. Example values are "Delta Pool", "Delta Snapshot", "Full Snapshots". This affects the value of `IntendedUsage` property of `StorageSetting`, e.g., if "`Delta Snapshot`" is defined here, then this indicates to a CIM client that a `StorageSetting` can be created with `IntendedUsage` set to "`Delta Snapshot`", which limits the `StorageSetting` to only delta snapshot creation. However, for the Inform OS, `StorageSetting` cannot be specified for the replica volume during a physical or virtual copy creation, which means only a value of "Not Specialized" (0) is supported for `StorageSetting.IntendedUsage`. In other words, we cannot specialize a `StorageSetting` to only be used as a goal for creating replica volumes since that is not supported (`StorageSetting` can only be used as a goal in a base volume creation). Thus, we cannot specify "`Full Snapshot`" nor "`Delta Snapshot`" here. Value can only contain the following:
   - `Delta Pool` – `StoragePool` corresponding to the SA/SD spaces of a volume, valid only for virtual copy. This indicates to the CIM client that a specialized pool exists from which to allocate a delta snapshot replica. Since our provider does not support StoragePool creation, a client cannot set `StorageSetting.IntendedUsage` to "`Delta Pool`" either.
5. `InitialReplicationState` – initial state of the replica after it is first created, supported values are:
   a. `Initialized` – Resynchronization or copy task is queued but the copy engine has not started. This is the initial state for a physical copy.
   b. `Idle` - an `UnSyncAssoc` replica is ready to be managed. This is the initial state for a virtual copy.
6. `SupportedModifyOperations[]` - Enumeration indicating which `TPD_StorageConfigurationService.ModifySynchronization()` Operation are supported by this instance of `StorageReplicationCapabilities`. Valid values are outlined in section, `ModifySynchronization()`.

7. `ReplicaHostAccessibility` - Indicates host access restrictions for replicas with these capabilities. Valid values are:
   a. `Not accessible` – this applies to physical copy volumes, which cannot be exported. If a physical copy is promoted to a base volume, then this restriction is lifted.
   b. `No restrictions` – any host may access; this applies to virtual copy volumes.
8. `HostAccessibleState[]` - lists the replica synchronization states in which the provider allows host access to replicas. For physical copy, this is an empty array, as hosts are never allowed to access it. For virtual copy volumes, there are no restrictions whatsoever, so this array contains all the valid states that a virtual copy can have:
   a. `Restore In Progress` – a virtual copy promotion is ongoing.
   b. `Idle` – normal state.
9. `PersistentReplicasSupported` - True indicates replicas can persist during power off or system reset.
10. `MaximumReplicasPerSource` – Maximum number of replicas that can be associated with one source element. If the replica is physical copy, then the max number of replica per source volume is the max number of volume allowed for a storage system minus 1. If the replica is a virtual copy, then the max number of replica per source volume is the maximum read-write snapshot per source volume. RO snapshots are not counted.
11. `MaximumReplicasPerReadWriteSource` - Maximum number of replicas that can be associated with one read-write source element. This only applies to base volume and read-write snapshots. For physical copy, this is the max number of volume allowed for a storage system minus 1. For virtual copy, this is the maximum read-write snapshot per source volume. RO snapshots are not counted.
12. `MaximumReplicasPerReadOnlySource` - Maximum number of replicas that can be associated with one read-only source element. This only applies to read-only snapshots. The resulting replica can only be read-write. The max number of replica per source volume is the maximum read-write snapshot per source volume. For physical copy this is 0.
13. `MaximumLocalReplicationDepth` - Maximum local mirror replication depth allowed by this instance of `StorageReplicationCapabilities`, including the parent volume. For asynchronous physical copy, this is the maximum number of volumes that can be created on a storage system minus 1. For independent physical copy, this is 0. For virtual copy, this is the maximum read-write snapshot per source volume. RO snapshots are not counted.

14. `DeltaReplicaPoolAccess` - Indicates that a specialized pool is required as a container for delta replica elements. In our case, delta replica (virtual copy snapshots) always requires a specialized pool, the `TPD_DeltaReplicaStoragePool`, which corresponds to the SA/SD space of the base volume. Possible values are:
    a. `Any` – specialized pool not required for delta replicas. This is not the case for our snapshots.
    b. `Shared` – a single shared pool is required for all delta replicas. This is not the case for our snapshots.
    c. `Exclusive` - one specialized, exclusive pool must be created for each source element that has associated delta replicas. This is the only supported value for a virtual copy.

Table 40 Matrix of StorageReplicationCapabilities Properties

Property/Synch Type	UnSyncAssoc-Full (capability to create a physical copy)	UnSyncAssoc-Delta (capability to create a virtual copy)	UnSyncUnAssoc (capability to create an independent clone)
SupportedSynchronizationType	UnSyncAssoc-Full	UnSyncAssoc-Delta	UnSyncUnAssoc
SupportedAsynchronousActions[]	Local Replica Attachment, Local Replica Modification	NULL	Local Replica Attachment
SupportedSynchronousActions[]	NULL	Local Replica Creation	NULL
SupportedSpecializedElements[]	NULL	Delta Pool	NULL
InitialReplicationState	Initialized	Idle	Initialized
SupportedModifyOperations[]	Resync, Detach, Start Copy, Stop Copy	Restore	NULL
ReplicaHostAccessibility	Not accessible	No restrictions	No restrictions
HostAccessibleState[]	NULL	Restore In Progress, Idle	NULL
PersistentReplicasSupported	TRUE		
MaximumReplicasPerSource	Max # of volumes - 1	Max # of snapshots	Max # of volumes - 1
MaximumReplicasPerReadWriteSource	Max # of volumes - 1	Max # of snapshots	Max # of volumes - 1
MaximumReplicasPerReadOnlySource	0	Max # of RW snapshots	0
MaximumLocalReplicationDepth	Max # of volumes - 1	Max # of RW snapshots +1	0
DeltaReplicaPoolAccess	NULL	Exclusive	NULL

TPD_StorageSetting

The provider screens the following properties when a CIM client is creating or modifying a `TPD_StorageSetting` object, since only one value is allowed for each. Any other values passed in by the client results in an Invalid Parameter exception. These properties are initialized with the allowed value upon creation.

1. `PersistentReplica` – True indicates the associated replicas persist during power off or system reset. Only TRUE is accepted since we do not support transient replicas. Applicable only for those settings that are associated to replica volumes.
2. `IntendedUsage` – Describes the use of the storage elements associated with this StorageSetting. This can only be set to “Not Specialized” (0) since we do not support specifying a StorageSetting as a goal when creating physical or virtual copy.

TPD_StorageSynchronized

Association class that associates the replica with the source volume. It contains several important properties that describe the relationship between the source volume and the replica.

1. `SyncMaintained` – whether synchronization is maintained; false for both physical and virtual copy.
2. `CopyType` – the kind of association between source volume and replica; values are:
   a. `Sync` – create and maintain a synchronous mirror copy of the source, e.g., synchronized remote copy.
   b. `Async` – create and maintain an async mirror copy of the source e.g., async remote copy.
   c. `UnSyncAssoc` – create and maintain an un-synchronized copy associated to the source, e.g. virtual copy or physical copy with Save snapshot for later resync option turned on.
   d. `UnSyncUnAssoc` – create an un-synchronized clone of the source element and does not maintain the source association after completing the copy operation, e.g., physical copy with 'Save snapshot for later resync' option turned off.
3. `SyncState` – state of the association between source volume and replica, e.g., ResyncInProgress, RestoreInProgress, Idle, CopyInProgress.
4. `SyncType` - Type of association between source and target.
5. `Mode` - Specifies when target elements are updated.
6. `CopyState` - indicates the current state of the association.

7. `ProgressStatus` - Status of association between source and target groups.

Table 41 Alignment of SyncType/Elements Mode and CopyType

Copy Type (Copy Services)	SyncType/Mode (Replication Services)	Notes
UnsyncAssoc	Clone/Synchronous	Sync-ed physical copy
	Snapshot/Synchronous	Snapshot volume (sv)
UnsyncUnAssoc	Clone/Synchronous	Non-sync (independent) physical copy

Table 42 Alignment of SyncState and CopyState/ProgressStatus

Copy Type(Copy Services)	SyncType/Mode (Replication Services)	Notes
Initialized	Initialized/Completed	When a physical copy task is queued but before the actual copy has begun.
ResyncInProgress	Unsynchronized/Synchronizing	Physical or virtual copy is being created, and the write is not completed yet.
Frozen	Synchronized/Completed	Normal state for a target physical copy, when neither copying nor resync-ing is going on
Idle	Inactive/Completed	Normal state for an sv
Broken	Broken/Not applicable	1. Creation of physical or virtual copy has failed 2. Resync or copy of physical copy has failed 3. Promote of sv has failed
CopyInProgress	Unsynchronized/Resyncing	Copy of an existing physical copy is ongoing or resync of a physical copy
RestoreInProgress	Synchronized/Restoring	Promote of an sv is ongoing
Synchronized	Synchronized/Complete	

TPD_SystemVolume

Associations between `TPD_StorageSystem` and `StorageVolumes`, including physical copies and snapshots.

TPD_DynamicStoragePool

`TPD_DeltaReplicaStoragePool` associated to a CPVV is allocated from `DynamicStoragePool`.

SA/SD Space as StoragePool

A prerequisite for creating snapshots is the allocation of SA and SD spaces to the source volume. This is done during volume creation time or during modify volume operation. `TPD_DeltaReplicaStoragePool`, sub classed from `CIM_StoragePool`, represents this space. Snapshot volumes are associated via `TPD_AllocatedFromStoragePool` to the `TPD_DeltaReplicaStoragePool` instead of a concrete `TPD_StoragePool`. There are two ways to create a `DeltaReplicaStoragePool`:

Goal StorageSetting

If the SnapVolumeSize (-szs) or SnapVolumePercentage (-pct) properties are specified when creating a legacy volume, or if DSPName (-cpg) property is specified when creating a non-legacy volume, within the input parameter Goal (a TPD_StorageSetting object) during create or modify volume operation, a TPD_DeltaReplicaStoragePool will be instantiated automatically.

CreateOrModifyStoragePool

DeltaReplicaStoragePool can also be created by CreateOrModifyStoragePool method (see Section (page 29) for the method signature). Following table outlines the parameters for this method:

Table 43 Creating StoragePool using CreateOrModifyStoragePool

Type of Pool	CreateOrModifyStoragePool Parameters				
	Goal. StoragePoolInitialUsage	InPools	InExtents	Size	Element Name
DeltaReplicaStoragePool (allocates from DynamicStoragePool)	"Reserved as a Delta Replica Container"(4)	Reference to parent DynamicStoragePool	Reference to the StorageVolume associated with the snapspace	NULL	NULL
DeltaReplicaStoragePool (allocates from concrete pool like "all-FC", aka static snapspace)	"Reserved as a Delta Replica Container" (4)	NULL or reference to concrete pool (must be same as parent pool of volume)	Reference to the StorageVolume associated with the snapspace	Size of snapspace	NULL
DynamicStoragePool	NULL or "Unrestricted" (2)	Reference to Concrete pool	NULL	NULL or size of the pool (sdgl)	name

Note that using this way of creating DeltaReplicaStoragePoolis a two-step process: (1) create the volume, then (2) create the DeltaReplicaStoragePool.

Associations

TPD_ReplicaPoolForStorage

The TPD_DeltaReplicaStoragePool is associated to the source StorageVolume via TPD_ReplicaPoolForStorage.

TPD_AllocatedFromStoragePool

New instances of TPD_AllocatedFromStoragePool are created that associate the TPD_DeltaReplicaStoragePool to:

- the parent StoragePool. For non-CPVV, the parent pool is a concrete pool, and whether this is all-FC, all-NL, or all-SSD depends on the device type of the source volume. For CPVV, the parent pool is a DSP.SpaceConsumed property in either case is the sum of raw snap volume size and raw admin size. SpaceLimit and SpaceLimitThresholdWarning are valid only for CPVV.
- all virtual copy snapshot volumes that are created from the source volume. SpaceConsumed property for each association in this case is the space consumed by the snapshot.

For CPVV, the TPD_AllocatedFromStoragePool association links the TPD_DeltaReplicaStoragePool with the source DSP. For ThinStorageVolume (TPVV), the TPD_AllocatedFromStoragePool association links the TPD_DeltaReplicaStoragePool with the source SD DynamicStoragePool (as opposed to the user DynamicStoragePool).

Capacity Calculations

Non-CPVV

TotalManagedSpace of the TPD_DeltaReplicaStoragePool equals to the raw SD space. Space used by SA is not included in TotalManagedSpace, but is included in the TPD_AllocatedFromStoragePool.SpaceConsumed value, so to deduce the raw SA space, one can deduct Pool.TotalManagedSpace from SpaceConsumed. The UpdateDeltaSnapshotSpace() method should be called to prevent the value of RemainingManagedSpace from getting too stale.

```
Pool.TotalManagedSpace = raw SD size
Pool.RemainingManagedSpace = raw SD size - SUM(space used by snapshots)
```

SpaceConsumed of the TPD_AllocatedFromStoragePool between the concrete pool and TPD_DeltaReplicaStoragePool equals to the sum of the raw admin and raw snap user size. SpaceLimit and SpaceLimitWarningThreshold are both NULL since these properties apply only to CPVV.

```
SpaceConsumed by Replica Pool from Concrete Pool = raw SD size + raw SA size
```

SpaceConsumed of the TPD_AllocatedFromStoragePool between TPD_DeltaReplicaStoragePool and a snapshot volume equals the space consumed by the snapshot. This value should be updated via the UpdateDeltaSnapshotSpace() method to prevent it from getting too stale.

```
SpaceConsumed by Snapshot Vol from Replica Pool = space used by a snapshot
```

CPVV

When a base volume is created with a StorageSetting in which TSPName is specified, the resulting TPD_DeltaReplicaStoragePool is allocated from the DSP. A TPD_AllocatedFromStoragePool association is created between the TPD_DeltaReplicaStoragePool and the parent DSP. TPD_AllocatedFromStoragePool.SpaceLimit and TPD_AllocatedFromStoragePool.SpaceLimitThresholdWarning of this association corresponds to the allocation limit (-al) and allocation warning (-aw) respectively.

```
SpaceLimit = (Source Volume Size * al)/100
SpaceLimitWarningThreshold = aw
```

FYI, SpaceLimit is the consumption limit for the allocated TPD_DeltaReplicaStoragePool from the associated DSP in bytes. SpaceLimitThresholdWarning indicates when a warning indication should be generated based on the total amount of space consumed being greater than or equal to (SpaceLimit*SpaceLimitWarningThreshold)/100. A SpaceLimit value of 0 (default) means that no limit is specified. SpaceLimitThresholdWarning is meaningless if SpaceLimit is 0.

SpaceLimit and SpaceLimitThresholdWarning for the TPD_AllocatedFromStoragePool association between the snapshot volume and the parent TPD_DeltaReplicaStoragePool are not used, and hence is not populated, since the InformOS does not support specification of allocation limit on a per snapshot volume basis.

Calculation for TotalManagedSpace of the TPD_DeltaReplicaStoragePool of a CPVV is the similar to that for a non-CPVV, except that the parent pool is now a DSP and not a concrete pool. Space used by SA is not included in TotalManagedSpace, but is included in the TPD_AllocatedFromStoragePool.SpaceConsumed value, so to deduce the raw SA space, one can deduct Pool.TotalManagedSpace from SpaceConsumed. The UpdateDeltaSnapshotSpace() method should be called to prevent the value of RemainingManagedSpace from getting too stale.

```
Pool.TotalManagedSpace = raw SD size
Pool.RemainingManagedSpace = raw SD size - SUM(space used by snapshots)
```

SpaceConsumed of the TPD_AllocatedFromStoragePool between the parent DSP and TPD_DeltaReplicaStoragePool equals to the sum of the raw admin and raw snap user size.

```
SpaceConsumed by Replica Pool from DSP = raw SD size + raw SA size
```

SpaceConsumed of the TPD_AllocatedFromStoragePool between TPD_DeltaReplicaStoragePool and a snapshot volume equals the space consumed by the snapshot. This value should be updated via the UpdateDeltaSnapshotSpace() method to prevent it from getting too stale.

```
SpaceConsumed by Snapshot Vol from Replica Pool = space used by a snapshot
```

UpdateDeltaSnapshotSpace() Method

It is important to note that calculation of space consumed by a snapshot volume can potentially be a long process, and it is not updated real-time by sysmgr. One has to issue the command "updatesnapspace<snapshot vol name>" from CLI (from which a task is generated) to start the calculation of SA/SD space used by a snapshot. Hence, when a CIM client retrieves the RemainingManagedSpace value for a SA/SD pool or the SpaceConsumed value for a snapshot volume, it can potentially be stale if updatesnapspace has not been done for a while. As a result, a new extrinsic method is defined in TPD_StorageConfigurationService class called UpdateDeltaSnapshotSpace() which mirrors the CLI updatesnapspace command.

A client wishing to obtain a more accurate picture of the aforementioned capacity values must first call this method, monitor the returned job for its completion, then finally get at the properties.

```
uint32 UpdateDeltaSnapshotSpace(
   [In] TPD_StorageVolume ref SnapshotVolume[],
   [Out] TPD_ConcreteJob ref Job);
```

- SnapshotVolume: Array of references to the snapshot volume to update. If NULL, then all snapshots are updated.
- Job: A Job is created as a side-effect of the execution of the method, and a reference to that Job is returned through this parameter.

Table 44 Return Values for UpdateDeltaSnapshotSpace()

ValueMap	Values	Explanation/Notes
4	Failed	
0x1000	Method parameters checked – job started	
32768	Volume does not exist	No such snapshot volume, or the specified volume is not a virtual copy snapshot.

Indications

Job control indications are supported for the job started with the `UpdateDeltaSnapshotSpace()` method. These indications are converted from events generated by the task framework.

1. Modification of Operational Status for a Concrete Job to Complete and OK – update successful.
2. Modification of Operational Status for a Concrete Job to Complete and Error – update failed.
3. Modification of Job Status for Concrete Job – is generated if the job is started. For this, the task framework must be enhanced to support component state change events.

   The type of indication generated is `TPD_ConcreteJobInstModification`, sub-classed from CIM_InstModification.

Physical Copy

StorageSynchronized

A `TPD_StorageSynchronized` association is instantiated the moment a vv copy process is started. This associates the target volume with the source volume. The values of this instance are listed in the following table. For a `CopyType` of `UnSyncUnAssoc`, an instance of `TPD_StorageSynchronized` is maintained while the copy is taking place, but is relinquished once the copy is finished.

Table 45 StorageSynchronized Property Values for a Physical Copy

Property	Value
CopyType	UnsyncAssoc (4) Creates an unsynchronized copy and maintain an association to the source
	UnsyncUnAssoc (5) Creates an unsynchronized copy but do not maintain an association to the source
ReplicaType	Full Copy (2) This indicates that a full copy of the source object is to be generated.
SyncState	Varies; refer to the various “SyncState Transitions” sections below for each type of operation.
SyncMaintained	False Synchronization is not maintained between replica and source volume.
WhenSynced	NULL to start with; non-NULL upon successful completion of a copy; the point in time that the Elements were synchronized.

StorageReplicationCapabilities

The ability to create a physical copy is described by the following 2 instances of `TPD_StorageReplicationCapabilities` (see Section (page 50)):

1. UnSyncAssoc-Full – describes the capability of our provider to create an unsynchronized full-snapshot replica that maintains an association with the source volume.
2. UnSyncUnAssoc - describes the capability of our provider to create an independent clone of a volume.

CREATION

CLI EQUIVALENCE

```
createvvcopy -p <parvol> <destvol>
createvvcopy -p <parvol> -s <destvol>
```

SMI-S METHOD

```
A physical copy can be created by a CIM client using the
TPD_StorageConfigurationService.AttachOrModifyReplica() method.
uint32 AttachOrModifyReplica(
[Out] CIM_ConcreteJob ref Job,
[In, REQ] CIM_ManagedElement ref SourceElement,
[In, REQ] CIM_ManagedElement ref TargetElement,
[In, REQ,
  Values {"Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc"},
  ValueMap {"2", "3", "4", "5"}]
 Uint16 CopyType,
[In, EmbeddedInstance ( "CIM_SettingData" )]
 String Goal,
[In] CIM_NetworkPipe ref ReplicationPipe);
```

- **Job**: A Job is created as a side-effect of the execution of the method, and a reference to that Job is returned through this parameter.
- **SourceElement**: Reference to the base `StorageVolume` to copy from.
- **TargetElement**: Reference to the base `StorageVolume` to copy to.
- **CopyType**: Can only be "`UnsyncAssoc` `(4)` or "`UnsyncUnAssoc` `(5)`
- **UnsyncAssoc**: Creates an unsynchronized physical copy that is associated to the source `StorageVolume`. This creates a snapshot volume as a side-effect. This is equivalent to the CLI command "`createvvcopy -p <parvol> -s <destvol>`".
- **UnsyncUnAssoc**: Creates an unsynchronized physical copy of the source `StorageVolume` and does not maintain the source association after completing the copy operation. This is equivalent to the CLI command "`createvvcopy -p <parvol> <destvol>`", i.e., without the `-s` option.
- **Goal**: The `StorageSetting` properties to be created or modified for the target `StorageVolume`. We does not support changing the volume characteristics during `createvvcopy`, so this can only be NULL. A Non-NULL value is ignored.
- **ReplicationPipe**: The `NetworkPipe` element that scopes the remote mirror pair. This is only for remote copy, which is not supported in Phase I, so this can only be NULL.

SMI-S also specifies that providers have to accept the `AttachReplica()` method, which is defined in an earlier version of the copy services subprofile, for backward compatibility.

AttachReplica is similar to AttachOrModifyReplica with the omission of the Goal and ReplicationPipe parameters.

```
uint32 AttachReplica(
[Out] CIM_ConcreteJob ref Job,
[In, REQ] CIM_ManagedElement ref SourceElement,
[In, REQ] CIM_ManagedElement ref TargetElement,
[In, REQ,
  Values {"Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc"},
  ValueMap {"2", "3", "4", "5"}]
 Uint16 CopyType);
```

Table 46 Return values for AttachOrModifyReplica()/AttachReplica()

ValueMap	Values	Explanation/Notes
0	Success	
4	Failed	For example: the source volume does not have snapspace, target volume is incompatible.
5	Invalid Parameter	For example: the wrong CopyType is specified.
6	In Use	Target volume is already a physical copy of another volume.
0x1000	Method parameters checked – job started	

The method execution, if successful, always returns 0x1000, which means that job has started. A client can either monitor Job.OperationalStatus for job completion or wait for the arrival of modification of OperationalStatus for a ConcreteJob to indicate 'Complete' and 'OK' or 'Complete' and 'Error'.

If the operation is of CopyType UnSyncAssoc, a resync snapshot of the physical copy is created. This snapshot has the same characteristics as a regular snapshot.

STATE TRANSITIONS

Table 47 (page 60) shows the values for SyncState and WhenSynced during various stages of physical copy creation. Internal vvol_t volume state flags (v_overall_state) are mapped to each SyncState.

Table 47 State Transitions for Physical Copy Creation

Operation	SyncState	WhenSynced
Copy task queued but copy engine not yet started	Initialized (2)	NULL
Copy in progress	ResyncInProgress (5) Initial copy in progress	NULL
Copy complete	Frozen (14) All blocks copied from source to an UnSyncAssoc replica and the copy engine is stopped	Time of when copy is completed
Copy Failed	Broken (12) The relationship is non-functional due to errors in the source or the target	NULL

RESYNCHRONIZATION

CLI EQUIVALENCE

```
createvvcopy -r <destvol>
```

SMI-S METHOD

A physical copy can be resynchronized with the source volume by a CIM client using the `TPD_StorageConfigurationService.ModifySynchronization()` method with the Operation parameter set to `Resync(4)`. A target volume that is in a Broken state (copy failed) cannot be resync-ed.

```
uint32 ModifySynchronization(
[In, REQ,
  Values {"Detach", "Fracture", "Resync", "Restore",
     "Prepare", "Unprepare", "Quiesce", "Unquiesce",
     "Reset to Sync", "Reset to Async", "Start Copy",
     "Stop Copy"},
  ValueMap {"2", "3", "4", "5", "6", "7", "8", "9",
     "10", "11", "12", "13"}]
 Uint16 Operation,
[Out]CIM_ConcreteJob ref Job,
[In, REQ] CIM_StorageSynchronized ref Synchronization);
```

- **Operation**: describes the type of modification to be made to the replica:
  - **Detach**: 'Forget' the synchronization between two storage objects. Start to treat the objects independently. This performs the same function as promoting a physical copy.
  - **Fracture**: Suspend the synchronization between two storage objects using Sync or Async replication. Not supported for local copy.
  - **Resync**: Causes a snapshot to be restarted as a new PIT (Point-In-Time) image with a new value assigned to `WhenSynced`. This performs the same function as resynchronizing a physical copy.
  - **Restore**: Copies a snapshot to the source element. This performs the same function as promoting virtual copy.
  - **Prepare**: Get the link ready for a Resync operation to take place. Not supported.
  - **Unprepare**: Clear a prepared state if a Prepare is not to be followed by a Resync operation. Not supported.
  - **Quiesce**: Stops the copy engine for snapshots and the snapshot no longer consume space. Not supported.
  - **Unquiesce**: Take the link out of the quiesce state. Not supported.
  - **Reset to Sync**: Changes the `CopyType` value of a mirror replica from “Async” to “Sync”. Not supported.
  - **Reset to Async**: Changes the `CopyType` value of a mirror replica from “Sync” to “Async”. Not supported.

- **Start Copy**: Initiate a full background copy of the source to the `UnSyncAssoc` replica. Replica enters a Frozen state when the copy operation is completed. This performs the same function as issuing a `createvvcopy` again on an existing physical copy.
- **Stop Copy**: Stop the background copy previously started. This performs the same function as halting a physical copy.

- **Job**: A Job is created as a side-effect of the execution of the method, and a reference to that Job is returned through this parameter.
- **Synchronization**: The reference to the `StorageSynchronized` association describing the storage source/replica relationship.

Table 48 Return Values for the Various ModifySynchronization() Operations

ValueMap	Values	Explanation/Notes
0	Job Completed with No Error	Applicable only for Detach (promote).
4	Failed	For example: the TPD_StorageSynchronized object does not exist.
5	Invalid Parameter	For example: the wrong CopyType is specified.
0x1000	Method parameters checked – job started	Does not apply to Detach.

The method execution, if successful, always returns 0x1000, which means that job has started. A client can either monitor `Job.OperationalStatus` for job completion or wait for the modification of `OperationalStatus` for a `ConcreteJob` to indicate Complete and OK or Complete and Error.

Table 49 (page 62) shows the values for `SyncState` and `WhenSynced` during various stages of physical copy resynchronization.

Table 49 State Transitions for Physical Copy Resynchronization

Operation	SyncState	WhenSynced
Resync in progress	ResyncInProgress (5)	Previous sync time
Resync complete	Frozen (14)	Time of when resync is completed
Resync Failed	Broken (12)	Previous sync time

RESYNCHRONIZATION

CLI EQUIVALENCE

```
promotevvcopy
```

SMI-S METHOD

A physical copy can be promoted with the source volume by a CIM client using the `TPD_StorageConfigurationService.ModifySynchronization()` method with the Operation parameter set to `Detach (2)`. After a successful Detach operation, the `StorageSynchronized` association between the source and target volume is relinquished and the target volume becomes an independent base volume. The Detach operation is synchronous and does not require a job.

Table 50 (page 63) shows the values for `SyncState` and `WhenSynced` during various stages of physical copy promotion.

Table 50 State Transitions for Physical Copy Promotion

Operation	SyncState
Promote in progress	Frozen (14) There are no corresponding SyncState in SMI-S for detaching, so SyncState remains in Frozen until operation is complete
Promote complete	N/A – StorageSynchronized gone
Promote Failed	Broken (12)

START PHYSICAL COPY

CLI EQUIVALENCE

```
createvvcopy -p <parvol> -s <destvol>
```

SMI-S METHOD

Copy can be done on an existing physical copy using the `TPD_StorageConfigurationService.ModifySynchronization()` method with the Operation parameter set to `Start Copy (12)`.

The method execution, if successful, always returns 0x1000, which means that job has started. See "StorageReplicationCapabilities" (page 59) for possible return values.

STATE TRANSITIONS

Table 51 (page 63) shows the values for `SyncState` and `WhenSynced` during various stages of start physical copy.

Table 51 State Transitions for Physical Copy to an Existing Copy

Operation	SyncState	WhenSynced
Copy in progress	CopyInProgress (15)	Previous sync time
Copy complete	Frozen (14)	Time of when copy is completed
Copy Failed	Broken (12)	Previous sync time

HALT PHYSICAL COPY

CLI EQUIVALENCE

```
createvvcopy -halt
```

SMI-S METHOD

Copy can be halted using the `TPD_StorageConfigurationService.ModifySynchronization()` method with the Operation parameter set to `Stop Copy(13)`.

This operation is synchronous and does not require a job. The job that is produced as a side-effect of the physical copy can also be terminated by `TPD_ConcreteJob.RequestStateChange`.

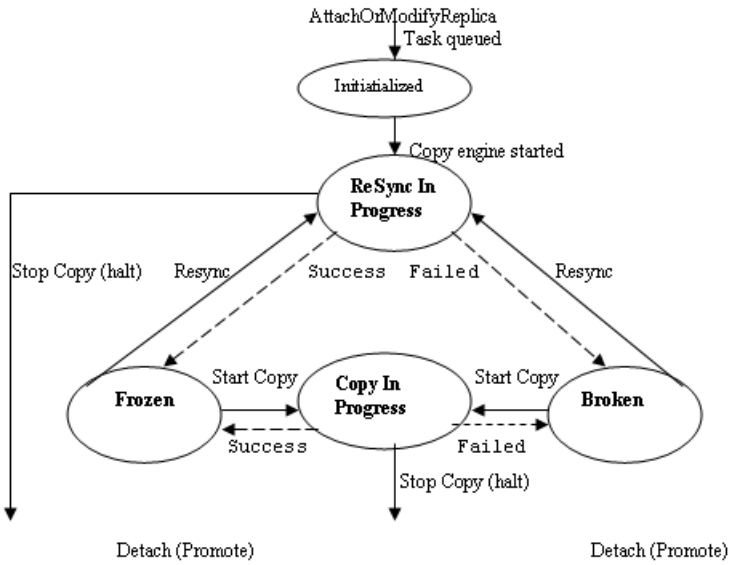
STATE TRANSITIONS

Table 52 (page 64) shows the values for `SyncState` and `WhenSynced` during various stages of halting a physical copy.

Table 52 State Transitions for Halting a Physical Copy

Operation	SyncState	WhenSynced
Copy in progress	ResyncInProgress (5) or CopyInProgress (15)	Previous sync time
Halt Copy Complete	N/A – StorageSynchronized gone	
Halt Copy Failed (This indicates that the halt has failed and the copy was allowed to finish.)	Frozen (14)	New sync time

Figure 5 State Transition Diagram of Physical Copy



Job Control Subprofile Indications

The following job control subprofile indications are supported for the job that is started as a result of the copy operation:

- Modification of Job Status for a Concrete Job - copy/resync started.
- Modification of Operational Status for a Concrete Job to Complete and OK – copy/resync successful.
- Modification of Operational Status for a Concrete Job to Complete and Error – copy/resync failed.

The type of indication generated is `TPD_ConcreteJobInstModification`, sub-classed from `CIM_InstModification`.

Virtual Copy

StorageSynchronized

A `TPD_StorageSynchronized` association is instantiated after the virtual copy volume is created. This associates the target volume with the source volume. A virtual copy volume is also created as a side-product of creating a physical copy that can be resynchronized at a later time. Table 53 (page 64) shows the property values of this instance.

Table 53 StorageSynchronized Property Values for a Virtual Copy

Property	Value
CopyType	UnsyncAssoc (4)

Table 53 StorageSynchronized Property Values for a Virtual Copy *(continued)*

Property	Value
	Creates an unsynchronized copy and maintains an association to the source.
ReplicaType	"After Delta" (4) Indicates that the replica is maintained as delta data from the source object.
SyncState	Varies; refer to the various "SyncState Transitions" sections below for each type of operation.
SyncMaintained	False Synchronization is not maintained between replica and source volume.
WhenSynced	NULL to start with; non-NULL upon successful completion of a copy. The point in time that the Elements were synchronized.

StorageReplicationCapabilities

The ability to create a virtual copy is described by the following instance of `TPD_StorageReplicationCapabilities`:

- UnSyncAssoc-Delta – describes the capability of our provider to create an unsynchronized delta snapshot replica that maintains an association with the source volume

CREATION

CLI EQUIVALENCE

```
createsv
```

SMI-S METHOD

```
A client can create a physical copy from the source volume by using the

TPD_StorageConfigurationService.CreateReplica() method.
uint32 CreateReplica(
[In] string ElementName,
[Out] CIM_ConcreteJob ref Job,
[In, REQ] CIM_LogicalElement ref SourceElement,
[Out] CIM_LogicalElement ref TargetElement,
[In] CIM_StorageSetting ref TargetSettingGoal,
[In] CIM_StoragePool ref TargetPool,
[In, REQ,
  Values {"Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc"},
  ValueMap {"2", "3", "4", "5"}]
 Uint16 CopyType);
```

- **ElementName**: Name of the target snapshot volume.
- **Job**: No job is created. This is NULL on output.
- **SourceElement**: Reference to the base `StorageVolume` to copy from.
- **TargetElement**: Reference to the target snapshot `StorageVolume` that is created.
- **TargetSettingGoal**: The definition for the `StorageSetting` to be maintained by the target storage object. Only the `BaseID` property is applicable.

- **TargetPool**: Reference to the `StoragePool` from which the virtual copy volume is allocated. This can either be NULL or the `TPD_DeltaReplicaStoragePool` associated with the `SourceElement` volume. Any other values results in a Invalid Parameter error.
- **CopyType**: Can only be "`UnsyncAssoc`" `(4)`
  -
  - **UnsyncAssoc**: Creates an unsynchronized virtual copy that is associated to the source `StorageVolume`.

Table 54 Return values for CreateReplica() Operation

ValueMap	Values	Explanation/Notes
0	Job Completed with No Error	
4	Failed	
5	Invalid Parameter	For example: the wrong CopyType is specified

STATE TRANSITIONS

Table 55 (page 66) shows the values for `SyncState` and `WhenSynced` during various stages of the virtual copy creation.

Table 55 State Transitions for Virtual Copy Creation

Operation	SyncState	WhenSynced
Copy in progress	ResyncInProgress (5)	NULL
Copy complete	Idle (11)	Time when copy is completed.
Copy Failed	Broken (12)	Previous sync time.

PROMOTION

CLI EQUIVALENCE

```
promotesv
```

SMI-S METHOD

Virtual copy promote can be done using the `TPD_StorageConfigurationService.ModifySynchronization()` method with the Operation parameter set to Restore (5).

The method execution, if successful, always returns 0x1000, which means that job has started.

STATE TRANSITIONS

Here are the values for `SyncState` and `WhenSynced` during various stages of promoting virtual copy.

Table 56 State Transitions for Virtual Copy Promote

Operation	SyncState	WhenSynced
Promote in progress	RestoreInProgress (10)	Previous sync time
Promote complete	Idle (11)	Time of when copy is completed
Promote Failed	Broken (12)	Previous sync time

HALT SNAPSHOT PROMOTION

CLI EQUIVALENCE

```
promotesv -halt
```

SMI-S METHOD

Virtual copy promote can be halted via the Job Control method `TPD_ConcreteJob.RequestStateChange()` method with the `RequestedState` parameter set to `Terminate(4)`.

STATE TRANSITIONS

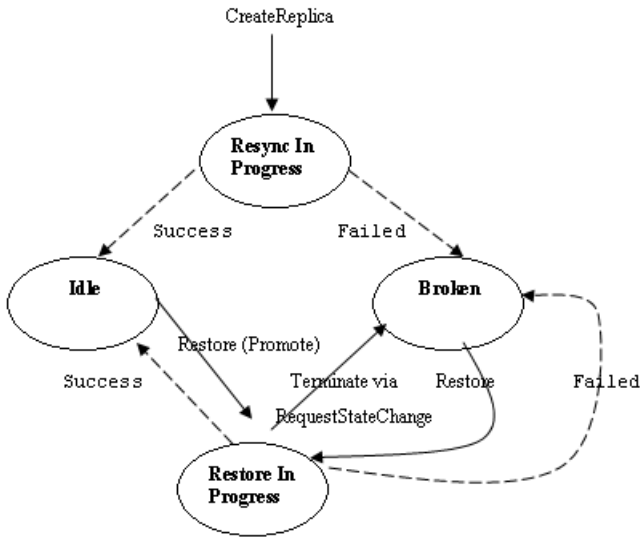
Here are the values for `SyncState` and `WhenSynced` during various stages of halting virtual copy promote.

Table 57 State Transitions for Halting Virtual Copy Promote

Operation	SyncState	WhenSynced
Promote in progress	RestoreInProgress (10)	Previous sync time
Halt complete	Broken (12)	Previous sync time
Promote Failed Promote Failed (This indicates that the halt has failed and the promote was allowed to finish).	Idle (11)	New sync time

STATE TRANSITIONS

Figure 6 State Transition Diagram of Virtual Copy



Job Control Indications

The following job control subprofile indications are supported for the job that is started as a result of promote operation:

- Modification of Job Status for a Concrete Job – promote sv/update snapspace started.
- Modification of Operational Status for a Concrete Job to Complete and OK – promote sv/update snapshot space successful.
- Modification of Operational Status for a Concrete Job to Complete and Error – promote sv/update snapshot space failed.

The type of indication generated is `TPD_ConcreteJobInstModification`, sub-classed from `CIM_InstModification`.

Job Control

Overview

Only the following tasks are supported:

- Create physical copy
- Resync physical copy
- Promote virtual copy
- Update snapshot space

TPD_ConcreteJob

This is a new class, and each instance of this class represents a task.

RequestStateChange() Method

This method can be used to cancel a task. The input parameter `RequestedState` can only be set to `Terminate(4)`, since the InformOS task framework only supports task canceling.

```
uint32 RequestStateChange(
[In, REQ,
  Values {"Start", "Suspend", "Terminate", "Kill", "Service"},
  ValueMap {"2", "3", "4", "5", "6"}]
 Uint16 RequestedState,
[In] dateTime TimeoutPeriod);
```

- **RequestedState**: Changes the state of a job.
  -
  - **Start**: changes the state to running; not supported.
  - **Suspend**: stops the job temporarily which can be resumed; not supported.
  - **Terminate**: stops the job cleanly and orderly; this is the only supported value.
  - **Kill**: terminate the job immediately; not supported.
  - **Service**: puts the job into a service state; not supported.
- **TimeoutPeriod**: A timeout period that specifies the maximum amount of time that the client expects the transition to the new state to take. This is not supported and must be NULL.

Table 58 Return Values for RequestStateChange() Operation

ValueMap	Values	Explanation/Notes
1	Not supported	RequestedState not supported.
4	Failed	
5	Invalid Parameter	
4096	Method Parameters Checked - Transition Started	This is the normal successful return case, as a task may not be terminated immediately.
4098	Use of Timeout Parameter Not Supported	If TimeoutPeriod parameter is not NULL.

GetError() Method

When the job is executing or has terminated without error, then this method returns no CIM_Error instance. However, if the job has failed because of some internal error, then a CIM_Error instance is returned. This method is only used to fetch the failure reason for jobs that failed on their own, i.e., not jobs that are canceled by the user.

```
uint32 GetError(
[Out,
  EmbeddedInstance ("CIM_Error"}]
 String Error);
```

- **Error**: If the `OperationalStatus` on the Job is not “OK”, then this method returns a CIM Error instance. Otherwise, when the Job is "OK", null is returned.

Table 59 Return Values for GetError()Operation

ValueMap	Values	Explanation/Notes
0	Success	
5	Invalid Parameter	

Mapping Error Code to CIM_Error

A CIM_Error object consists of the following properties:

- **OwningEntity** – the string “SNIA”
- **MessageID** –DRM26, Resource not available. Consult SMI-S section on “Standard Messages” for detailed information on the meaning of the DRM messages.
- **MessageArguments[]** – An array containing the dynamic content of the message. There is only one element, “Operation has failed”.
- **Message** – The formatted message. In the case of DRM26, MessageArguments and Message are the same, “Operation has failed”.
- **PerceivedSeverity** – describes the severity of the error from the notifier's point of view. This value is Low (2).
- **ProbableCause** - describes the probable cause of the error. This value is “Configuration/Customization Error” (8).
- **CIMStatusCode** - The CIM status code that characterizes this instance. This value is ERR_ACCESS_DENIED (2).
- **CIMStatusCodeDescription** – “CIM_ERR_ACCESS_DENIED”

DELETING JOB

CLI EQUIVALENCE

```
removetask
```

DELETEINSTANCE

An instance of `TPD_ConcreteJob` can be deleted using the intrinsic method `TPD_ConcreteJob.DeleteInstance()`. If not deleted, a job exists indefinitely.

TPD_METHODRESULT

This class is used to report the extrinsic method that triggered the job and the parameters passed to it, so that a third party client can tell what the job is and what it is doing. The class consists of the following mandatory properties:

- **InstanceID** – this is a key and is identical to that of `ConcreteJob` this is associated with.
- **PrecallIndication** – string representing an embedded instance of the `CIM_InstMethodCall` class, which contains the method called that triggered this job, (e.g., if this is a create vv copy job, then this contains a `CIM_InstMethodCall` instance representing `AttachOrModifyReplica`. `CIM_InstMethodCall` contains the following values:
  -
  - **MethodName** – string of the `methodName`, e.g., `AttachOrModifyReplica`.
  - **MethodParameters** - parameters of the method, formatted as an `EmbeddedObject` (with a predefined class name of "`__MethodParameters`"). This only contains the input parameters; output parameter (Job) is not included.
  - **PreCall** - Boolean indicating whether the Indication is sent before the method begins executing (TRUE) or when the method completes (FALSE). In this case this value is TRUE.
- **PostCallindication** – string representing an embedded instance of the `CIM_InstMethodCall` class, which contains the method called that triggered. However, this differs from the case in `PreCallIndication` in that output parameters is also included in the `MethodParameters` parameter. Output parameter in this case is Job.
  -
  - **MethodName** – string of the `methodName`, e.g., `AttachOrModifyReplica`.
  - **MethodParameters** - parameters of the method, formatted as an `EmbeddedObject` (with a predefined class name of "`__MethodParameters`"). This contains both input and output parameters. Job parameter is a reference to the `TPD_ConcreteJob` that was created.
  - **PreCall** – FALSE
  - **ReturnValue** - contains string representation of the method's return value, e.g., "Failed"
  - **ReturnValueType** - the type of the method return value. This is always "uint32" (9) since all replication methods that we support have a return value type of uint32.

TPD_OwningJobElement

Association between `TPD_StorageConfigurationService` and `TPD_ConcreteJob`.

TPD_AffectedJobElement

Association between `TPD_ConcreteJob` and one or more `TPD_StorageVolume` affected by the job. The affected volumes are the source and target volumes, with the exception of update snapspace task, where the affected volumes are those snapshots that are being updated.

TPD_AssociatedJobMethodResult

Association between `TPD_ConcreteJob` and `TPD_MethodResult`.

Indications

The following Job Control subprofile indication is supported:

```
"SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob".
```

Location Subprofile

The Location subprofile models information about the location of a physical device. In the case of the Array profile, it is used to provide information about the physical location of a storage array. The information provided through this subprofile is the physical location of the HP 3PAR Storage System.

For detailed information regarding the Location subprofile, refer to SMI-S at http:// www.snia.org.

Location Subprofile CIM Classes

Table 60 Location Subprofile CIM Classes

Class	Description
TPD_SystemPackage	The Physical Element whose Location is specified. This represents the physical aspects of the HP 3PAR Storage System storage array.
TPD_SystemLocation	The Location of the specified Physical Element.

Supported Methods

Table 61 Methods for TPD_SystemLocation

Method	Description
createInstance()	Creates an instance of `TPD_SystemLocation`.
deleteInstance()	Deletes an instance of `TPD_SystemLocation`.

Multiple Computer System Subprofile

The Multiple Computer System subprofile models the underlying systems or controllers that cooperate to present a single top-level system (such as a Cluster) that provides added redundancy or increased functionality. This subprofile provides information about the Controller Nodes that comprise the HP 3PAR Storage System cluster. The cluster is modeled as multiple tiers with each tier representing a pair of Controller Nodes.

For detailed information regarding the FC Target Ports subprofile, refer to SMI-S at http:// www.snia.org.

Multiple Computer System Subprofile CIM Classes

Table 62 Multiple Computer System Subprofile CIM Classes

Class	Description
TPD_StorageSystem	Represents the storage array as a whole.
TPD_NodeSystem	Represents a Controller Node on an HP 3PAR Storage System.
TPD_NodePairSystem	Represents a pair of Controller Nodes on an HP 3PAR Storage System.
TPD_NodePairRedundancySet	Members of this set are node pairs that make up the HP 3PAR Storage System cluster.
TPD_RedundancySet	Members of this set are Controller Nodes that make up a node pair within an HP 3PAR Storage System.

Supported Methods

None.

Software Subprofile

The Software subprofile models information on installed controller software, including version information, build numbers, and manufacturer identification. The information provided through this subprofile is specific for the version of the InForm OS installed on the `TPD_ComputerSystem` that represents the HP 3PAR Storage System.

For detailed information regarding the Software subprofile, refer to SMI-S at http:// www.snia.org.

Software Subprofile CIM Classes

Table 63 Software Subprofile CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the software is installed.
TPD_SoftwareIdentity	Information on installed software on the specified `TPD_ComputerSystem`.

Supported Methods

Table 64 Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

Masking and Mapping Subprofile

The Masking and Mapping subprofile is used to determine which LUNs are visible to which initiators through which target ports. The effect is that a given exported volume is only visible to specific hosts through specific target ports. Per SMI-S, the ability to limit access is defined as Device Masking; the ability to specify the device address seen by particular initiators is defined as Device Mapping. The subprofile also specifies methods to allow administrators to actively manage (export and delete) LUNs. The CIM Server supports the Masking and Mapping subprofile and allows for both the display and active management of LUNs.

For detailed information regarding the Masking and Mapping subprofile, refer to SMI-S at http:// www.snia.org.

Masking and Mapping Subprofile CIM Classes

Table 65 Masking and Mapping Subprofile CIM Classes

Class	Description
TPD_AuthorizedPrivilege	Represents the access permission that a `StorageHardwareID` is allowed on a `SCSIProtocolController`.
TPD_StorageSystem	The storage array.
TPD_SCSIController	Represents the SCSI view of ports as seen by SCSI initiators.
TPD_iSCSIController	Represents the iSCSI view of ports as seen by iSCSI initiators.
TPD_ControllerConfigurationService	Provides methods that allow a client to export and delete LUNs on an HP 3PAR Storage System.
TPD_StorageHardwareID	An initiator port identifier, i.e., a host path.
TPD_StorageHardwareIDCollection	A collection of related host paths. This is equivalent to a Host as seen on an HP 3PAR Storage System.

Table 65 Masking and Mapping Subprofile CIM Classes *(continued)*

Class	Description
CIM_StorageHardwareIDManagementService	Provides methods that allow a client to create, add/or delete hosts or host paths.
TPD_ProtocolControllerMaskingCapabilities	Defines the masking related capabilities of the HP 3PAR Storage System.
TPD_StorageVolume	The storage volume to be exported.

Supported Methods

Table 66 (page 73) - Table 68 (page 73) shows the supported methods of the masking and mapping subprofile:

Table 66 Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

Table 67 Methods for TPD_HardwareIDManagementService

Method	Description
CreateStorageHardwareID()	Expose SCSI Logical units.
DeleteStorageHardwareID()	Hide a list of SCSI logical units.
CreateStorageHardwareIDCollection()	Create a collection of `StorageHardwareIDs`.
DeleteStorageHardwareIDCollection()	Delete a collection of `StorageHardwareIDs`.

Table 68 Methods for TPD_ControllerConfigurationService

Method	Description
ExposePaths()	Expose SCSI Logical units.
HidePath()	Hide a list of SCSI logical units.
ExportDefaultLUs()	Expose a list of SCSI logical units (such as RAID volumes) through a "default view" `SCSIProtocolController` (SPC) through a list of target ports. The "default view" SPC exposes logical units to all initiators. This SPC is identified by an association to a `StorageHardwareID` with Name property set to the empty string.
HideDefaultLUs()	Hide a list of SCSI Logical Units (such as RAID volumes) through a default view `SCSIProtocolController` through a list of target ports on a default view `SCSIProtocolController`.
ExposeLUsToStorageHardwareIDCollection()	Expose a list of SCSI logical units to a list of `StorageHardwareIDCollections` (hosts).
HideLUsFromStorageHardwareIDCollection()	Hide a list of SCSI logical units through a list of `StorageHardwareIDCollections` (hosts).

SMI-S View and Paths

Storage arrays usually provide a means for the administrator to specify which initiators can access what volumes via what logical unit number (LUN). The goal is that a given volume is only accessible to SCSI commands originated from the specified initiators through specific sets of target ports. The ability to limit access is called **Device Masking**. The ability to specify the device address (LUN)

seen by particular initiators is called **Device Mapping**. In the HP 3PAR Storage System, storage volumes must be explicitly exported to be visible to hosts or they are not accessible to any host.

A **view** is a list of logical units exposed through a list of target ports, modeled as `SCSIProtocolController` (SPC) with associated `StorageVolume`, `StorageHardwareIDs` (host WWN/iSCSIName), and `SCSIProtocolEndPoints` (port WWNs/iscsiNames). A path is a combination of one each: logical unit (volume), initiator port (host port), and target port. An SPC serves as a collection of paths.

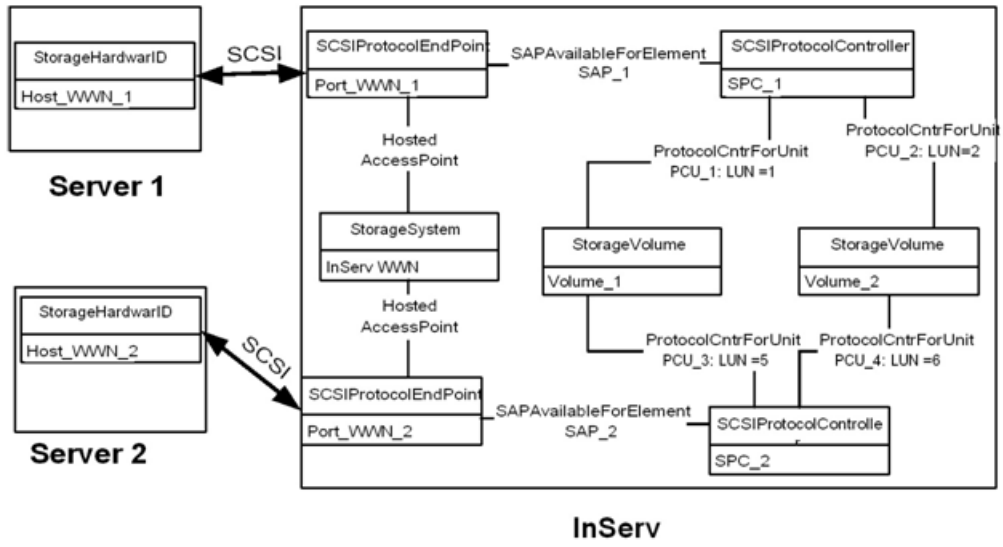
Three kinds of SPCs are supported in the HP 3PAR Storage System environment:

- **Matched-Set SPC**, its deviceID format is `MS:<n:s:p>:<host id>`, where `<n:s:p>` is the notation of the target port and `<host_id>` is the system assigned ID of the host where initiator port(s) reside. For this kind of view, a volume is exposed to the specified host only through the specified target port. The host can contain multiple initiator ports, hence this kind of view can have associations to more than one StorageHardwareIDs. This view can also have associations to multiple StorageVolumes, but only one association to the target port.
  - Supported ExposePaths Use Cases:
    - Create new view
    - Add LU to view
    - Add initiator port to view — allowed only if the initiator port is zoned together with the target port
  - Supported HidePaths Use Cases:
    - Remove view
    - Remove LU from view
    - Remove initiator port from view
- **Host-See SPC**, its deviceID format is `HS:<host id>`, where the `<host_id>` is the system assigned ID of the host where initiator port(s) reside. For this kind of view, a volume is exposed to the specified host regardless of the target port. The host can contain multiple initiator ports, hence this kind of view can have associations to more than one StorageHardwareIDs. This view can also have associations to multiple StorageVolumes and target ports.
  - Supported ExposePaths Use Cases:
    - Create new view
    - Add LU to view
    - Add initiator port to view
  - Supported HidePaths Use Cases:
    - Remove view
    - Remove LU from view
    - Remove initiator port from view
- **Port-Present SPC**, its deviceID format is `PP:<n:s:p>`, where `<n:s:p>` is the notation of the target port. For this kind of view, a volume is exposed through the specified target port regardless of the host the port is connected to or zoned with. The host can contain multiple initiator ports, hence this kind of view can have associations to more than one

StorageHardwareIDs. This view can also have associations to multiple StorageVolumes, but only one association to the target port.

- Supported ExposePaths Use Cases:
  - Create new view
  - Add LU to view
- Supported HidePaths Use Cases:
  - Remove view
  - Remove LU from view

Figure 7 Masking and Mapping Example 1 (Direct Attached Host)



Example 1 shows that there are views from 2 FC hosts with same volumes:

- SPC_1: Volume_1 is exported to Server 1 via Port_WWN_1 (target port on the storage system) as LUN 1 and Volume_2 as LUN 2. The host HBA is represented as Host_WWN_1.
- SPC_2: Volume_2 is exported to Server 2 via Port_WWN_2 (target port on the system) as LUN 5 and Volume_2 as LUN 6. The host HBA is represented as Host_WWN_2. Different LU numbers are used in this example, but they can be assigned the same LUNs as in SPC_1.

Figure 8 Masking and Mapping Example 2 (one storage system port connecting to fabric)

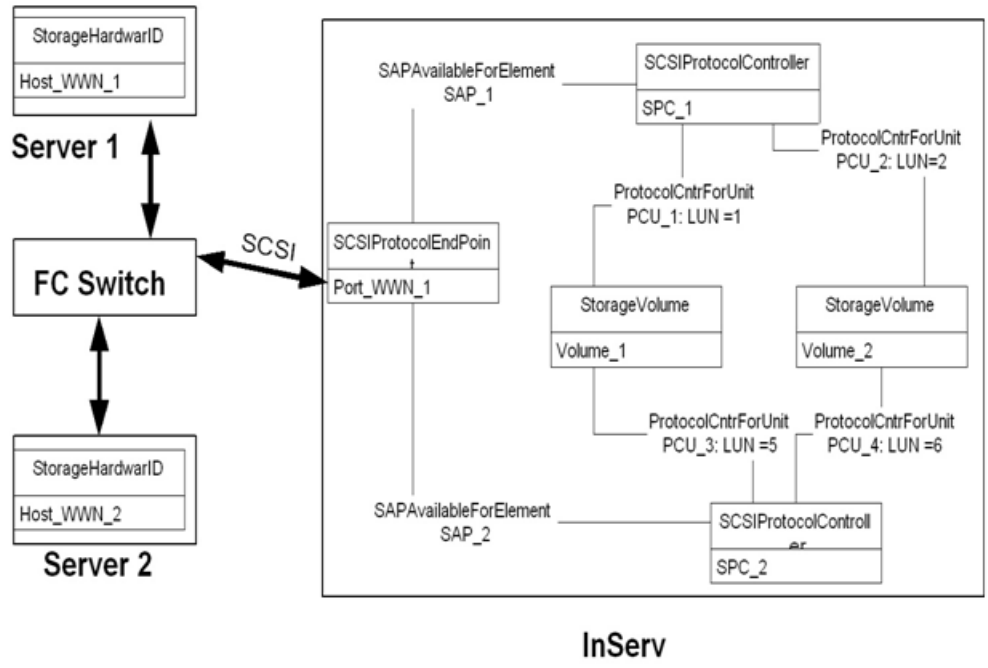
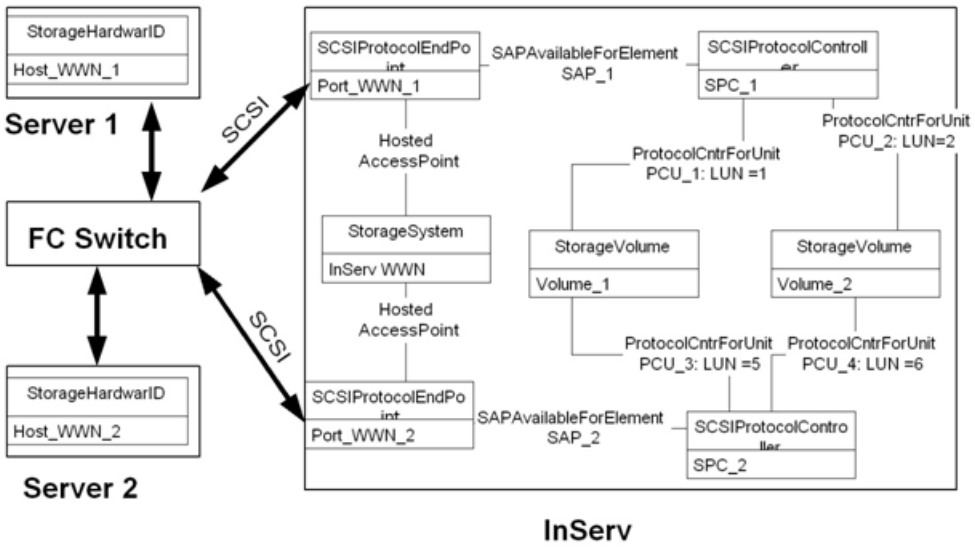


Figure 9 Masking and Mapping Example 3 (two storage system ports connecting to fabric).



ProtocolControllerMaskingCapabilities

This class describes the capabilities of the SPC in a storage system, which describes the kind of controller configuration and `StorageHardwareID` management services that are available to clients. The following settings are fixed and cannot be modified:

- **PortsPerView**: All Ports share the same view (4); this reflects the fact that Host-See SPC is supported which means that TargetPortIDs do not need to be specified when calling ExposePaths.
- **ClientSelectableDeviceNumbers**: true; LUN ID can be selected by a client.
- **AttachDeviceSupported**: false.
- **OneHardwareIDPerView**: false.
- **UniqueUnitNumbersPerPort**: false.

- **PrivilegeDeniedSupported**: false.
- **ProtocolControllerRequiresAuthorizedIdentity**: false; `CreateProtocolControllerWithPorts` method is not supported.
- **ProtocolControllerSupportsCollections**: true; a collection of `StorageHardwareIDs` (host HBA WWNs/iSCSNames) is supported.
- **ExposePathsSupported**: true; the storage system supports the `ExposePaths` method.
- **CreateProtocolControllerSupported**: false; explicit creation of SPC is not supported.
- **MaximumMapCount**: 0 (unlimited); a virtual volume can be exported unlimited number of times.
- **SPCAllowsNoLUs**: false; clients cannot create SPC that has no associations to LUs (volumes).
- **SPCAllowsNoTargets**: true; a host-sees SPC might not have associations to any target port if it is in a dormant `OperationalStatus`.
- **SPCAllowsNoInitiators**: true; a port–present SPC (port-present type) might not have associations to any initiator port if it is in a dormant `OperationalStatus`.
- **SPCSupportDefaultViews**: false.

ControllerConfigurationService

`ControllerConfigurationService` (CCS) is the main class of Masking and Mapping (MM) subprofile. This class enables clients to export and delete VLUNs. A client can get a handle of CCS by using a `HostedService` association from `ComputerSystem` (`StorageSystem`).

Table 69 (page 77) provides a brief summary of the CSS methods and their corresponding CLI commands:

Table 69 CCS Methods and CLI Command Equivalents

Methods	CLI Equivalences	Comments
ExposePaths	createvlun vvname lunID hostname n:s:p	Host HBA WWN/iSCSIName (StorageHardwareID) is specified for ExposePaths method; host name (StorageHardwareIDCollection.ElementName) is specified for the CLI createvlun command, not the individual HBA WWN/iSCSIName.
	createvlun vvname lunID hostname	Host-sees type vlun can be created by `ExposePaths` by setting `TargetPortIDs` to NULL or empty array.
	createvlun vvname lunID n:s:p	
HidePaths	removevlun vvname lunID hostname n:s:p	Host HBA WWN/iSCSIName (StorageHardwareID) is specified for HidePaths method; host name (StorageHardwareIDCollection.ElementName) is specified for the CLI removevlun command, not the individual HBA WWN/iSCSIName.
	removevlun vvname lunID hostname	Remove a host-sees vlun.
	removevlun vvname lunID n:s:p	
ExposeLUsToStorageHardwareIDCollection	createvlun vvname lunID hostname	
HideLUsFromStorageHardwareIDCollection	removevlun vvname lunID hostname	

ExposePaths (Creating VLUNs)

`ControllerConfigurationService.ExposePaths()` performs VLUN export in one method call. It exposes a list of volumes to a list of host port (HBAs - initiators) via a list of system ports (targets). This method can be used either to create an `SCSIProtocolController` (SPC) and modify an existing SPC. If the `ProtocolController` parameter is null, the provider is supposed to create a new SPC.

This operation can be used to create matches-set, host-see or port-present VLUNs. Host-sees VLUNs can also be created by `ExposeLUsToStorageHardwareIDCollection`. One difference between this method and the CLI `createvlun` command is that for this method, the host StorageHardwareID (FC port WWN or iSCSI name) is specified in InitiatorPortIDs parameter, whereas for the `createvlun` command, the name of the host is specified, not the StorageID itself.

The method signature is described below:

```
uint32 ExposedPaths (
 [Out] CIM_ConcreteJob ref Job,
 [In] String LUNames[],
 [In] String InitiatorPortIDs[],
 [In] String TargetPortIDs[],
 [In] String DeviceNumbers[],
 [In] unit16 DeviceAccesses[],
 [In/Out] CIM_SCSIProtocolController ref ProtocolController[],
 [In] boolean Override,
 [In] boolean NoVCN,
 [Out] String ResultDescriptions[]);
```

- **Job**: This is ignored since storage system does not support job control.
- **LUNames[]**: An array of IDs of logical unit instances. The LU instances must already exist. The members of this array must match the Name property of `StorageVolume` instances.
- **InitiatorPortIDs[]**: IDs of initiator ports (host HBA WWNs or iSCSI names). If an existing `StorageHardwareID` instance exist, it is used. If no `StorageHardwareID` instance matches, one is implicitly created as is usually done from SMI-S.
- **TargetPortIDs[]**: IDs of target ports (the system's front-end FC port). If this is NULL or contains an empty array during creation mode, then a host-sees SPC is created; if array contains actual values, then a matched-set SPC is created
- **DeviceNumbers[]**: A list of logical unit numbers to assign to the corresponding logical unit in the `LUNames` parameter. (within the context of the elements specified in the other parameters). If the `LUNames` parameter is null, then this parameter must be null. If this parameter is NULL, the logical unit number is assigned by the system.
- **DeviceAccesses[]**: A list of permissions to assign to the corresponding logical unit in the `LUNames` parameter. This specifies the permission to assign within the context of the elements specified in the other parameters. This is mandatory only because SMI-S requires it. For the storage system, access permissions cannot be specified when creating a VLUN. Rather, each volume has its own access permission. The `ControllerConfigurationService` provider checks the passed in `DeviceAccesses` values against the permission of the specified LU (volume), and the operation fails if the two values do not match.
- **ProtocolControllers**: An array of references to `SCSIProtocolControllers` (SPCs). On input, this can be null, or contains exactly one element; there may be multiple references on output. If null on input, the instrumentation creates one or more new SPC instances. If SPC is included on input, this can either be a reference to `TPD_SCSIController` for an FC host or `TPD_iSCSIController` or`TPD_iSCSICNode` for an iSCSI host.

- **Override**: Override existing lower priority VLUNs, if necessary. Can be used only when exporting to a specific host. This parameter is a vendor-specific extension. The default value is false.
- **NoVCN**: Do not issue a VLUN (Virtual Logical Unit Number) Change Notification (VCN) after export. For direct connect or loop configuration, a VCN consists of a fibre channel Loop Initialization Primitive (LIP). For fabric configuration a VCN consists of Registered State Change Notification (RSCN) being sent to the fabric controller. This parameter is a vendor-specific extension. The default value is false.
- **ResultDescriptions[]**: An array of descriptive text of the result of the operation, with each entry containing the result of each expose path operation. This parameter is a vendor-specific extension.

Table 70 ExposePath Use Cases

Use Cases	LUNames	InitiatorPortIDs	TargetPortIDs	DeviceNumbers	DeviceAccesses	ProtocolControllers (On input)
Create a new view	Mandatory	Mandatory if creating a matched-set or host-see SPC, NULL if creating a port-present SPC	Mandatory if creating a matched-set or port-present SPC; NULL if creating a host-sees SPC	Optional	Mandatory	NULL
Add InitiatorIDs to View (valid only for host-see SPC, and matched-set SPC only if initiator port is already zoned with the target port)	NULL	Mandatory	NULL	NULL	NULL	Contains a single SPC ref
Add LUs to view	Mandatory	NULL	NULL	Optional	NULL	Contains a single SPC ref; if this is a TPD_iSCSINode, this operation is equivalent to creating a port-present SPC for that iSCSI port

Parameters “Override” and “NoVCN” are optional. All other combinations of the parameters are rejected.

HidePaths (Removing VLUNs)

`HidePaths()` is the inverse of `ExposePaths`. It hides a list of storage volumes from a list of initiators through a list of target ports, through one or more `SCSIProtocolControllers` (SPCs).

This operation is similar to `removevlun`. Host-sees VLUNs can also be removed by `HideLUsFromStorageHardwareIDCollection`. One difference between this method and the CLI `removevlun` command is that for this method, the exact host WWN/iscsiName is specified (`InitiatorPortIDs`), whereas for the `removevlun` command, the name of the host is specified, not the WWN/iscsiName itself.

The method signature is described below:

```
uint32 HidePaths (
 [Out] CIM_ConcreteJob ref Job,
 [In] String LUNames[],
 [In] String InitiatorPortIDs[],
 [In] String TargetPortIDs[],
 [In] String DeviceNumbers[],
 [In/Out] CIM_SCSIProtocolController ref ProtocolController[],
 [In] boolean NoVCN,
 [Out] String ResultDescriptions[]);
```

- **Job**: This is ignored since the storage system does not support job control.
- **LUNames[]**: An array of IDs of logical unit instances. The LU instances must already exist. The members of this array must match the Name property of the `StorageVolume` instances.
- **InitiatorPortIDs[]**: IDs of initiator ports (host HBA WWNs)
- **TargetPortIDs[]**: IDs of target ports (the storage system's front-end FC port).
- **DeviceNumbers[]**: A list of logical unit numbers corresponding to the logical unit in the `LUNames` parameter. This is a vendor-specific extension and therefore is optional.
- **ProtocolControllers[]**: An array of references to `SCSIProtocolControllers` (SPCs). On input, this contains exactly one element; there may be multiple references on output. If SPC is included on input, this can either be a reference to `TPD_SCSIController` for FC host or `TPD_iSCSIController` or`TPD_iSCSINode` for iSCSI host.
- **NoVCN**: Do not issue a VLUN (Virtual Logical Unit Number) Change Notification (VCN) after removal. For direct connect or loop configuration, a VCN consists of a fibre channel Loop Initialization Primitive (LIP). For fabric configuration a VCN consists of Registered State Change Notification (RSCN) being sent to the fabric controller. This parameter is a vendor-specific extension. The default value is false.
- **ResultDescriptions[]**: An array of descriptive text of the result of the operation, with each entry containing the result of each hide path operation. This parameter is a vendor-specific extension .

Table 71 HidePaths Use Cases

Use Cases	LUNames	InitiatorPortIDs	TagetPortIDs	ProtocolControllers (On input)
Remove LUs from a view	Mandatory	NULL	NULL	Contains a single SPC ref; if this is a TPD_iSCSINode, this operation is equivalent to removing an LU from a port-present SPC for that iSCSI port.
Hide full paths from a view	Mandatory	Mandatory	Mandatory	Contains a single SPC ref
Remove Initiator IDs from view	NULL	Mandatory	NULL	Contains a single SPC ref

Parameters `DeviceNumbers` and `NoVCN` are optional. All other combinations of the parameters are rejected.

ExposeDefaultLUs

Use of `ExposeDefaultLUs` has been deprecated. Please use ExposePaths instead to create a port-present SPC.

HideDefaultLUs

Use of HideDefaultLUs has been deprecated. Please use HidePaths instead.

ExposeLUsToStorageHardwareIDCollection (Creating Host-Sees VLUNs) (3PAR proprietary method)

`ControllerConfigurationService.ExposeLUsToStorageHardwareIDCollection()` is similar to `ExposePaths`, except `ExposeLUsToStorageHardwareIDCollection` exposes volumes to all the initiator ports (`StorageHardwareID`) on the specified hosts (`StorageHardwareIDCollection`) regardless of which target ports they are connected to.

The parameter, `TargetPortLIPDs`, which if specified, also allows for creation of matched-set VLUNs; this allows for more flexibility in creating template VLUNs to target ports that might not exist yet.

The method signature is described below:

```
uint32 ExposedLUsToStorageHardwareIDCollection (
 [Out] CIM_ConcreteJob ref Job,
 [In] String LUNames[],
 [In] String HostNames[],
 [In] string TargetPortLPIDs[],
 [In] String DeviceNumbers[],
 [In] unit16 DeviceAccesses[],
 [In/Out] CIM_SCSIProtocolController ref ProtocolController[],
 [In] boolean Override,
 [In] boolean NoVCN,
 [Out] String ResultDescriptions[]);
```

- **Job**: This is ignored since the storage system does not support job control.
- **LUNames[]**: An array of IDs of logical unit instances. The LU instances must already exist. The members of this array must match the Name property of `StorageVolume` instances.
- **HostNames[]**: An array of names for the hosts (`StorageHardwareIDCollection`). Instances MUST already exist. The members of this array MUST match the `ElementName` property of `TPD_StorageHardwareIDCollection`.
- **TargetPortLPIDs[]**: An array of target port LPIDs, in '`<node>:<slot>:<port>`' format. If this is empty, then host-sees vlun is created; otherwise matched set vlun(s) is created.
- **DeviceNumbers[]**: A list of logical unit numbers to assign to the corresponding logical unit in the `LUNames` parameter. (within the context of the elements specified in the other parameters). If the `LUNames` parameter is null, then this parameter must be null. If this parameter is NULL, the logical unit number will be assigned by the system.
- **DeviceAccesses[]**: A list of permissions to assign to the corresponding logical unit in the `LUNames` parameter. This specifies the permission to assign within the context of the elements specified in the other parameters. This is mandatory only to be consistent with other methods. For the storage system, access permissions cannot be specified when creating a VLUN. Rather, each volume has its own access permission. The `ControllerConfigurationService` provider checks the passed in `DeviceAccesses` value against the permission of the specified LU (volume), and the operation fails if the two values do not match.
- **ProtocolControllers[]**: An array of references to `SCSIProtocolControllers` (SPCs). On input, this can be null, or contain exactly one SPC; there may be multiple references on output. If null on input, the instrumentation creates one or more new SPC instances. If SPC is included

on input, this can either be reference to `TPD_SCSIController` for FC host or `TPD_iSCSIController` for iSCSI host

- **Override**: Override existing lower priority VLUNs, if necessary. Can be used only when exporting to a specific host. This parameter is a vendor-specific extension for 3PAR. The default value is false.
- **NoVCN**: Do not issue a VLUN (Virtual Logical Unit Number) Change Notification (VCN) after export. For direct connect or loop configuration, a VCN consists of a fibre channel Loop Initialization Primitive (LIP). For fabric configuration a VCN consists of Registered State Change Notification (RSCN) being sent to the fabric controller. This parameter is a vendor-specific extension. The default value is false.
- **ResultDescriptions[]**: An array of descriptive text of the result of the operation, with each entry containing the result of each expose path operation. This parameter is a vendor-specific extension.

Table 72 ExposeLUsToStorageHardwareIDCollection Use Cases

Use Cases	LUNames	HostNames	TargetPortLPIDs	DeviceNumbers	DeviceAccesses	ProtocolControllers (On input)
Create a new view	Mandatory	Mandatory	Optional	Mandatory for 2.2.4 and before, optional for 2.3.1 and beyond.	Mandatory	NULL

Parameters "`Override`" and "`NoVCN`" are optional. All other combinations of the parameters are rejected.

There is one use case of `ExposeLUsToStorageHardwareIDCollection()` method (See Table 72 (page 82)).

- **Creating a new view**: this creates host-sees VLUN if `TargetPortLPIDs` is NULL, or matched-set VLUN if `TargetPortLPIDs` is specified. `ProtocolControllers` parameter must be NULL.

HideLUsFromStorageHardwareIDCollection(Removing Host-Sees VLUNs)

`HideLUsFromStorageHardwareIDCollection()` is a method that is similar to `HidePaths`, except that it hides volumes from all host HBA initiator ports on the specified hosts.

This operation is similar to `removevlun` of host-sees VLUNs. For matched-set VLUNs, see `HidePaths`. For port-present VLUNs, see `HideDefaultLUs`.

The method signature is described below:

```
uint32 HideLUsFromStorageHardwareIDCollection (
 [Out] CIM_ConcreteJob ref Job,
 [In] String LUNames[],
 [In] String DeviceNumbers[],
 [In/Out] CIM_SCSIProtocolController ref ProtocolController[],
 [In] boolean NoVCN,
 [Out] String ResultDescriptions[]);
```

- **Job**: This is ignored since the storage system does not support job control.
- **LUNames[]**: An array of IDs of logical unit instances. The LU instances must already exist. The members of this array must match the Name property of StorageVolume instances.
- **DeviceNumbers[]**: A list of logical unit numbers corresponding to the logical unit in the `LUNames` parameter. This is a vendor-specific extension and is optional.

- **ProtocolControllers[]**:An array of references to `SCSIProtocolControllers` (SPCs). On input, this must contain exactly one SPC. On output, this is the set of SPCs affected. If SPC is included on input, this can either be reference to `TPD_SCSIController` for a FC host or `TPD_iSCSIController` for an iSCSI host.
- **NoVCN**: Do not issue a VLUN (Virtual Logical Unit Number) Change Notification (VCN) after removal. For direct connect or loop configuration, a VCN consists of a fibre channel Loop Initialization Primitive (LIP). For fabric configuration a VCN consists of Registered State Change Notification (RSCN) being sent to the fabric controller. This parameter is a vendor-specific extension. The default value is false.
- **ResultDescriptions[]**: An array of descriptive text of the result of the operation, with each entry containing the result of each hide path operation. This parameter is a vendor-specific extension .

Table 73 HideLUsFromStorageHardwareIDCollection Use Cases

Use Cases	LUNames	ProtocolControllers (On input)
Remove LUs from a view	Mandatory	Contains a single SPC ref

Parameters "`DeviceNumbers`" and "`NoVCN`" are optional. All other combinations of the parameters are rejected.

The use cases are:

- Remove volumes from a view: the volume has been exported to an SPC, it'll be removed from the SPC (e.g., it won't be accessible from host HBA anymore).

StorageHardwareIDManagementService

`StorageHardwareIDManagementService` provides methods for manipulating instances of `StorageHardwareIDs` and manipulating the trust of these IDs in the underlying storage system. This class enables clients to create and delete `StorageHardwareID`, create `StorageHardwareIDCollection` and add `StorageHardwareID` to a collection. A client can get a handle of `StorageHardwareIDManagementService` by using a `HostedService` association from `ComputerSystem` (`StorageSystem`).

Here is a brief summary of the methods and their corresponding CLI commands:

Table 74 StorageHardwareIDManagementService Methods and CLI Equivalents

Methods	CLI Equivalence	Comments
CreateStorageHardwareID	createhost hostname WWN	Only 1 host HBA WWN/iscsiName (`StorageID`) can be specified at a time for `CreateStorageHardwareID` method.
DeleteStorageHardwareID	removehost hostname WWN	Only 1 host HBA WWN/iscsiName (`StorageID`) can be deleted at a time for `DeleteStorageHardwareID` method.
CreateStorageHardwareIDCollection	createhost hostname [WWN …]	
AddHardwareIDsToCollection	createhost –add hostname WWN…	
StorageHardwareIDCollection.DeleteInstance	removehost hostname	Delete the entire host and any path to the host (`StorageHardwareID`).

CreateStorageHardwareID (Creating Host and Path)

This method creates a `StorageHardwareID` (host HBA WWN/iscsiBane) on a host, it creates the association `TPD_MgmtServicesForStorageHWIDCollection` between `TPD_StorageHardwareIDManagementService` and the new `TPD_StorageHardwareID`.

This operation is similar to "`createhost <hostname> <wwn>`" or "`createhost –iscsi <hostname> <iscsiName>`". If ElementName is NULL, a new host with a system-generated `ElementName` will be created first and the `StorageID` will be added to the host. If a host already exists with the same `ElementName`, then the `StorageID` will be added to that host. Otherwise a new host will be created with the `ElementName` and `StorageID` will be added to the host.

The method signature is described below:

```
uint32 CreateStorageHardwareID (
 [In] String ElementName,
 [In] String StorageID,
 [In] Uint16 IDType,
 [In] String OtherIDType,
 [In] CIM_StorageClientSettingData REF Setting,
 [Out] CIM_StorageHardwareID REF HardwareID);
```

- **ElementName**: This is the name of the host to create the host HBA WWN/iSCSI name on. If this is NULL, a new host with a system-generated `ElementName` will be created. If a host already exists with the same `ElementName`, then the `StorageID` will be added to that host. Otherwise a new host will be created with the `ElementName` and `StorageID` will be added to the host.
- **StorageID**: WWN of the FC host HBA or `iscsiName` of the iSCSI host HBA to create. This is mandatory.
- **IDType**: Type of `StorageID` property. This must be 2 (Port WWN for FC target port) or 5 (`iscsiName` for iSCSI target port); all other values are rejected. This is mandatory.
- **OtherIDType**: Description of the type of `StorageID` if IDType is 1 (Other). Since only IDType of 2 and 5 is accepted, this value is ignored.
- **Setting**: Reference to the `StorageClientSettingData` containing the `OSType` appropriate for this initiator. This refers to an `TPD_StorageClientSettingData` which is equivalent to a host persona; if not specified then the host persona defaults to `Generic`. If this operation is being applied to a `StorageID` on an existing host with a different Setting, then the host will be reconfigured to use this Setting.
- **HardwareID**: Reference to the new `StorageHardwareID` instance. Must be NULL on input.

DeleteStorageHardwareID (Removing a Path from a Host)

This method deletes a named `StorageHardwareID` (host HBA WWN/iscsiName) from a host, and also tears down the associations that are no longer needed, including `CIM_ConcreteDependency` and `CIM_AuthorizedSubject`.

This operation is similar to "`removehost <hostname> <wwn>`" or "`removehost –iscsi <hostname> <iscsiName>`".

The method signature is described below:

```
uint32 DeleteStorageHardwareID (
[In] CIM_StorageHardwareID REF HardwareID);
```

- **HardwareID**: Reference to the new `StorageHardwareID` instance to delete. This is mandatory.

CreateStorageHardwareIDCollection (Creating a Host)

This method creates a `StorageHardwareIDCollection` (host) and optionally a list of WWNs/iscsiNames to assign to it. This is useful to define a set of authorized subjects that can access volumes. This method causes the creation of the `HostedCollection` association and `MemberOfCollection` association to the members of the IDs parameter.

This operation is similar to "`createhost <hostname>`" and "`createhost <hostname> <wwn …>`" or "`createhost –iscsi <hostname> <iscsiName>`". The host as specified in `<hostname>` must not have existed. If `HardwareIDs` (WWN/iscsiNames) are not specified, then only the host is created.

The method signature is described below:

```
uint32 CreateStorageHardwareIDCollection (
 [In] String ElementName,
 [In] String HardwareIDs[],
 [In] CIM_StorageClientSettingData REF Setting,
 [In] String Domain,
 [Out] CIM_SystemSpecificCollection REF Collection);
```

- **ElementName**: This is the name of the host to create the host HBA WWN/iscsiName on. The host must not have existed. This corresponds to the `ElementName` property of the `StorageHardwareID` instance. If the host already exist, use `AddHardwareIDsToCollection` instead. This is mandatory.
- **HardwareIDs**: A list of WWN's/iscsiNames to assign to the host. If this is null, only the host (without WWN/iscsiName assigned) is created.
- **Setting**: Reference to the `StorageClientSettingData` containing the `OSType` appropriate for this initiator. This refers to an `TPD_StorageClientSettingData` which is equivalent to a host persona; if not specified then the host persona defaults to Generic.
- **Domain**: Name of the domain the host should belong to
- **Collection**: Reference to the new `StorageHardwareIDCollection` (host) instance. Must be NULL on input.

AddStorageHardwareIDsToCollection (Adding a Path to a Host)

This method adds a list of WWNs (`StorageHardwareID`) to the host (`StorageHardwareIDCollection`). This creates the `MemberOfCollection` instances between the specified Collection and the `StorageHardwareIDs`.

This operation is similar to "`createhost -add <hostname> <wwn …>`" or "`createhost –add –iscsi <hostname> <iscsiName>`". The host as specified in <hostname> must have existed.

The method signature is described below:

```
uint32 AddHardwareIDsToCollection (
[In] String HardwareIDs[],
[In] CIM_SystemSpecificCollection REF Collection);
```

- **HardwareIDs**: A list of WWN's/iscsiNames to assign to the host. This is mandatory.
- **Collection**: A reference to the `StorageHardwareIDCollection` (host) to add the WWN's/iscsiNames to. This is mandatory.

SetISCSICHAP

This method sets the initiator or the target host CHAP secrets or both.

This operation is similar to "`sethost initchap <secret> <hostname>`" or "`sethost targetchap <secret> <hostname>`". The host as specified in <hostname> must have existed.

The method signature is described below:

```
uint32 SetISCSICHAP (
[In] String InitiatorCHAPName,
[In] String TargetCHAPName,
[In] uint16 InitiatorSecretType,
[In] uint16 TargetSecretType,
[In] String InitiatorSecret,
[In] String TargetSecret,
[In] CIM_SystemSpecificCollection REF Collection);
```

- **InitiatorCHAPName**: initiator CHAP name. Defaults to the host name.
- **TargetCHAPName**: target CHAP name. Defaults to the system name.
- **InitiatorSecretType**: whether initiator secret string is ASCII or hex. If ASCII, the initiator secret string must be printable and 12 to 16 characters in length with no spaces; if hex, the initiator secret string must be 16 bytes in length. Defaults to ASCII.
- **TargetSecretType**: whether target secret string is ASCII or hex. If ASCII, the target secret string must be printable and 12 to 16 characters in length with no spaces; if hex, the target secret string must be 16 bytes in length. Defaults to ASCII.
- **InitiatorSecret**: CHAP secret for the host
- **TargetSecret**: CHAP secret for the target.
- **Collection**: A reference to the StorageHardwareIDCollection (host). This is mandatory.

RemoveISCSICHAP

This method removes just the target host CHAP secrets, or both the target and the initiator host CHAP secrets.

The method signature is described below:

```
uint32 RemoveISCSICHAP (
        [In] Uint16 RemovalScope,
        [In] CIM_SystemSpecificCollection REF Collection);
```

- **RemovalScope**: Remove target secret only or both initiator and target secrets. Valid values are 1: remove target CHAP only, 2: remove both initiator and target CHAP Defaults to 2 which means remove both.
- **Collection**: A reference to the StorageHardwareIDCollection (host). This is mandatory.

StorageHardwareIDCollection.ModifyInstance

The intrinsic method `ModifyInstance`is used to modify persona or domain of the host.

StorageHardwareIDCollection.DeleteInstance (Deleting a Host)

The intrinsic method `DeleteInstance` is used to delete `StorageHardwareIDCollection` (host).

ConcreteDependency

There are three ConcreteDependency classes in MM subprofile:

- `TPD_ConfigServicesForSCSIController`: association between CCS with `TPD_SCSIProtocolController`.
- `TPD_ConfigServicesForiSCSIController`: association between CCS with `TPD_iSCSIProtocolController`.
- `TPD_MgmtServicesForStorageHWIDCollection`: association between `StorageHardwareID` and `StorageHardwareIDManagementService`.

SAPAvailableForElement

This class replaces `PrococolControllerForPort` in SMI-S 1.0.2. `TPD_SAPAvailableForElement` associates `TPD_SCSIPrococolController` to `TPD_SCSIPrococolEndPoint` (FC target port). When a host is directly attached to the storage system, `SAPAvailableForElement` represents a one-to-one relationship between SPC and `SCSIProtocolEndPoint` because only one host can connect to one target port. When one storage system port is connected to a switch in a fabric, there can be multiple hosts connecting to the storage system via the same target port (1-to-many association). When more than one the storage system port are connecting to the fabric with multiple hosts in the same zone, this association becomes n-to-n relationship.

iSCSISAPAvailableForElement

This class replaces `PrococolControllerForPort` in SMI-S 1.0.2. `TPD_iSCSISAPAvailableForElement` associates `TPD_iSCSIPrococolController` to `TPD_iSCSIPrococolEndPoint` (iSCSI target port). When a host is directly attached to the storage system, `iSCSISAPAvailableForElement` represents a one-to-one relationship between SPC and `iSCSIPrococolEndPoint` because only one host can connect to one target port. When one storage system port is connected to a switch in a fabric, there can be multiple hosts connecting to the system via the same target port (1-to-many association). When more than one system port are connecting to the fabric with multiple hosts in the same zone, this association becomes n-to-n relationship.

SCSIPrococolController

`SCSIPrococolController` (SPC) represents the view described in Section (page 73) for a Fiber-channel host.

iSCSIPrococolController

`iSCSIPrococolController` (SPC) represents the view described in Section (page 73) for an iSCSI host.

SCSIPrococolEndpoint

`SCSIPrococolEndpoint` (subclass of `LogicalPort`) models SCSI aspect of an FC port. It is associated with one or more `FCPorts`. One instance of this class must be created for each target port (connecting to host or fabric).

Table 75 Properties for

Properties	Values
string **Name**	Target port WWN
Uint16 **ProtocolIFType**	56 (Fibre Channel)

Table 75 Properties for *(continued)*

Properties	Values
Uint16 **ConnectionType**	2 (Fibre Channel)
Uint16 **Role**	2 (target port)

iSCSIPrococolEndpoint

`iSCSIPrococolEndpoint` (subclass of `LogicalPort`) models SCSI aspects of an iSCSI port. Unlike `TPD_SCSIPrococolEndpoint`, however, this class is not associated to any `CIM_LogicalPort` class since one does not exist for iSCSI port. One instance of this class must be created for each target port (connecting to host or fabric).

Table 76 Properties Values for ISCSIPrococolEndpoint

Properties	Values
string **Name**	Target port iSCSIName + 't' + TPGT
Uint16 **ProtocolIFType**	1 (other)
string **OtherTypeDescription**	"iSCSI"
Uint16 **ConnectionType**	7 (iSCSI)
Uint16 **Role**	3 (target port)

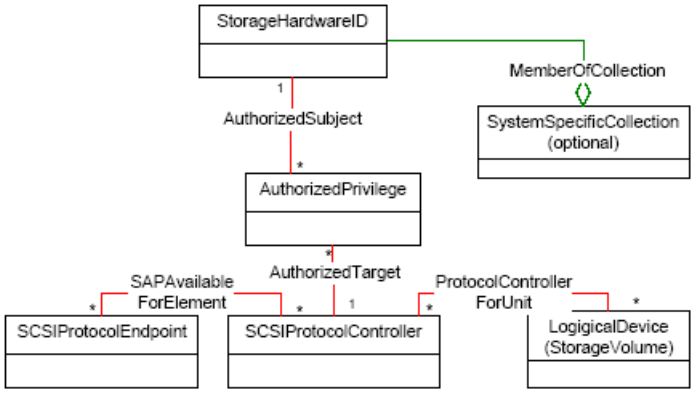
ControllerForUnit

`TPD_ControllerForUnit` associates `TPD_SCSIController` for an FC host with `StorageVolumes`. The LUN is modeled as the `DeviceNumber` property of `ProtocolControllerForUnit`. The access permission is represented as `DeviceAccess` property.

Table 77 Property Values for TPD_ControllerForUnit

Properties	Values
TPD_SCSIController **Antecedent**	Reference to a TPD_SCSIController instance.
TPD_StorageVolume **Dependent**	Reference to an exported TPD_StorageVolume instance that is visible to protocol controller above.
string **DeviceNumber**	LUN of the storage volume via the above protocol controller.
Uint16 **DeviceAccess**	This value has to match that of StorageVolume.

Figure 10 Relationships between StorageHardwareID, SCSIProtocolEndpoint and StorageVolume



iSCSIControllerForUnit

Similar to `TPD_SCSIControllerForUnit`, except that `TPD_iSCSIControllerForUnit` associates `TPD_iSCSIController` for an iSCSI host with `StorageVolumes`.

StorageClientSettingData

An array has several fixed instances of this class, each corresponding to a host persona. Each instance of `StorageHardwareID` and `StorageHardwareIDCollection` has an `ElementSettingData` association with an instance of `StorageClientSettingData` that corresponds to the persona of the host.

To change the `StorageClientSettingData` of a host, a client should do a `ModifyInstance` of the `TPD_StorageHardwareIDCollection.Setting` property.

StorageHardwareID

`StorageHardwareID` (subclass of Identity) represents the host HBA WWN/iscsiName.

TPD_StorageHardwareIDCollection

This is sub-classed from `SystemSpecificCollection` is used to hold a set of `StorageHardwareID` objects which can access storage volumes.

Supporting classes

AuthorizedPrivilege

`AuthorizedPrivilege` represents access permission (read-write) that a `StorageHardwareID` (host HBA) is allowed on a `SCSIProtocolController`. This permission is associated with `StorageHardwareID` via `AuthorizedSubject`.

Table 78 Property Values for AuthorizedPrivilege

Properties	Values
string **ElementName**	User friendly name.
string **InstanceID**	Opaque unique identifier.
boolean **PrivilegeGranted**	True – privilege is granted, False – not granted.
Uint16[] **Activities**	SMI-S: this array must contain 5 (read) and 6 (write).

PrivilegeForStorageHardwareID

`PrivilegeForStorageHardwareID` is subclassed from `AuthorizedSubject` and is an association between `StorageHardwareID` and the access permission (`AuthorizedPrivilege`) of the SPC that is exported to it.

PrivilegeForStorageHardwareIDCollection

`PrivilegeForStorageHardwareIDCollection` is subclassed from `AuthorizedSubject` and is an association between `StorageHardwareIDCollection` and the access permission (`AuthorizedPrivilege`) of the SPC that is exported to one or more of its member `StorageHardwareID`.

AuthorizedSubject

`AuthorizedSubject` associates `StorageHardwareID` to its access permission (`AuthorizedPrivilege`).

AuthorizedTarget

`AuthorizedTarget` is an association that ties `AuthorizedPrivilege` with `SCSIProtocolController`.

ConcreteDependency

`ConcreteDependency` links the `SCSIProtocolController` to `ControllerConfigurationService`.

SystemEndpoint (HostedAccessPoint)

`TPD_SystemEndpoint` (a subclass of `CIM_HostedAccessPoint`) is an association between `ComputerSystem` and `SCSIProtocolEndpoint`.

ElementCapabilities

`ElementCapabilities` is an association between `ComputerSystem` and `ProtocolControllerMaskingCapabilities`.

ElementSettingData

There are two kinds of `ElementSettingData` associations:

1. Association between `StorageHardwareID` and `StorageClientSettingData`.
2. Association between `StorageHardwareIDCollection` and `StorageClientSettingData`.

MemberOfCollection

`MemberOfCollection` is an association between `StorageHardwareID` and `SystemSpecificCollection`. It indicates that a specific `StorageHardwareID` (host hba wwn/iscsiName) is a member of `SystemSpecificCollection/StorageHardwareIDCollection` (host).

ProtocolControllerMaskingCapabilities

See Section (page 76).

FC Target Ports Subprofile

The FC Target Ports subprofile models Fibre Channel specific aspects of a target storage system. This subprofile provides information about FC ports on an HP 3PAR Storage System that are connected to hosts or FC switches.

For detailed information regarding the FC Target Ports subprofile, refer to SMI-S at http://www.snia.org.

FC Target Ports Subprofile CIM Classes

Table 79 FC Target Ports Subprofile CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the target ports reside.
TPD_SCSIProtocolFCEndpoint	A SCSI transport endpoint. In this context, this represents a target port on the HP 3PAR Storage System.
TPD_FCPort	A Fibre Channel Port residing on the `TPD_ComputerSystem`.
TPD_SCSIController	Represents the SCSI view of ports as seen by SCSI initiators.

Supported Methods

None

FC Initiator Ports Subprofile

The FC Initiator Ports subprofile models Fibre Channel specific aspects of a storage system. This subprofile provides information about the backend Fibre Channel ports that are used to connect to the storage system controller nodes to backend Disk Enclosures (Drive Cages).

For detailed information regarding the FC Target Ports subprofile, refer to SMI-S at www.snia.org.

FC Initiator Ports Subprofile CIM Classes

Table 80 FC Initiator Ports Subprofile CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the initiator ports reside.
TPD_SCSIProtocolFCEndpoint	A SCSI transport endpoint. In this context, this represents an initiator port on the HP 3PAR Storage System.
TPD_FCPort	A Fibre Channel Port residing on the `TPD_StorageSystem`.

Supported Methods

Table 81 (page 91) describes the supported methods of the FIbre Channel initiator ports subprofile:

Table 81 Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

iSCSI Target Ports Subprofile

The iSCSI Target Ports subprofile models iSCSI specific aspects of a target storage system. This subprofile provides information about FC ports on a storage system that are connected to hosts or FC switches.

The Multiple Computer System subprofile models the underlying systems or controllers that cooperate to present a single, top-level system (Cluster) which provides for added redundancy or increased functionality.

For detailed information regarding the FC Target Ports subprofile, refer to SMI-S at http://www.snia.org.

The CIM Server supports the Multiple Computer System subprofile as defined in SMI-S v1.1.0. This subprofile provides information about the controller nodes that comprise the storage system cluster. The cluster is modeled as multiple tiers with each tier representing a pair of controller nodes.

iSCSI Target Ports Subprofile CIM Classes

Table 82 iSCSI Target Ports Subprofile Key CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the target ports reside.
TPD_iSCSIProtocolEndpoint	An iSCSI transport endpoint. Represents an iSCSI port.
TPD_EthernetPort	An Ethernet Port residing on the `TPD_StorageSystem`.
TPD_SCSIController	Represents the SCSI view of ports as seen by SCSI initiators.
TPD_iSCSICapabilities	Provides information about the iSCSI capabilities of an HP 3PAR Storage System.
TPD_iSCSISession	Provides information about iSCSI Sessions, which are established between an initiator port and a target port.
TPD_iSCSISessionSettings	Provides information about session-wide operational properties, which are used during negotiation when creating new iSCSI Sessions.
TPD_iSCSIConnection	Provides information about the negotiated values for an iSCSI connection.
TPD_iSCSINode	The iSCSI Node represents a single iSCSI Target. Each iSCSI port has one instance of `TPD_iSCSINode` that can also be used for Masking and Mapping purpose.

Supported Methods

Table 83 (page 92) describes the supported methods of the iSCSI target ports subprofile:

Table 83 Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

Distinction between iSCSI and iSCSI over CNA Port

iSCSI port configured on a CNA card behaves exactly like an iSCSI target port, but the key difference is that the corresponding TPD_EthernetPort.PortType property for the former has a value of 105 (10GBase-SR) while that for the latter has a value of 53 (1000BaseT).

Disk Drive Lite Subprofile

The Disk Drive Lite subprofile models information for disk drive devices. The information provided via this subprofile is information about the disks that are accessible to a specific storage system.

For detailed information regarding the Disk Drive Lite subprofile, refer to SMI-S at http://www.snia.org.

Disk Drive Lite Subprofile CIM Classes

Table 84 Disk Drive Lite Subprofile Key CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the disk drives reside.
TPD_DiskDrive	Information about a specific disk drive residing in the TPD_StorageSystem.
TPD_DiskDrivePackage	Provides physical information about the disk drive such as Vendor, Model, Serial Number.
TPD_DiskSoftwareIdentity	Provides information about installed firmware for disk drives.
TPD_DiskStorageExtent	Provides capacity information for disk drives.
TPD_StoragePool	Identifies which Storage Pool contains the storage provided by specific disk drives.
TPD_VolumeBasedOnDiskExtent	Associations between a `StorageVolume` and the `DiskStorageExtent` that makes up the volume. Includes support for TPVV.
TPD_StoragePoolComponent	Associations between a storage pool and the `DiskStorageExtent` that make up the pool. Support for `DynamicStoragePool` (CPG) and `DeltaReplicaStoragePool` (snap space) is included.

Supported Methods

Table 85 (page 93) lists the supported method of the Disk Drive Lite subprofile:

Table 85 Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

Block Server Performance Subprofile

The Block Server Performance Subprofile defines classes and methods for managing performance information (statistics) in block servers.

For detailed information regarding the Block Server Performance subprofile, refer to SMI-S at http://www.snia.org.

Block Server Performance Subprofile CIM Classes

Table 86 Block Server Performance Subprofile Key CIM Classes

Class	Description
TPD_StorageSystem	The storage array on which the statistics are to be collected.
TPD_StatisticsCollection	A collection of statistics data.
TPD_DiskStatisticalData	An instance of statistics data for a `TPD_DiskDrive`.
TPD_PortStatisticalData	An instance of statistics data for a `TPD_FCPort`.
TPD_VolumeStatisticalData	An instance of statistics data for a `TPD_StorageVolume`.
TPD_LUNStatisticalData	An instance of statistics data for an exported `TPD_StorageVolume`.

Table 86 Block Server Performance Subprofile Key CIM Classes *(continued)*

Class	Description
TPD_ArrayStatisticalData	An instance of statistics data for the entire array.
TPD_NodeStatisticalData	An instance of statistics data for a particular node controller.

Replication Services Profile

This new profile extends the functionality of the Copy Services Subprofile by including enhanced local replication for thinly provisioned storage objects, remote replication (not supported in this release), and support for replication groups and consistency group.

Key features include:

- Specify individual or Groups of elements to manage replication
- Support Consistency Management
- Handle local, and in the future remote, replication seamlessly
- Explicitly replicate Thinly Provisioned elements
- Offer different Copy Methodologies

SMI-S Terminology:

1. **Synchronized Replication** - indicates that updates to a source element are reflected to the target element, although the target element may or may not be updated immediately
2. **SyncType** - describes the replication policy supported by this profile. There are three defined types:
   a. **Clone** - Creates a point-in-time, independent, copy of the source element.
   b. **Mirror** - Creates and maintains a synchronized mirror copy of the source. Writes done to the source element are reflected to the target element. The target element remains dependent on the source element.
   c. **Snapshot** - Creates a point-in-time, virtual image of the source element. The target element remains dependent on the source element. Snapshots are commonly known as delta replicas and contain incrementally changed data as well as references (e.g. pointers) to the unchanged source element data.
3. **Mode** - controls when the write operations are performed. There are 2 defined modes:
   a. **Synchronous** - The writer waits until the write operations are committed to both the source and target elements; or to both the source element and a target related entity, such as pointer tables.
   b. **Asynchronous** - The writer waits until the write operations are committed to the source elements only. In this mode, there can be a delay before the write operations are committed to the target elements.

4. **Locality** - specifies the relationship between the source and target element. There are two defined localities:
   a. **Local** - indicates the source and target elements are contained in a single managed system.
   b. **Remote** - indicates the source and target elements are contained in separate managed systems. In this case, the service must rely on a networking protocol for the copy operations. HP 3PAR SMI-S supports read-only implementation of remote replication.

SMI-S terminologies can be mapped to HP terminologies.

Table 87 SMI-S/HP Terminology Map

Type of HP Copy	Sync Type	Mode	Locality	Note
Physical copy	Clone	Synchronous	Local	Target is synchronized only manually
Virtual copy	Snapshot	Synchronous	Local	Target sv contains synchronously updated delta of the source volume
Remote copy, with synchronous mode volume groups	Mirror	Synchronous	Remote	Read-only supported
Remote copy, with asynchronous periodic mode volume groups	Mirror	Asynchronous	Remote	Read-only supported

CIM Classes

TPD_ReplicationGroup

TPD_ReplicationGroup is a volume set that can be used for local clones or snapshots.

Replication Group Key features:

- A group can be the source and/or the target of a copy operation, i.e., volume set
- Elements of a group may be optionally declared copy relations Consistent
- A group may contain zero elements (an empty group)

Volume set is represented by the `TPD_ReplicationGroup` class, derived from `CIM_ReplicationGroup` class. This profile enables management of replication group itself (create, delete, add member, remove member) as well as copying of the group.

SMI-S also specifies that synchronized groups can maintain replication relationship through the GroupSynchronized association, as opposed to StorageSynchronized with single volume pair. InForm OS tracks only the copy relationships between the individual volumes of a `vvset`, but not of the `vvset` itself, hence GroupSynchronized association is not supported for local replication.

Table 88 Relevant Properties of ReplicationGroup Class

Properties	Value	Description
InstanceID	3PAR:<vvset name>	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class
Persistent	True	If false, the group, not the elements associated with the group, may be deleted at the completion of a copy operation. This is always true.

Table 88 Relevant Properties of ReplicationGroup Class *(continued)*

Properties	Value	Description
DeleteOnEmptyElement	False	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty. HP supports empty set and thus this will always be false.
DeleteOnUnassociated	False	If true, the group will be deleted when the group is no longer associated with another group. This can happen if all synchronization associations to the individual elements of the group are dissolved. VV set can exist without a synchronization relationship, thus this will always be false.

TPD_RemoteReplicationGroup

TPD_RemoteReplicationGroup is a copy group that is used for remote copy (remote mirrors), sub-classed from CIM_ReplicationGroup. Please note that creation, modification and deletion of `RemoteReplicationGroup` is not supported.

Table 89 Relevant Properties of RemoteReplicationGroup

Properties	Value	Description
InstanceID	3PAR:<name of rcopy group>	
ElementName	Name of the rcopy group	
Persistent	True	
DeleteOnEmptyElement	False	A remote copy group can be empty.
DeleteOnUnassociated	False	Remote copy group can exist without synchronization.

ReplicationService

ReplicationService class contains extrinsic methods for group management as well as replication management; the latter meant to replace Copy Services methods in StorageConfigurationService.

All extrinsic methods in this class return one of the following status codes:

- 0: (Job) Completed with no error
- 1: Method not supported
- 4: Failed
- 5: Invalid Parameter
- 4096: Method Parameters Checked - Job Started

Replication Services includes methods to create and delete a group, and methods to add elements or pair of elements to an existing group(s) or to remove elements from a group. It can then utilize Groups of elements to manage replication activities that include more than one source or target element in a copy operation.

CreateGroup

```
uint32 ReplicationService.CreateGroup(
[IN]  string GroupName,
[IN]  CIM_LogicalElement REF Members[],
[IN]  boolean Persistent,
[IN]  boolean DeleteOnEmptyElement,
[IN]  boolean DeleteOnUnassociated,
[IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
[OUT] SNIA_ReplicationGroup REF ReplicationGroup );
```

Use to create a new replication group. Any required associations (such as HostedCollection) are created in addition to the instance of the group. Creation of RemoteReplicationGroup is not supported. Parameters are:

- **GroupName**: If nameable, represents a user-friendly name for the group being created. This parameter is mandatory and represents the name of the vv set. This will also become the value in ElementName property of the resulting ReplicationgGroup instance.
- **Members[]**: An array of strings containing object references to the volumes to add to the group – order is maintained. If NULL, the group will be empty. Duplicates members are not allowed.
- **Persistent**: If true, the group must persist across Provider reboots (group is not temporary). We only support persistent group, so the method will return an error if this is set to false.
- **DeleteOnEmptyElement**: If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. This can only be false, as deletion of empty vv set can only be done manually, so the method will return an error if this is set to true.
- **DeleteOnUnassociated**: If true, the group will be deleted when the group is no longer associated with another group. This can only be false, as deletion of non-synchronized vv set can only be done manually, so the method will return an error if this is set to true.
- **ServiceAccessPoint**: Reference to access point information to allow the service to create a group on a remote system. If NULL, the group is created on the local system. Since we do not support remote copy yet, this has to be NULL.
- **ReplicationGroup**: If the method completes successfully, then the ReplicationGroup is a reference to the group that is created.

ReplicationGroup.ModifyInstance

Use ModifyInstance method on a ReplicationGroup instance to change its ElementName and/or Description property, equivalent to changing the name and comment using the CLI command, setvvset.

DeletionGroup

```
uint32 ReplicationService.DeleteGroup(
[IN, Required]  SNIA_ReplicationGroup REF ReplicationGroup.
[IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
[IN]  boolean RemoveElements  );
```

Use to delete a replication group. All associations to the deleted group are also removed as part of the action. Parameters are:

- **ReplicationGroup**: This is a reference to the group that the client wants to delete.
- **ServiceAccessPoint**: NULL, which means the group is on the local system.
- **RemoveElements**: The client can request to delete the group even if it is not empty.
    - True - delete the group regardless of whether it is empty
    - False - delete the group ONLY IF it is empty

AddMembers

```
uint32 ReplicationService.AddMembers(
[IN]  CIM_LogicalElement REF Members[],
[IN, Required]  SNIA_ReplicationGroup REF ReplicationGroup,
[IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint );
```

Use to add members to an existing replication group. Parameters are:

- **Members[]**: An array of strings containing object references to the new volumes to add to the replication group. The new elements are added at the end of current members of the replication group. Duplicate members are not allowed.
- **ReplicationGroup**: A reference to an existing replication group.
- **ServiceAccessPoint**: NULL, which means the group is on the local system.

RemoveMembers

```
uint32 ReplicationService.RemoveMembers(
[IN]  CIM_LogicalElement REF Members[],
[IN]  boolean DeleteOnEmptyElement,
[IN, Required]  SNIA_ReplicationGroup REF ReplicationGroup,
[IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint );
```

Use this method to remove members from an existing replication group. Parameters are:

- **Members[]**: An array of strings containing object references to the elements to remove from the replication group. Attempting to remove a member that is not in the replication group returns an error.
- **DeleteOnEmptyElement**: If true and removal of the members causes the group to become empty, the group will be deleted. If this parameter is not NULL, it overrides the group's property DeleteOnEmptyElement.
    - True - if the deleted member is the last element of the group, also delete the group itself
    - False - if the deleted member is the last element of the group, do not delete the group itself
- **ReplicationGroup**: A reference to an existing replication group.
- **ServiceAccessPoint**: NULL, which means the group is on the local system.

CreateElementReplica

```
   uint32 ReplicationService.CreateElementReplica(
      [IN]  string ElementName,
      [IN, Required]  uint16 SyncType,
      [IN]  uint16 Mode,
      [IN, Required]  CIM_LogicalElement REF SourceElement,
```

```
[IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
[IN, OUT]  CIM_LogicalElement REF TargetElement,
[IN]  CIM_ServiceAccessPoint REF TargetAccessPoint,
[IN, EmbeddedInstance("SNIA_ReplicationSettingData")]
     string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[OUT] CIM_Synchronized REF Synchronization,
[IN]  CIM_SettingData REF TargetSettingGoal,
[IN]  CIM_ResourcePool REF TargetPool,
[IN]  uint16 WaitForCopyState);
```

Use to create (or start a job to create) a new storage object which is a replica of the specified source storage object (SourceElement). Parameters are:

- **ElementName**: For virtual copy, this is the end user relevant name for the element being created. The value will be stored in the 'ElementName' property for the created element. This is NULL for physical copy.
- **SyncType**: Describes the type of copy that will be made. For example, Mirror, Snapshot, and Clone.
- **Mode**: Describes whether the target elements will be updated synchronously or asynchronously.
- **SourceElement**: The source storage object which may be a StorageVolume
- **SourceAccessPoint**: NULL
- **TargetElement**:
  - As an input, refers to a target element to use for a physical copy. This is NULL for virtual copy.
  - As an output, refers to the created target storage element (i.e., the replica). If a job is created, the target element may not be available immediately.
- **TargetAccessPoint**: NULL
- **ReplicationSettingData**: NULL, as a client cannot modify the replication behavior of a copy operation
- **Job**: If a Job is created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter (may be NULL if job is completed).
- **Synchronization**: Refers to the created association between the source and the target element. If a job is created, this parameter may be NULL, unless the association is actually formed.
- **TargetSettingGoal**: The definition for the StorageSetting to be maintained by the target storage object (the replica). This is valid only for snapshot volumes (base ID, expiration timer, retention timer).
- **TargetPool**: The underlying storage for the target element (the replica) will be drawn from TargetPool if specified, otherwise the allocation is implementation specific. This is valid only for snapshot volumes, and if specified, this must be the DeltaReplicaStoragePool that is associated with the source StorageVolume.
- **WaitForCopyState**: Before returning, the method shall wait until this CopyState is reached.

Table 90 Parameter Matrix for CreateElementReplica

What to Create:	Physical copy	Virtual copy (snapshot volume)
Sync Type:	Clone	Snapshot
Mode:	Optional (default is Synchronous)	Optional (default is Synchronous

Table 90 Parameter Matrix for CreateElementReplica *(continued)*

ElementName:	Optional (applicable only if TargetElement is not supplied; if NULL, a name will be auto-generated)	Optional (if NULL, a name will be auto-generated)
SourceElement:	Mandatory	Mandatory
TargetElement:	Optional (if NULL, the target volume will be created)	NULL
Output Job?	Y	N
WaitForCopyState:	Synchronized – an independent clone will be created. Inactive – the clone will retain association with the parent	If non-NULL, then only 'Synchronized' is supported

CreateGroupReplica

```
uint32 ReplicationService.CreateGroupReplica(
    [IN]   string RelationshipName,
    [IN, Required]   uint16 SyncType,
    [IN]   uint16 Mode,
    [IN]   SNIA_ReplicationGroup REF SourceGroup,
    [IN]   CIM_LogicalElement REF SourceElement,
    [IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
    [IN, OUT]   SNIA_ReplicationGroup REF TargetGroup,
    [IN]   uint64 TargetElementCount,
    [IN]   CIM_ServiceAccessPoint REF TargetAccessPoint,
    [IN]   uint16 Consistency,
    [IN, EmbeddedInstance("SNIA_ReplicationSettingData")]
         string ReplicationSettingData,
    [IN]   CIM_ConcreteJob REF Job,
    [OUT]  CIM_Synchronized REF Synchronization,
    [IN]   CIM_SettingData REF TargetSettingGoal,
    [IN]   CIM_ResourcePool REF TargetPool,
    [IN]   uint16 WaitForCopyState);
```

Use to create (or start a job to create) a new group of storage objects which are replicas of the specified source storage or a group of source storage objects (SourceElements), i.e., replications of vv sets. Parameters are:

- **RelationshipName**: A mandatory user relevent name for the relationship between the source and target groups or between a source element and a target group (i.e. one-to-many). For SV, each of the name of the individual target volumes (ElementName) would be constructed using RelationshipName as prefix followed by \"_n\" sequence number, where n is a number beginning with 1.
- **SyncType**
- **Mode**
- **SourceGroup**: A group of source storage objects which may be a StorageVolume or storage object. If this parameter is not supplied, SourceElement is required. Both SourceGroup and SourceElement shall not be supplied.
- **SourceElement**: This is not supported and should be NULL.
- **SourceAccessPoint**: NULL

- **TargetGroup**:
  - As an input, refers to a target group to use.
  - As an output, refers to the created target group (i.e., the replica group). If a job is created, the target group may not be available immediately. If TargetGroup is supplied, TargetElementCount shall be NULL.
- **TargetElementCount**: This parameter applies to one-source-to-many-target elements. If TargetGroup is supplied, this parameter shall be NULL. For us this is always NULL.
- **TargetAccessPoint**: NULL
- Consistency: This parameter overrides the default group consistency. For example, "No Consistency", "Sequential Consistency". For us this can only be "Sequential Consistency".
- **ReplicationSettingData**: NULL
- **Job**: For physical copy of vv sets, InFormOS actually spawns one task for each of the vv pair in the set and not one task for the whole set. However, this job parameter can only contain reference to one job, so as a compromise, only the job of the first vv pair copy will be returned.
- **Synchronization**: Refers to the created association between the source element (or source replication group) and the target replication group. InFormOS does not have the capability to keep track of which vv set is synchronized with which other vv set; only individual sync-ed vv pair is known. Therefore, as a compromise, only the StorageSynchronized association of the first vv pair in the vv set will be returned.
- **TargetSettingGoal**: NULL
- **TargetPool**: NULL
- **WaitForCopyState**

Table 91 Parameter Matrix for CreateGroupReplica

What to Create:	Physical copy	Virtual copy (snapshot volume)
Sync Type:	Clone	Snapshot
Mode:	Optional (default is Synchronous)	Optional (default is Synchronous)
RelationshipName:	NULL	Mandatory
SourceGroup:	Mandatory	Mandatory
TargetGroup:	Mandatory	NULL
Output Job?	Y	N
WaitForCopyState:	Synchronized – independent clones will be created Inactive – the clones will retain association with the parent	If non-NULL, then only 'Synchronized' is supported

TPD_CreateGroupReplica

Due to the inadequacies of the SMI-S defined CreateGroupReplica in supporting vv set replication and consistent volume list replication, a couple of proprietary methods are defined.

```
uint32 ReplicationService.TPD_CreateGroupReplica(
   [IN]   string RelationshipName,
   [IN, Required]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   SNIA_ReplicationGroup REF SourceGroup,
   [IN, OUT]   SNIA_ReplicationGroup REF TargetGroup,
   [OUT]   CIM_ConcreteJob REF Jobs[],
   [OUT]  CIM_Synchronized REF Synchronizations[]);
```

Parameters are:

- **Jobs**: a list of jobs as a result of physical copy of vv set
- **Synchronizations**: a list of StorageSynchronized between each of the members of the source and target vv set

TPD_CreateConsistentReplicaList

```
uint32 ReplicationService.TPD_CreateConsistentReplicaList(
   [IN, Required]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   TPD_StorageVolume REF SourceElements[],
   [IN]   string TargetNames[],
   [IN, OUT]  TPD_StorageVolume REF TargetElements[],
   [IN]   CIM_SettingData REF TargetSettingGoal,
   [IN]   CIM_ConcreteJob REF Jobs[],
   [OUT]  CIM_Synchronized REF Synchronizations[]);
```

- **SyncType**
- **Mode**
- **SourceElements**: array of references to source volumes to replicate. Number of elements in this array must be the same as that of TargetNames or TargetElements.
- **TargetNames**: array of names, matched with SourceElements array, of the resulting target snapshot volumes. Valid only for virtual copy. This and TargetElements cannot both be specified.
- **TargetElements**: array of references to target volumes, matched with SourceElements. Valid only for physical copy. This and TargetNames cannot both be specified
- **TargetSettingGoal**: The definition for the StorageSetting to be maintained by the target storage object (the replica). This is valid only for snapshot volumes (base ID, expiration timer, retention timer).
- **Jobs**: a list of jobs as a result of physical copy
- **Synchronizations**: a list of StorageSynchronized between each of the source and target vv pair in the list

This is a 3PAR-specific method which is equivalent to creategroupvvcopy and creategroupsv commands, and allows for client to specify a list of volumes to copy instead of a single one (as is the case with CreateElementReplica) or a replica group (as is the case with CreateGroupReplica).

Table 92 Parameter Matrix for CreateConsistentReplicaList

What to Create:	Consistent list of async physical copies	Consistent list of independent physical copies	Consistent list of virtual copies
CLI equivalence:	`creategroupvvcopy -p -s <parent_VV>:<dest_VV>...`	`creategroupvvcopy -p <parent_VV>:<dest_VV>...`	`creategroupsv`
SyncType:	Clone	Clone	Snapshot
Mode:	Synchronous	Synchronous	Synchronous
TargetNames:	NULL	NULL	Mandatory
SourceGroups:	Mandatory	Mandatory	Mandatory
TargetGroups:	Mandatory	Mandatory	NULL

Table 92 Parameter Matrix for CreateConsistentReplicaList *(continued)*

Output Job?	Y	Y	N
WaitForCopyState	Inactive – the clones will retain association with the parent	Synchronized – independent clones will be created	If non-NULL, then only Synchronized is supported

ModifyReplicaSynchronization

```
uint32 ReplicationService.ModifyReplicaSynchronization(
    [IN, Required]   uint16 Operation,
    [IN, Required]   CIM_Synchronized REF Synchronization,
    [IN, EmbeddedInstance ( "SNIA_ReplicationSettingData" )]
           string ReplicationSettingData,
    [IN]   CIM_StorageSynchronized REF SyncPair[],
    [OUT]  CIM_ConcreteJob REF Job,
    [IN]   boolean Force,
    [OUT]  CIM_SettingsDefineState REF SettingsState,
    [IN]   uint16 WaitForCopyState);
```

Use to modify (or start a job to modify) the synchronization association between two storage objects or replication groups. Parameters are:

- **Operation**: This parameter describes the type of modification to be made to the replica and/or to the related associations
- **Synchronization**: The reference to the replication association describing the elements/groups relationship that is to be modified.
- **ReplicationSettingData**: NULL
- **SyncPair[]**: This parameter applies to AddSyncPair/RemoveSyncPair Operations. It allows a client to form a StorageSynchronized association between source and target elements and then add the association to existing source and target groups. Alternatively, a client can remove a StorageSynchronized association from source and target groups. We do not support modification of group replicas, so this should always be NULL.
- **Job**
- **SettingsState**: Reference to the association between the source or group element and an instance of SynchronizationAspect. This should always be NULL since we do not support SynchronizationAspect.
- **Force**: Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible. This should always be NULL since we do not support this.
- **WaitForCopyState**

Table 93 Parameter Matrix for ModifyReplicaSynchronization

Operation	ReplicationService Method
Resync physical copy	ModifyReplicaSynchrnonization • Operation = Resync Replica (14)
Promote physical copy	ModifyReplicaSynchrnonization • Operation = Dissolve (9)
Halt physical copy	ModifyReplicaSynchrnonization • Operation = Abort (2)
Promote sv	ModifyReplicaSynchrnonization • Operation = Restore from Replica (15)
Delete the copy	ModifyReplicaSynchrnonization • Operation = Return to Resource Pool (19)

ModifyListSynchronization

```
uint32 ReplicationService.ModifyListSynchronization(
    [IN, Required]   uint16 Operation,
    [IN, Required]   CIM_Synchronized REF Synchronization[],
    [IN, EmbeddedInstance ( "SNIA_ReplicationSettingData" )]
          string ReplicationSettingData,
    [OUT]  CIM_ConcreteJob REF Job,
    [IN]   boolean Force,
    [IN]   uint16 WaitForCopyState);
```

- **Operation**: This parameter describes the type of modification to be made to the replica and/or to the related associations.
- **Synchronization**: An array of references to the replication association describing the elements/groups relationship that is to be modified. All elements of this array shall have the same SyncType, the same Mode, and the Operation must be valid for the ReplicationType -- SyncType, Mode, Local/Remote.
- **ReplicationSettingData**: NULL
- **Job**
- **Force**: Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible. This shall always be NULL since we do not support this.
- **WaitForCopyState**

This is similar to ModifyReplicaSynchronization, but allows for operation on multiple replica pairs in one request.

TPD_ModifyGroupReplicaSynchronization

```
uint32 ReplicationService.TPD_ModifyGroupReplicaSynchronization(
    [IN, Required]   uint16 Operation,
    [IN, Required]   TPD_ReplicaGroup REF TargetGroup,
    [OUT]  CIM_ConcreteJob REF Jobs[]);
```

- **Operation**: This parameter describes the type of modification to be made to the replica and/or to the related associations.
- **TargetGroup**: Reference to the replication group (vvset)
- **Jobs**: Array of references to ConcreteJob associated with each of the volume in ReplicaGroup.

This is a 3PAR-specific method used for resync and promotes of physical copy vvset or promote of snapshot vvset.

GetAvailableTargetElements

```
uint32 ReplicationService.GetAvailableTargetElements(
   [IN, Required]   CIM_LogicalElement REF SourceElement,
   [IN, Required]   uint16 SyncType,
   [IN, Required]   uint16 Mode,
   [IN, EmbeddedInstance ( "SNIA_ReplicationSettingData" )]
       string ReplicationSettingData,
   [IN]   CIM_SettingData REF TargetSettingGoal,
   [IN]   CIM_ResourcePool REF TargetPools[],
   [IN]   CIM_ServiceAccessPoint REF TargetAccessPoint,
   [OUT]  CIM_ConcreteJob REF Job,
   [OUT]  CIM_LogicalElement REF Candidates[] );
```

Use to get all of the candidate target elements for the supplied source element. Parameters are:

- **SourceElement**: The source StorageVolume.
- **SyncType**: Snapshot or Clone
- **Mode**: Synchroous or Asynchronous
- **ReplicationSettingData**: The parameter is useful for requesting a specific combination of thinly and fully provisioned elements. For us this will be NULL.
- **TargetSettingGoal**: Desired target StorageSetting. If NULL, settings of the source elements shall be used. We do not support this so input should be NULL.
- **TargetPools[]**: The storage pools for the target elements. If specified, this must always be the DeltaReplicaStoragePool associated with the source volume.
- **TargetAccessPoint**: Reference to target access point information. If NULL, only local targets are returned. For us this can only be NULL.
- **Job**: This will always be NULL.
- **Candidates[]**: The list of the candidate target StorageVolume found.

GetReplicationRelationships

```
uint32 ReplicationService.GetReplicationRelationships(
   [IN]   uint16 Type,
   [IN]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   uint16 Locality,
   [IN]   uint16 CopyState,
   [OUT]  CIM_ConcreteJob REF Job,
   [OUT]  CIM_Synchronized REF Synchronizations[] );
```

Use to get all of the synchronization relationships known to the processing replication service. Parameters are:

- **Type**: The type of synchronization relationships, for example, StorageSynchronized or GroupSynchronized. If this parameter is not supplied, all such relationships are retrieved.
- **SyncType**: If this parameter is not supplied, all SyncTypes are retrieved.

- **Mode**: If this parameter is not supplied, all Modes are retrieved.
- **Locality**: Describes the desired locality. If this parameter is not supplied, all replication relationships are retrieved, regardless of the locality of elements. Choices are: Local only – Source and target elements are contained in the same system; and Remote only -- Source and target elements are contained in two different systems.
- **CopyState**: Only retrieve synchronization relationships that currently has this CopyState. If this parameter is not supplied, relationships are retrieved regardless of their current CopyState.
- **Job**: This will always be NULL.
- **Synchronizations[]**: An array of elements found

SUMMARY OF REPLICATIONSERVICE METHODS

The following tables shows a summary of the supported ReplicationService methods and their corresponding CLI commands:

Operation	CLI Command	Which Method to Use (in ReplicationService class, unless specified otherwise)	Output Job(s)?
Create vvset	`createvvset <vvset>`	CreateGroup	N
Remove vvset	`removevvset <vvset>`	RemoveGroup	N
Add to vvset	`createvvset -add <vvset> <vv>`	AddMembers	N
Remove member from vvset	`removevvset <vvset> <vv>`	RemoveMembers	N
Modify vvset	`setvvset`	ReplicaGroup.ModifyInstance	N
Create clone physical copy vv	`createvvcopy -p <par> <dest>`	CreateElementReplica • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Synchronized (4)	Y
Create clone physical copy vvset	`createvvcopy -p set:<par> set:<dest>`	CreateGroupReplica/ TPD_CreateGroupReplica (preferred) • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Synchronized (4)	Y
Create async physical copy vv	`createvvcopy -p <par> -s <dest>`	CreateElementReplica • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Inactive (8)	Y
Create async physical copy vvset	`createvvcopy -p set:<par> -s set:<dest>`	CreateGroupReplica/ TPD_CreateGroupReplica (preferred) • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Inactive (8)	Y

Operation	CLI Command	Which Method to Use (in ReplicationService class, unless specified otherwise)	Output Job(s)?
Resync physical copy vv	`createvvcopy -r <dest>`	ModifyReplicaSynchronization • Operation = Resync Replica (14)	Y
Resync physical copy vvset	createvvcopy -r set:<vvset>	TPD_ModifyGroupReplicaSynchronization • Operation = Resync Replica (14)	Y
Halt physical copy of vv	`createvvcopy -halt <dest>`	ModifyReplicaSynchronization • Operation = Abort (2)	Y
Halt physical copy of vvset	`createvvcopy -halt set:<vvset>`	TPD_ModifyGroupReplicaSynchronization • Operation = Abort (2)	N
Promote physical copy vv	`promotevvcopy <vv>`	ModifyReplicaSynchronization • Operation = Dissolve (9)	N
Promote physical copy vvset	`promotevvcopy set:<vvset>`	TPD_ModifyGroupReplicaSynchronization • Operation = Dissovle (9)	N
Create group physical copy clones	`creategroupvvcopy -p <parent vv>:<dest vv>...`	TPD_CreateConsistentReplicaList • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Synchronized (4)	Y
Create async group physical copies	`creategroupvvcopy -p -s <parent vv>:<dest vv>...`	TPD_CreateConsistentReplicaList • SyncType = Clone (8) • Mode = Synchronous (2) • WaitForCopyState=Inactive (8)	Y
Resync group physical copies	`creategroupvvcopy -r <dest vv> ...`	ModifyListSynchronization • Operation = Resync Replica (14)	Y
Halt group physical copies	`creategroupvvcopy -halt <dest vv>...`	ModifyListSynchronization • Operation = Abort (2)	N
Create snapshot volume	`createsv <name> <parent vv>`	CreateElementReplica • SyncType = Snapshot (7) • Mode = Synchronous (2)	N
Create snapshots from vvset	`createsv <name> set:<vvset>`	CreateGroupReplica/ TPD_CreateGroupReplica (preferred) • SyncType = Snapshot (7) • Mode = Synchronous (2)	N
Promote snapshot volume	`promotesv <vv>`	ModifyReplicaSynchronization • Operation = Restore from Replica (15)	Y
Promote vvset containing snapshot volumes	`promotesv set:<vvset>`	TPD_ModifyGroupReplicaSynchronization • Operation = Restore from Replica (15)	Y
Create group sv	`creategroupsv <parent vv>[:<sv name>]`	TPD_CreateConsistentReplicaList • SyncType = Snapshot (7) • Mode = Synchronous (2)	N

ReplicationServicesCapabilities

This class defines all of the capability properties for the replication services.

Table 94 Relevant Properties of ReplicationServiceCapabilities

Properties	Value	Description
SupportedReplicationTypes	Synchronous Mirror Remote (4), Asynchronous Mirror Remote (5), Synchronous Snapshot Local (6), Synchronous Clone Local (10)	Enumeration indicating the supported SyncType/Mode/ Local-or-Remote combinations.
SupportedStorageObjects	StorageVolume (2)	Enumeration indicating the supported storage objects.
SupportedAsynchronousActions	CreateElementReplica (2), ModifyReplicaSynchronization (5), ModifyListSynchronization (6)	Identify replication methods using job control.
SupportedSynchronousActions	CreateElementReplica (2), CreateGroupReplica (3), ModifyReplicaSynchronization (5), ModifyListSynchronization (6), GetAvailableTargetElements (8), GetReplicationRelationship(10), CreateGroup(12), DeleteGroup(13), AddMembers(14), RemoveMembers(15)	Identify replication methods not using job control.

In addition, this class contains a bunch of extrinsic methods. All methods return one of the following status codes:

- 0: (Job) Completed with no error
- 1: Method not supported
- 4: Failed
- 5: Invalid Parameter
- 4096: Method Parameters Checked - Job Started

Table 95 Alignment of ReplicationServiceCapabilities.SupportedReplicationType and StorageReplicationCapabilities.SupportedSynchronizationType

SupportedReplicationType	SupportedSynchronizationType	Notes
Synchronous Clone Local	UnsyncAssoc - Full	Async physical copy
Synchronous Snapshot Local	UnsyncAssoc - Delta	Snapshot volume
Synchronous Clone Local	UnsyncUnassoc	Independent physical copy

ConvertSyncTypeToReplicationType

```
uint32 ReplicationServiceCapabilities.ConvertSyncTypeToReplicationType(
      [IN]   uint16 SyncType,
      [IN]   uint16 Mode,
      [IN]   uint16 LocalOrRemote,
      [OUT]  uint16 SupportedReplicationTypes);
```

The majority of the methods in this class accept`ReplicationType` which represents a combination of SyncType, Mode, and Local/Remote. This method accepts the supplied information and returns the corresponding ReplicationType, which can be passed to other methods to get the additional capabilities.

The following tables show the values for the `CovertSyncTypeToReplicationType` parameters, which appear in the value maps in the appropriate MOF files.

Table 96 SyncType/Mode/LocalOrRemote and ReplicationTypes Mapping

SyncType	Mode	LocalORRemote	ReplicationType	Notes
Snapshot (7)	Synchronous (2)	Local	Synchronous Snapshot Local (6)	Snapshot Volume
Clone (8)	Synchronous (2)	Local (2)	Synchronous Clone Local (10)	Physical Copy
Mirror (6)	Synchronous (2)	Remote (3)	Synchronous Mirror Remote (4)	
Mirror (6)	Asynchronous (2)	Remote (3)	Asynchronous Mirror Remote (5)	

This method will return Not Supported (2) error code for all other combinations not listed in the table above.

ConvertReplicationTypeToSyncType

```
uint32 ReplicationServiceCapabilities.ConvertReplicationTypeToSyncType(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SyncType,
      [OUT]  uint16 Mode,
      [OUT]  uint16 LocalOrRemote);
```

This method does the opposite of the method `ConvertSyncTypeToReplicationType`. This method translates `ReplicationType` to the corresponding SyncType, Mode, and Local/Remote.

GetSupportedFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedFeatures(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 Features[]);
```

For a given ReplicationType, this method returns the supported features.

Table 97 SupportedFeatures for ReplicationType

ReplicationType	Features	Description
Synchronous Snapshot Local (6)	Replication Groups (2)	Elements in a replication group are supported in a replication operation.
	Targets allocated from Exclusive storage pool (10)	Targets are allocated from exclusive storage pools, i.e., from snap space.
	Target must remain associated to source (16)	A dependent target element must remain associated to source element at all times.
Synchronous Clone Local (10)	"Replication Groups (2)"	
	Targets allocated from Any storage pool (8)	

Table 97 SupportedFeatures for ReplicationType *(continued)*

ReplicationType	Features	Description
	Synchronized clone target detaches automatically (18)	The clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation.
Synchronous Mirror Remote (4)	“Replication Groups” (2)	
Asynchronous Mirror Remote (5)	“Requires full discovery of target ComputerSystem” (6)	Remote ComputerSystem needs to be discovered first before a remote replication can happen.
	“Service suspends source I/O when necessary" (7)	Provider is able to suspend I/O to source elements before splitting the target elements.
	"Targets allocated from Any storage pool" (8)	
	“Target must remain associated to source” (16)	
	"Remote resource requires remote CIMOM” (17)	Client is required to interact with two providers: the provider controlling the source element and the provider controlling the target element

This method will return "Not Supported" (2) error code for any other ReplicationType not listed in the table above.

GetSupportedGroupFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupFeatures(
  [IN]   uint16 ReplicationType,
  [OUT]  uint16 GroupFeatures[]);
```

For a given ReplicationType, this method returns the supported replication group features.

Table 98 SupportedGroupFeatures for ReplicationType

ReplicationType	Features	Description
Synchronous Snapshot Local (6)	Many-to-many replication (3)	One or more elements in the source group and one or more elements in the target group.
	Consistency enabled for all groups (4)	By default, all groups are Consistent.
	Source element can be removed from group (11)	A source element can be removed even when the group is associated with another replication group.
	Group is nameable (14)	A user friendly name can be given to a replication group (ElementName).
Synchronous Clone Local (10)	Many-to-many replication (3)	
	Consistency enabled for all groups (4)	
	Source element can be removed from group (11)	
	Target element can be removed from group (12)	
	Group is nameable" (14)	

Table 98 SupportedGroupFeatures for ReplicationType *(continued)*

ReplicationType	Features	Description
	Synchronized clone target detaches automatically (16)	The clone target group detaches automatically when the target group becomes synchronized; otherwise, the client needs to explicitly request a detach operation.
Synchronous Mirror Remote (4) Asynchronous Mirror Remote (5)	"Consistency enabled for all groups" (4)	
	"Empty replication groups allowed" (5)	
	"Source element can be removed from group" (11)	
	"Target element can be removed from group" (12)	
	"Group is nameable" (14)	

This method will return Not Supported (2) error code for any other ReplicationType not listed in the table above.

GetSupportedCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedCopyStates(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SupportedCopyStates[],
      [OUT]  boolean HostAccessible[]);
```

For a given ReplicationType, this method returns the supported CopyStates and a parallel array to indicate whether for a given CopyState the target element is host accessible or not (true or false).

Table 99 SupportedCopyStates for ReplicationType

ReplicationType	CopyStates	HostAccessible	Description
Synchronous Snapshot Local (6)	Synchronized	True	The copy operation is complete. The target element is an exact replica of the source element.
	Unsynchronized	True	Not all the source element data has been copied to the target element.
	Broken	False	Replica is not a valid view of the source element. OperationalStatus of replica may indicate an Error condition.
Synchronous Clone Local (10)	Initialized	False	
	Synchronized	False	
	Unsynchronized	False	
	Suspended	False	
	Broken	False	
	Inactive	False	
Synchronous Mirror Remote (4) Asynchronous Mirror Remote (5)	Initialized	True	
	Synchronized	True	

Table 99 SupportedCopyStates for ReplicationType *(continued)*

ReplicationType	CopyStates	HostAccessible	Description
	Unsynchronized	True	
	Suspended	True	
	Broken	True	
	Inactive	True	
	FailedOver	True	
	Skewed	True	

This method will return Not Supported (2) error code for any other ReplicationType not listed in the table above.

GetSupportedGroupCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupCopyStates(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SupportedCopyStates[] );
```

For a given ReplicationType, this method returns the supported replication group CopyStates.

Table 100 SupportedGroupCopyStates for ReplicationType

ReplicationType	CopyStates
Synchronous Mirror Remote (4)	Initialized
Asynchronous Mirror Remote (5)	Synchronized
	Unsynchronized
	Suspended
	Broken
	Inactive
	FailedOver
	Skewed
	Mixed

GetSupportedWaitForCopyStates

```
   uint32 ReplicationServiceCapabilities.GetSupportedWaitForCopyStates(
     [IN]   uint16 ReplicationType,
     [IN]   unit16 MethodName,
     [OUT]  uint16 SupportedCopyStates[] );
```

This method, for a given ReplicationType and method, returns the supported CopyStates that can be specified in the method's WaitForCopyState parameter.

Table 101 Possible SupportedCopyStates for Various ReplicationType/MethodName

	Method Name			
ReplicationType	**CreateElement Replica (2)**	**Create GroupReplica (3)**	**ModifyReplicaSynchronization (4)**	**ModifyList Synchronization (5)**
Synchronous Clone Local (10)	Synchronized (4) Inactive (8)	Synchronized (4) Inactive (8)	Synchronized (4)	NULL
Synchronous Snapshot Local (6)	Synchronized (4)	Synchronized (4)	Synchronized (4)	Synchronized (4)

This method will return Not Supported (2) error code for any other ReplicationType.

GetSupportedConsistency

```
uint32 ReplicationServiceCapabilities.GetSupportedConsistency(
  [IN]   uint16 ReplicationType,
  [OUT]  uint16 SupportedConsistency[] );
```

For a given ReplicationType, this method returns the supported Consistency.

This method will always return Sequentially Consistent (2) in the SupportedConsistency parameter array for all three supported replication types. This method will return "Not Supported" (2) error code for any other ReplicationType.

GetSupportedOperations

```
uint32 ReplicationServiceCapabilities.GetSupportedOperations(
  [IN]   uint16 ReplicationType,
  [OUT]  uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported Operations on a StorageSynchronized association that can be supplied to the ModifyReplicaSynchronization method.

Table 102 Possible SupportedOperations for Various ReplicationType

ReplicationType	Supported Operations
Synchronous Clone Local (10)	Abort (2) Detach (8) Dissolve (9) Resync (14)
Synchronous Snapshot Local (6)	Detach (8) Dissolve (9) Restore Replica (15) Return to Resource Pool (19)

This method will return Not Supported (2) error code for any other ReplicationType.

GetSupportedGroupOperations

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupOperations(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication group Operations on a GroupSynchronized association that can be supplied to the ModifyReplicaSynchronization method.

Since remote replication implementation is read-only, no group operations are supported. SupportedOperations for either ReplicationType of Synchronous Mirror Remote (4) or Asynchronous Mirror Remote (5) will be an empty array. This method will return “Not Supported” (2) error code for any other ReplicationType.

GetSupportedListOperations

```
uint32 ReplicationServiceCapabilities.GetSupportedListOperations(
      [IN]   uint16 ReplicationType,
      [IN]   uint16 SynchronizationType,
      [OUT]  uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication Operations on a list of associations that can be supplied to the ModifyListSynchronization method. The parameter SynchronizationType specifies the operations as they apply to a list of StorageSynchronized or GroupSynchronized. If SynchronizationType is not specified, StorageSynchronized is assumed.

Possible SupportedOperations are the same as that for GetSupportedOperations. However, only StorageSynchronized (2) is supported for SynchronizationType. This method will return "Not Supported" (2) error code for any other SynchronizationType or ReplicationType.

GetSupportedSettingsDefineStateOperations

For a given ReplicationType this method returns the supported operations on a SettingsDefineState association that can be supplied to the ModifySettingsDefineState method.

We do not support SettingsDefineState, so this method will always return "Not Supported" (2) error code. FYI, SettingsDefineState marks a physical copy clone as a target even if there is no longer a replication relationship with the source.

GetSupportedThinProvisioningFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedThinProvisioningFeatures(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SupportedThinProvisioningFeatures[] );
```

For a given ReplicationType this method returns the supported features related to thin provisioning. Possible SupportedThinProvisioningFeatures are:

1. Thin provisioning is not supported: Feature is unavailable.
2. Zeros written in unused allocated blocks of target: In copying thin to full, the unused blocks of target will be written with zeros.
3. Unused allocated blocks of target are not initialized: In copying thin to full, the unused blocks of target will remain uninitialized.

For Asynchronous Snapshot Local and Asynchronous Clone Loca" ReplicationType's, the SupportedThinProvisioningFeatures shall contain only 2.

For Synchronous Snapshot Local ReplicationType, the SupportedThinProvisioningFeatures shall not contain anything since the options seem to apply only to physical copy.

GetSupportedMaximum

```
uint32 ReplicationServiceCapabilities.GetSupportedMaximum(
      [IN]   uint16 ReplicationType,
      [IN]   uint16 Component,
      [OUT]  uint64 MaxValue );
```

This method accepts a ReplicationType and a component, it then returns a static numeric value representing the maximum number of the specified component that the service supports. A value of 0 indicates unlimited components of the given type. In all cases the maximum value is bounded

by the availability of resources on the computer system. If the information is not known, the method returns 7, which indicates, Information is not available.

If ReplicationType parameter contains an unsupported value, an error of Not Supported (2) shall be returned.

The following table shows the list of components that can be specified:

Table 103 MaxValue for Various ReplicationType/Component

	ReplicationType			
Component	**Asynchronous Mirror Remote (5)**	**Synchronous Mirror Remote (4)**	**Synchronous Clone Local (10)**	**Synchronous Snapshot Local (6)**
Number of groups	2400 (equals to max mirrored vv per system, assuming 1 vv per group)	800 (equals to max mirrored vv per system, assuming 1 vv per group)	Max # of total source or target elements (assuming 1 vv per group)	MIN(Max # of total source elements, Max # of total target elements) (assuming 1 vv per group)
Number of elements per source group	100	100	Max # of base volume / 2	Max # of base volume
Number of elements per target group	100	100	Max # of base volume / 2	Max # of base volume
Number of target elements per source element	1	1	Max # of base volume – 1	Max # snapshot volume per base volume
Number of total source elements	2400	800	Max # of base volume / 2	Max # of base volume
Number of total target elements	2400	800	Max # of base volume / 2	Max # of volume – Max # of base volume
Number of peer systems	4 (max N-to-1 configuration)	2 (max N-to-1 configuration)	0	0
Number of hops in multi-hop replication	1	1	Max # of base volume – 1	(Max # of RW snapshots * 2) -1

GetDefaultConsistency

```
uint32 ReplicationServiceCapabilities.GetDefaultConsistency(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 DefaultConsistency );
```

This method for a given ReplicationType, returns the default consistency value for the replication groups.

We will always return Sequentially Consistent (2) in DefaultConsistency parameter for all supported ReplicationTypes.

GetDefaultGroupPersistency

```
uint32 ReplicationServiceCapabilities.GetDefaultGroupPersistency(
      [OUT]  uint16 DefaultGroupPersistency );
```

This method returns the default persistency for a newly created group.

We will always return Persistent (2) in DefaultGroupPersistency parameter for all supported ReplicationTypes.

GetSupportedReplicationSettingData

```
      [Description (
          "This method, for a given ReplicationType and a supplied "
          "property, returns an array of supported settings that "
          "can be utilized in an instance of the "
          "ReplicationSettingData class." ),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "0x8000.." },
       Values { "Success", "Not Supported", "Unknown", "Timeout",
          "Failed", "Invalid Parameter", "In Use", "DMTF Reserved",
          "Vendor Specific" }]
   uint32 GetSupportedReplicationSettingData(
         [IN, Description (
             "A value representing the ReplicationType." ),
          ModelCorrespondence {
             "CIM_ReplicationServiceCapabilities.SupportedReplicationTypes" }]
      uint16 ReplicationType,
         [IN, Description (
             "A value representing the property name." ),
          ValueMap { "2", "3", "4", "5", "6", "7", "8", "9",
             "..", "0x8000.." },
          Values { "ConsistentPointInTime",
             "DesiredCopyMethodology", "Multihop",
             "OnGroupOrListError", "UnequalGroupsAction",
             "TargetElementSupplier", "ThinProvisioningPolicy",
             "Pairing", "DMTF Reserved", "Vendor Specific" }]
uint16 PropertyName,
         [OUT, Description (
             "An array containing the supported values that can "
             "be supplied in an instance of a "
             "ReplicationSettingData. Refer to the class "
             "ReplicationSettingData for the possible values for "
             "each property. For boolean data, use the following "
             "data mapping: 2=\"false\", 3=\"true\"." ),
          ModelCorrespondence {
             "CIM_ReplicationSettingData.ConsistentPointInTime",
             "CIM_ReplicationSettingData.DesiredCopyMethodology",
             "CIM_ReplicationSettingData.Multihop",
             "CIM_ReplicationSettingData.OnGroupOrListError",
             "CIM_ReplicationSettingData.Pairing",
             "CIM_ReplicationSettingData.UnequalGroupsAction",
             "CIM_ReplicationSettingData.TargetElementSupplier",
             "CIM_ReplicationSettingData.ThinProvisioningPolicy" }]
uint16 SupportedValues[]);
```

This method, for a given ReplicationType, returns an array of supported settings that can be used in an instance of the ReplicationSettingData class. See the MOF for the ReplicationSettingData class for the value map of the properties.

Table 104 Possible SupportedValues forVarious ReplicationType/PropertyName

	ReplicationType			
PropertyName	**Asynchronous Mirror Remote (5)**	**Synchronous Mirror Remote (4)**	**Synchronous Clone Local (10)**	**Synchronous Snapshot Local (6)**
ConsistentPointInTime (2) *This property applies to a group of elements. If it is true, it means the point-in-time to be created at an exact time with no I/O activities in such a way the data is consistent among all the elements of the group.*	True (3)	True (3)	True (3)	True (3)
DesiredCopyMethodology (3) *Control what copy methodology the service should use. For the most part, the service decides the best methodology based on the SyncType.*	Delta-Update (8) *Difference based replication where initially the source element is copied to the target element. Then, at regular intervals, only changes to the source element that have taken place since the previous copy operation are incrementally updated to the target element.*	Differential-Copy (5) *Only the new writes are copied to the target element.*	Snap-And-Clone (9) *The service creates a snapshot of the source element first, then uses the snapshot as the source of the copy operation to the target element*	Incremental-Copy (4) *Only changed data is copied to the target element.*
Multihop (4) *This property applies to multihop copy operation. It specifies the number of hops the starting source (or group) element is expected to be copied.*	1	1	1	1
OnGroupOrListError (5) *This property applies to group or list operations. It specifies what the implementation should do if an error is encountered before all entries in the group or list is processed.*	Stop (3)	Stop (3)	Stop (3)	Stop (3)
UnequalGroupsAction (6) *Indicates what should happen if the number of elements in source and target groups are not the same. The default is to return an*	Return an Error (2)	Return an Error (2)	Return an Error (2)	Return an Error (2)

Table 104 Possible SupportedValues forVarious ReplicationType/PropertyName *(continued)*

	ReplicationType			
PropertyName	**Asynchronous Mirror Remote (5)**	**Synchronous Mirror Remote (4)**	**Synchronous Clone Local (10)**	**Synchronous Snapshot Local (6)**
error, unless one-to-many replication is supported and there is only one source and more than one target.				
TargetElementSupplier (7) *If target elements are not supplied, this property indicates where the target elements should come from.*	Use Existing (1)	Use Existing (1)	Create New (2)	Create New (2)
ThinProvisioningPolicy (8) *If the target element is not supplied, this property specifies the provisioning of the target element.*	Provisioning of the target same as source (5) [technically target element has to exist]	Provisioning of the target same as source (5) [technically target element has to exist]	Provisioning of the target same as source (5)	Implementation decided provisioning of target (7)
Pairing (9) *Controls how source and target elements are paired.*	Exact order (3)	Exact order (3)	Exact order (3)	Exact order (3)

This method will return Not Supported (2) error code for any other ReplicationType not listed in the table above.

GetDefaultReplicationSettingData

```
uint32 ReplicationServiceCapabilities.GetDefaultReplicationSettingData(
      [IN]   uint16 ReplicationType,
      [OUT, EmbeddedObject]
       string DefaultInstance );
```

This method, for a given ReplicationType, returns the default ReplicationSettingData as an instance.

The output DefaultInstance will be constructed according to the table in the section on GetSupportedReplicationSettingData.

TPD_RemoteStorageSynchronized

TPD_RemoteStorageSynchronized, sub-classed from CIM_StorageSynchronized, represents the remote replication association between each volume pair in the remote replication group pairs. The object path of the StorageVolume or ReplicationEntity of the peer system does not have a valid hostname, as neither the hostname nor IP address of the peer system is known; instead, WWN of the peer system is used.

Table 105 Relevant Properties of RemoteStorageSynchronized

Properties	Description
SystemElement	Source volume; SystemElement and SyncedElement will switch if the role is reversed.
SyncedElement	Target volume; SystemElement and SyncedElement will switch if the role is reversed.
WhenSynced	When remote copy group is last synced. This property is NULL for synchronous mode.
SyncMaintained	CopyState describes the state of the association with respect to Replication activity. Possible values are: • Initialized (2) • Unsynchronized (3) • Synchronized (4) • Broken (5) • Inactive (8) • FailedOver (10) • Skewed (13)
SyncState	SyncState describes the state of the association with respect to Replication activity. Possbile values are: • Initialized (2) • ResyncInProgress (5) • Synchronized (6) • Broken (12) • Frozen (14)
Mode	Mode describes whether the target elements will be updated synchronously or asynchronously. Possible values are: • Synchronous (2) • Asynchronous (3)
PercentSynced	Percent synchronized; applicable only when resyncing.
ProgressStatus	ProgressStatus describes the status of the association with respect to Replication activity. Possible values are: • Completed (2) • Synchronizing (6) • Resyncing (7) • Restoring (8) • Failing Over (11) • Failing Back (12) • Requires Activate (19)
RequestedCopyState	RequestedCopyState is an integer enumeration that indicates the last requested or desired state for the association. The actual state of the association is represented by CopyState. Note that when CopyState reaches the requested state, this property will be set to 'Not Applicable. If CopyState is Broken or Skewed, this property will be set to “Synchronized”.
SyncType	SyncType describes the intended outcome of the replication. For remote replication, this property can only be Mirror (6), which means create and maintain a copy of the source.

Table 105 Relevant Properties of RemoteStorageSynchronized *(continued)*

Properties	Description
CopyType	CopyType describes the Replication Policy. Possible values are: • Async (2) • Sync (3)
ReplicaType	ReplicaType provides information on how the Replica is being maintained. For remote replication, this can only be "Full Copy (2)", which indicates that a full copy of the source object is (or will be) generated.
UndiscoveredElement	This property specifies whether the source, the target, or both elements involved in a copy operation are undiscovered. Possbile values are: • SystemElement (2) • SyncedElement (3) If either SystemElement or SyncedElement is undiscovered, the corresponding property of the association will have object path to a ReplicationEntity instead of StorageVolume. If neither end is undiscovered, then this property is NULL.
SyncInterval	Specifies that groups that are in asynchronous periodic mode should be periodically synchronized in accordance with the specified value of the replication. 0 means synchronization will not happen unless it is done manually.
SyncPolicy	Specifies the policy of the Remote Copy volume groups for dealing with I/O failure and error handling of the replication. Valid values are: • `no_fail_wrt_on_err` - Specifies that if Remote Copy is started for the volume group and a write to the secondary system fails, then the Remote Copy operation is stopped and an I/O error is not returned to the host (default). This allows the application writing the data to continue, but makes the secondary volumes out of date with the primary volumes. • `auto_recover` - Specifies that if the Remote Copy is stopped as a result of the Remote Copy links going down, the group is restarted automatically after the links come back up. If this policy is enabled for a group while the group is stopped after link failures, it will be only be started when the links come up for the failed target. If the links are already up at the time the policy is set then the group will not be restarted at that time. • `no_auto_recover` - Specifies that if the Remote Copy is stopped as a result of the Remote Copy links going down, the group must be restarted manually after the links come back up (default). • `over_per_alert` - If a synchronization of a periodic Remote Copy group takes longer to complete than its synchronization period then an alert will be generated. This is the default behavior. • `no_over_per_alert` - If a synchronization of a periodic Remote Copy group takes longer to complete than its synchronization period then an alert will not be generated.

State Transition During Normal Operation

Table below shows the values of CopyState, SyncState and ProgressStatus for various CLI SyncStatus of the volumes under normal environment. “Normal” means that the CLI Target Status field shows “ready”.

Table 106 Mapping between CopyState/SyncState/ProgressStatus and CLI State

Showrcopy Group Info SyncStatus	CopyState	SyncState	ProgressStatus
New	Initialized	ResyncInProgress	Requires Activate
Syncing	Unsynchronized	ResyncInProgress	Synchronizing
Synced	Synchronized	Synchronized	Completed
NotSynced	Broken	Broken	Completed
Stale	Skewed	Broken	Completed
Stopped	Inactive	Frozen	Dormant

State Transition During Disaster Recovery

If the primary 3PAR array goes down, the CLI Target Status field on the secondary 3PAR array will show `failed`. CopyState on the secondary 3PAR array will show `FailedOver` only if role has been reversed AND the link with original primary 3PAR array is down. A vendor extension Boolean property called RoleReversed specifies whether the role of source and target element has been reversed from its configured one due to the failover. If on a secondary 3PAR array the role is reversed but the link is back up and volumes synced, CopyState will show `Synchronized` and RoleReversed true.

Following is a table of the state transitions during disaster recovery after a primary 3PAR array goes down.

Table 107 CopyState/SyncState/ProgressStatus Transitions During Disaster Recovery

Sequence of Events	Showrcopy Group Info Role/SyncStatus	CopyState	SyncState	ProgressStatus
State of Secondary 3PAR array after Primary 3PAR array goes down	Secondary/Stopped	Broken	Broken	Completed
After setrcopygroup failover is issued on Secondary 3PAR array	Primary-Rev/Stopped	FailedOver	Broken	FailedOver
State of Secondary 3PAR array after setrcopygroup failover is done Primary-Rev/Stopped FailedOver Broken Completed Broken Broken Completed	Primary-Rev/Stopped	FailedOver	Broken	Completed
		Broken	Broken	Completed
State of Primary 3PAR array after it is brought back up	Secondary-Rev/Stopped	Broken	Broken	Completed
State of Secondary 3PAR array after setrcopygroup recover is issued on it	Primary-Rev/Syncing	Unsynchronized	RestoreInProgress	Restoring
State of Primary 3PAR array after	Secondary-Rev/Syncing	Unsynchronized	RestoreInProgress	Restoring

Table 107 CopyState/SyncState/ProgressStatus Transitions During Disaster Recovery *(continued)*

Sequence of Events	Showrcopy Group Info Role/SyncStatus	CopyState	SyncState	ProgressStatus
setrcopygroup recover is issued on the secondary 3PAR array:				
State of Secondary 3PAR array after setrcopygroup recover is done	Primary-Rev/Synced	Synchronized	Synchronized	Completed
State of Secondary 3PAR array if setrcopygroup recover fails	Primary-Rev/Stopped	FailedOver	Broken	Completed
State of Primary 3PAR array after setrcopygroup recover is done	Secondary-Rev/Synced	Synchronized	Synchronized	Completed
State of Primary 3PAR array if setrcopygroup recover fails	Secondary-Rev/Stopped	FailedOver	Broken	Completed
State of Secondary 3PAR array after setrcopygroup restore is issued on it	Primary-Rev/Synced	Synchronized	Synchronized	Failing Back
State of Secondary 3PAR array after setrcopygroup restore is done	Secondary/Synced	Synchronized	Synchronized	Completed
State of Secondary 3PAR array if setrcopygroup restore failed	Primary-Rev/Synced	Synchronized	Synchronized	Completed
State of Primary 3PAR array after setrcopygroup recover is issued on the secondary 3PAR array	Secondary-Rev/Synced	Synchronized	Synchronized	Failing Back
State of Primary 3PAR array if setrcopygroup restore is done	Primary/Synced	Synchronized	Synchronized	Completed
State of Primary 3PAR array if setrcopygroup restore failed	Secondary-Rev/Synced	Synchronized	Synchronized	Completed

Synchronous Long Distance (SLD) Backup Link

For SLD configuration in which a primary 3PAR array copy group has 2 target groups, one to a synchronous secondary 3PAR array and one to an asynchronous periodic secondary 3PAR array, there exists also a backup link between the 2 secondary 3PAR array's (showrcopy Group Info Status field shows “Backup”). Under normal operation this link between the remote replication groups is not active (they only get activated during disaster recovery). However, the StorageSynchronized associations will still be discovered between these 2 groups of volumes. In CLI, they both show up as “Secondary” in the Group Info Role field, but for StorageSynchronized, the volumes in the synchronous 3PAR array (the one with the FC link to the primary) will be picked

as the SystemElement and those in the asynchronous periodic 3PAR array will be picked as SyncedElement. The reason for this is that if the primary 3PAR array goes down, by default the synchronous 3PAR array will assume the primary role.

Under normal operation, this StorageSynchronized will have the following state/status values:

- CopyState – Initialized (2), the link to enable replication is established and source/replica elements are associated, but the data flow has not started.
- SyncState – Initialized (2), the link to enable replication is established and source/replica elements are associated, but the Copy engine has not started.
- ProgressStatus – Dormant (3), indicates data flow is inactive.

TPD_ReplicationEntity

Under certain circumstances after the remote replication link goes down, the remote StorageVolume WWN cannot be calculated. When this happens, the provider will not be able to construct a valid object path for the remote StorageVolume. Instead, SystemElement or the SyncedElement in question will contain a reference to a ReplicationEntity, with an InstanceID of 3PAR:`<remote volume name>`, and the UndiscoveredElement property will be set to either SystemElement or SyncedElement. Similiar to remote StorageVolume, this ReplicationEntity cannot be retrieved via EnumerateInstances/EnumerateInstanceNames.GetInstance.

SyncedElement. Similar to remote StorageVolume, this ReplicationEntity cannot be retrieved via EnumerateInstances/EnumerateInstanceNames.GetInstance.

ReplicationEntity cannot be discovered via EnumerateInstances, EnumerateInstanceNames nor GetInstance. It will appear as an instance as a result of an Associators request, and will have the following value:

Table 108 Relevant Properties of TPD_ReplicationEntity

Property	Description
InstanceID	3PAR:<remote volume name>
Type	StorageVolume (4)
Persistent	Always FALSE, which means that ReplicationEntity will disappear if link is back up and copy is successful.
EntityID	Remote volume name
OtherTypeDescription	NULL

Since there is no full-blown support for ReplicationEntity, in the replication service capabilities the SupportedFeatures property will still include “Requires full discovery of target ComputerSystem“ which indicates that the service does not support undiscovered resources.

TPD_RemoteReplicationGroup

This corresponds to the remote replication group, sub-classed from CIM_ReplicationGroup.

Table 109 Relevant Properties of TPD_RemoteReplicationGroup

Property	Description
InstanceID	3PAR:<remote volume name>
ElementName	Name of the rcopy group
Persistent	Always TRUE.

Table 109 Relevant Properties of TPD_RemoteReplicationGroup *(continued)*

Property	Description
DeleteOnEmptyElement	If true, group will be deleted when the last element is removed. This is always FALSE since an rcopy group can be empty.
DeleteOnUnassociated	If true, group will be deleted when the group is no longer associated with another group. This is always FALSE.

TPD_RemoteGroupSynchronized

This is the synchronization association between the remote copy group pairs, sub-classed from CIM_GroupSynchronized. The RemoteReplicationGroup of the peer system cannot be retrieved via EnumerateInstances, EnumerateInstanceNames or GetInstance. The object path of the RemoteReplicationGroup of the peer system will not have a valid hostname, as neither the hostname nor IP address of the peer system is known to the local provider; instead, target name of the peer system will be used. Target name is an arbitrary name given to the peer system when a user creates a remote copy target. Consult Remote Copy User□s Guide for details.

Table 110 Relevant Properties of RemoteStorageSynchronized

Property	Description
SystemElement	Source RemoteReplicationGroup (rm_natural is TRUE); the „natural□ source group will remain as the SystemElement even after a fail over.
SyncedElement	Target RemoteReplicationGroup (rm_natural is FALSE); the „natural□ target group will remain as the SyncedElement even after a fail over.
WhenSynced	When rcopy group is last synced. This property is NULL for synchronous mode.
SyncMaintained	TRUE only if rcopy and rcopygroup are started.
CopyState	CopyState describes the state of the association with respect to Replication activity. Possible values are: • Initialized (2) • Unsynchronized (3) • Synchronized (4) • Broken (5) • Suspended (9) • FailedOver (10) • Skewed (13) • Mixed (14) If any of the member volumes have different CopyState, this property will be set to Mixed.
Mode	Mode describes whether the target elements will be updated synchronously or asynchronously. Possible values are: • Synchronous (2) • Asynchronous (3)
PercentSynced	Percent of the individual volumes in the group synced.
ProgressStatus	ProgressStatus describes the status of the association with respect to Replication activity. Possible values are: • Completed (2) • Synchronizing (6)

Table 110 Relevant Properties of RemoteStorageSynchronized *(continued)*

Property	Description
	• Resyncing (7) • Restoring (8) • Failing Over (11) • Failing Back (12) • Mixed (14) • Requires Activate (19) If any of the member volumes have different ProgressStatus, this property will be set to Mixed.
RequestedCopyState	RequestedCopyState is an integer enumeration that indicates the last requested or desired state for the association. The actual state of the association is represented by CopyState. Note that when CopyState reaches the requested state, this property will be set to 'Not Applicable. If CopyState is Broken or Skewed, this property will be set to “Synchronized”.
SyncType	SyncType describes the intended outcome of the replication. For remote copy, this property can only be Mirror (6), which means create and maintain a copy of the source.
RelationshipName	A unique identifier for the relationship. This is set to <src group>:<target group>.
ConsistencyEnabled	Set to true if consistency is enabled. This is always TRUE.
ConsistencyType	Indicates the consistency type used by the source and its associated target group. This is always Sequential Consistency (2).
ConsistencyState	Indicates the current state of consistency. This is always Consistent (3).
ConsistencyStatus	Indicates the current status of consistency. Consistency may have been disabled or is experiencing an error condition. This is always Completed (2).

TPD_OrderedMemberOfRemoteReplicationGroup

This is the association between RemoteReplicationGroup and a member StorageVolume, sub-classed from CIM_OrderedMemberOfCollection.

Property	Description
Collection	Reference to a RemoteReplicationGroup
Member	Reference to a member StorageVolume
AssignedSequence (uint64)	Position of the volume in the group

Please note that this association exists only for the RemoteReplicationGroup residing on the local array.

TPD_ReplicationServiceAffectsRemoteReplicationGroup

This is the association between ReplicationService and the RemoteReplicationGroup, sub-classed from CIM_ServiceAffectsElement.

Property	Description
AffectingElement	Reference to ReplicationService
AffectedElement	Reference to a RemoteReplicationGroup

Please note that this association exists only for the RemoteReplicationGroup residing on the local array.

TPD_HostedRemoteReplicationGroup

This is the association between ComputerSystem and the RemoteReplicationGroup, sub-classed from CIM_HostedCollection.

Property	Description
Antecedent	Reference to TPD_StorageSystem
Dependent	Reference to a RemoteReplicationGroup

Please note that this association exists only for the RemoteReplicationGroup residing on the local array.

TPD_SAPAvailableForRemoteReplicaVolume

This is the association between StorageVolume and the SCSIProtocolEndpoint, sub-classed from CIM_SAPAvailableForElement.The endpoints denotes the port used for remote replication.

Property	Description
ManagedElement	Reference to StorageVolume
AvailableSAP	Reference to: • TPD_SCSIProtocolFCEndpoint if RCFC • TPD_IPProtocolEndpoint if RCIP

Please note that this association exists only for the StorageVolumes in a RemoteReplicationGroup that are residing on the local array.

TPD_SAPAvailableForRemoteReplicationGroup

This is the association between RemoteReplicationGroup and the SCSIProtocolEndpoint, sub-classed from CIM_SAPAvailableForElement.The endpoints denotes the port used for remote replication.

Property	Description
ManagedElement	Reference to RemoteReplicationGroup
AvailableSAP	Reference to: • TPD_SCSIProtocolFCEndpoint if RCFC • TPD_IPProtocolEndpoint if RCIP

Please note that this association exists only for the RemoteReplicationGroup residing on the local array.

Thin Provisioning Profile

Support for Thin Provisioning objects such as TPD_DynamicStoragePool and thinly provisioned TPD_StorageVolume are described in the Block Services Package chapter.

5 CIM Indications

SMI-S provides for asynchronous notification of events that indicate changes in the CIM server or the managed elements controlled by the CIM server. CIM indications are the mechanism for delivery of such events. A CIM client must subscribe to indications it wants to receive event notifications from the CIM server. For detailed information regarding Indications, refer to SMI S at www.snia.org.

Fibre Channel Ports

The CIM Server currently supports indication subscriptions for changes in the operational status of Fibre Channel ports. Clients must send the following query string to the CIM Server:

```
SELECT * FROM CIM_InstModification WHERE SourceInstance = CIM_FCPort
```

Job Control for Copy Services

The CIM Server supports indication subscriptions for changes in the operational status of a concrete job related to Copy Services. Clients must send the following query string to the CIM Server:

```
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob
```

Thin Provisioning

The CIM Server supports indication subscriptions for various thin provisioning related alert indications. Clients may send the following query string to the CIM Server to receive all of the alert indications:

```
SELECT * FROM CIM_AlertIndication
```

As an alternative, the client can send any of the following to choose the specific indication(s) to receive:

```
SELECT * FROM TPD_StoragePoolGrowWarningAlert
SELECT * FROM TPD_StoragePoolGrowLimitAlert
SELECT * FROM TPD_StoragePoolGrowFailureAlert
SELECT * FROM TPD_StoragePoolOverUsedWarningAlert
SELECT * FROM TPD_StorageVolumeAllocationFailureAlert
SELECT * FROM TPD_StorageVolumeAllocationWarningAlert
SELECT * FROM TPD_StorageVolumeAllocationLimitAlert
SELECT * FROM TPD_DeltaReplicaStoragePoolGrowWarningAlert
SELECT * FROM TPD_DeltaReplicaStoragePoolGrowFailureAlert
SELECT * FROM TPD_StoragePoolCapacityClearAlert
SELECT * FROM TPD_StorageVolumeCapacityClearAlert
SELECT * FROM TPD_DeltaReplicaStoragePoolCapacityClearAlert
```

6 CIM-API Extensions

Health Management

The CIM Server provides the ability to retrieve health information about the physical devices that comprise the HP 3PAR Storage System. This can be broken down into the following areas:

- The controller node Subsystem
- The Disk Enclosure Subsystem
- Power and Cooling

Controller Node Subsystem

The CIM Server provides the ability to manage the health of controller nodes and all of the components that comprise the controller node subsystem portion of the storage system.

controller node Subsystem CIM Classes

The CIM Server provides the ability to manage the health of controller nodes and all of the components that comprise the controller node subsystem portion of the storage system.

Table 111 controller node Subsystem CIM Key Classes

Class	Description
TPD_StorageSystem	The Storage array.
TPD_NodeSystem	Provides information about the controller nodes that make up the storage system.
TPD_NodePackage	Physical aspects of a controller node. This class is used to gather information about all associated physical components of a Node System.
TPD_PCICard	Provides information about any PCI Cards installed in a controller node.
TPD_NodeCPU	Provides information about the CPU(s) contained within each controller node.
TPD_PhysicalMemory	Provides information about the physical memory contained within each controller node.
TPD_IDEDrive	Provides information about the onboard IDE drive contained within each controller node.

Properties for TPD_StorageSystem

Table 112 TPD_StorageSystem Properties

Property	Description
Name	The node WWN for the storage system.
ElementName	User friendly-name of the storage system.
OperationalStatus	Overall status of the storage system. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `StorageSystem`.

Properties for TPD_NodeSystem

Table 113 TPD_NodeSystem Properties

Property	Description
Position	The position of the controller node in the storage system (i.e., node number).
ElementName	User friendly-name of the controller node.
OperationalStatus	Overall status of the controller node. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `NodeSystem`.

Properties for TPD_PCICard

Table 114 TPD_PCICard Properties

Property	Description
Slot	The position/CPU number of this component.
OperationalStatus	Overall status of the CPU. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the CPU.

Properties for TPD_NodeCPU

Table 115 TPD_NodeCPU Properties

Property	Description
Position	The position/CPU number of this component.
OperationalStatus	Overall status of the CPU. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the CPU.

Properties for TPD_PhysicalMemory

Table 116 TPD_PhysicalMemory Properties

Property	Description
Slot	Slot number in which this memory card is inserted.
SlotID	Connector ID where this card is inserted (such as J1101).
OperationalStatus	Overall status of the memory card. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the memory card.

Properties for TPD_IDEDrive

Table 117 TPD_IDEDrive Properties

Property	Description
Position	The position number of this component.
OperationalStatus	Overall status of the IDE Drive. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the IDE Drive.

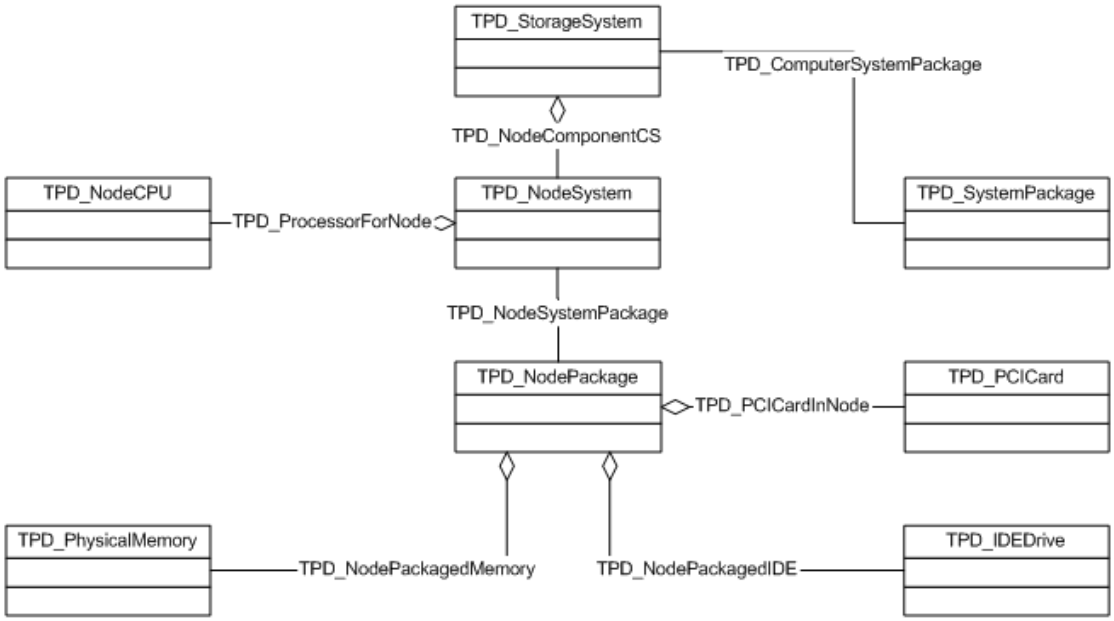
Supported Methods

Table 118 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 11 controller node Subsystem



Disk Enclosure Subsystem

The CIM Server provides the ability to manage the health of physical devices that comprise the backend Disk Enclosure portion of the storage system.

Disk Enclosure Subsystem CIM Classes

Table 119 Disk Enclosure Subsystem Key Classes

Class	Description
TPD_StorageSystem	The storage array.
TPD_DriveCage	Provides information about storage system drive cages.
TPD_CageInterfaceCard	Provides information about the Fibre Channel interface cards contained within a `TPD_DriveCage`.
TPD_Magazine	Provides information about drive magazines contained within a `TPD_DriveCage`.
TPD_DiskDrive	Provides information about disk drives.
TPD_DiskDrivePackage	Physical aspects of a disk drive. This class is used to gather information about the association of drive magazines to disk drives.

Properties for TPD_StorageSystem

Table 120 TPD_StorageSystem Properties

Property	Description
Name	The node WWN for the storage system.
ElementName	User-friendly name of the storage system.
OperationalStatus	Overall status of the storage system. Refer to the SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `StorageSystem`.

Properties for TPD_DriveCage

Table 121 TPD_DriveCage Properties

Property	Description
Position	The position of the drive cage in the storage system (i.e., cage number).
ElementName	User-friendly name of the drive cage.
OperationalStatus	Overall status of the drive cage. Refer to SMI-S for possible values for this property.
OtherOperationalStatus	3PAR-specific operational status for the `DriveCage`.

Properties for TPD_CageInterfaceCard

Table 122 TPD_CageInterfaceCard Properties

Property	Description
Position	The position of the `CageInterfaceCard` in the `DriveCage`
OperationalStatus	Overall status of the `CageInterfaceCard`. Refer to SMI-S for possible values of this property.

Properties for TPD_Magazine

Table 123 TPD_Magazine Properties

Property	Description
ElementName	User-friendly name of the drive magazine. The format vary depending on type of drive cage.
Position	The position of this drive magazine in the drive cage.
OperationalStatus	Overall status of the drive magazine. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the Magazine.

Properties for TPD_DiskDrive

Table 124 TPD_DiskDrive Properties

Property	Description
ElementName	User-friendly name of the disk drive. The format varies depending on the type of the drive cage.
Name	The Fibre Channel WWN of this disk drive.

Table 124 TPD_DiskDrive Properties *(continued)*

Property	Description
Position	The position of this disk drive in the drive magazine.
ID	ID assigned to this Disk Drive by the system.
OperationalStatus	Overall status of the disk drive. Refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `DiskDrive`.

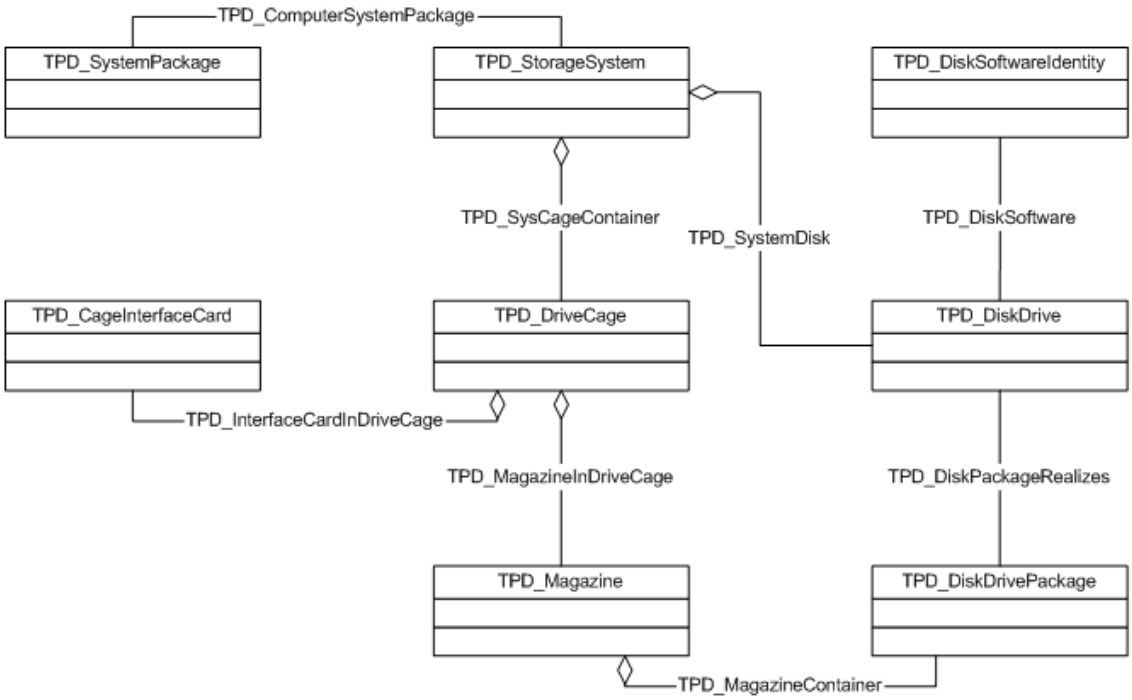
Supported Methods

Table 125 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 12 UML Diagram



Power and Cooling

The CIM Server provides information about the devices used to power and cool the storage system.

Power and Cooling CIM Classes

Table 126 Power and Cooling Key CIM Classes

Class	Description
TPD_StorageSystem	The storage array.
TPD_NodeSystem	Provides information about the controller nodes that make up the storage system.

Table 126 Power and Cooling Key CIM Classes *(continued)*

Class	Description
TPD_DriveCage	Provides information about storage system drive cages.
TPD_PowerSupply	Provides information about power supplies for both controller nodes and disk cages.
TPD_Battery	Provides information about batteries attached to controller node power supplies.
TPD_Fan	Provides information about fans, including fans within each power supply and external fans connected to controller nodes.

Properties for TPD_StorageSystem

Table 127 TPD_StorageSystem Properties

Property	Description
Name	The node WWN for the storage system.
ElementName	User-friendly name of the storage system.
OperationalStatus	Overall status of the `StorageSystem`. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `StorageSystem`.

Properties for TPD_NodeSystem

Table 128 TPD_NodeSystem Properties

Property	Description
Position	The position of the controller node in the storage system (such as the node number).
ElementName	User-friendly name of the controller node.
OperationalStatus	Overall status of the controller node. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `NodeSystem`.

Properties for TPD_DriveCage

Table 129 TPD_DriveCage Properties

Property	Description
Position	The position of the `DriveCage` in the storage system (such as the cage number).
ElementName	User-friendly name of the `DriveCage`.
OperationalStatus	Overall status of the `DriveCage`. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `DriveCage`.

Properties for TPD_PowerSupply

Table 130 TPD_PowerSupply Properties

Property	Description
Position	The position of the `PowerSupply` in the storage system (such as the PS number).
OperationalStatus	Overall status of the `PowerSupply`. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the `PowerSupply`.
ACStatus	Provides information about the AC status of a power supply.
DCStatus	Provides information about the DC status of a power supply.

Properties for TPD_Battery

Table 131 TPD_Battery Properties

Property	Description
Position	The position of the Battery in the storage system (such as the battery number).
OperationalStatus	Overall status of the Battery. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the Battery.
BatteryStatus	Detailed status information about the Battery.

Properties for TPD_Fan

Table 132 TPD_Fan Properties

Property	Description
Position	The position of the Fan in the `PowerSupply` or Storage System (such as the fan number).
OperationalStatus	Overall status of the Fan. Please refer to SMI-S for possible values of this property.
OtherOperationalStatus	3PAR-specific operational status for the Fan.

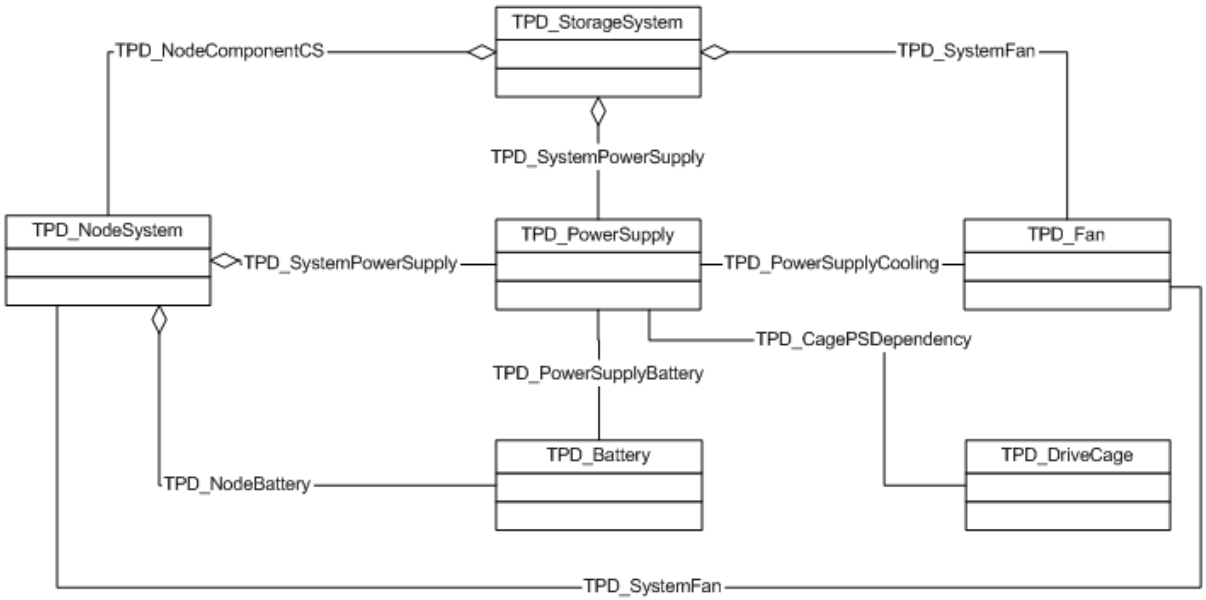
Supported Methods

Table 133 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 13 Power and Cooling



Inventory Management

The CIM Server provides the ability to retrieve inventory information for controller nodes and all of the components that comprise the controller node Subsystem portion of the storage system.

Controller Node Subsystem

The CIM Server provides the ability to retrieve inventory information about the physical devices that comprise the storage system. This can be broken down into the following areas:

- The controller node Subsystem
- The Disk Enclosure Subsystem
- Power and Cooling

Controller Node Subsystem CIM Classes

Table 134 Controller Node Subsystem CIM Key Classes

Class	Description
TPD_StorageSystem	The Storage array.
TPD_NodeSystem	Provides information about the controller nodes that make up the storage system.
TPD_NodePackage	Physical aspects of a controller node. This class is used to gather information about all associated physical components of a Node System.
TPD_PCICard	Provides information about any PCI Cards installed in a controller node.
TPD_NodeCPU	Provides information about the CPU(s) contained within each controller node.

Table 134 Controller Node Subsystem CIM Key Classes *(continued)*

Class	Description
TPD_PhysicalMemory	Provides information about the physical memory contained within each controller node.
TPD_IDEDrive	Provides information about the onboard IDE drive contained within each controller node.

Properties for TPD_StorageSystem

Table 135 TPD_StorageSystem Properties

Property	Description
Name	The Node WWN for the storage system.
ElementName	User friendly-name of the storage system.

Properties for TPD_SystemPackage

Table 136 TPD_SystemPackage Properties

Property	Description
Manufacturer	Manufacturer of the `StorageSystem` (HP 3PAR).
Model	Model of the `StorageSystem`.
SerialNumber	Serial number of the `StorageSystem`.
Version	Version of the software running on the `StorageSystem`.

Properties for TPD_NodeSystem

Table 137 TPD_NodeSystem Properties

Property	Description
Position	The position of the controller node in the storage system (such as the node number).
ElementName	User friendly name of the controller node.

Properties for TPD_NodePackage

Table 138 TPD_NodePackage Properties

Property	Description
Manufacturer	Manufacturer of the controller node.
Model	Model of the controller node.
SerialNumber	Serial number of the controller node.

Properties for TPD_PCICard

Table 139 TPD_PCICard Properties

Property	Description
Slot	Slot number in which this card is inserted.
Manufacturer	Manufacturer of the PCI Card.
Model	Model of the PCI Card.

Table 139 TPD_PCICard Properties *(continued)*

Property	Description
SerialNumber	Serial number of the PCI Card.
Revision	The revision of the PCI Card.
FirmwareVersion	Revision of the firmware running on the card.

Properties for TPD_NodeCPU

Table 140 TPD_NodeCPU Properties

Property	Description
Position	The position/CPU number of this component.
Manufacturer	Manufacturer of the CPU.
Model	Model number for the CPU.
SerialNumber	Serial number for the CPU.

Properties for TPD_PhysicalMemory

Table 141 TPD_PhysicalMemory Properties

Property	Description
Slot	Slot number where this memory card is inserted.
SlotID	Connector ID where this card is inserted (such as slot J1101).
Manufacturer	Manufacturer for the memory card.
PartNumber	Part number for the memory card.
SerialNumber	Serial number for the memory card.

Properties for TPD_IDEDrive

Table 142 TPD_IDEDrive Properties

Property	Description
Position	The position number of this component.
Manufacturer	Manufacturer of the drive.
Model	Model number for the drive.
SerialNumber	Serial number for the drive.
FwRevision	Revision of the firmware running on the drive.

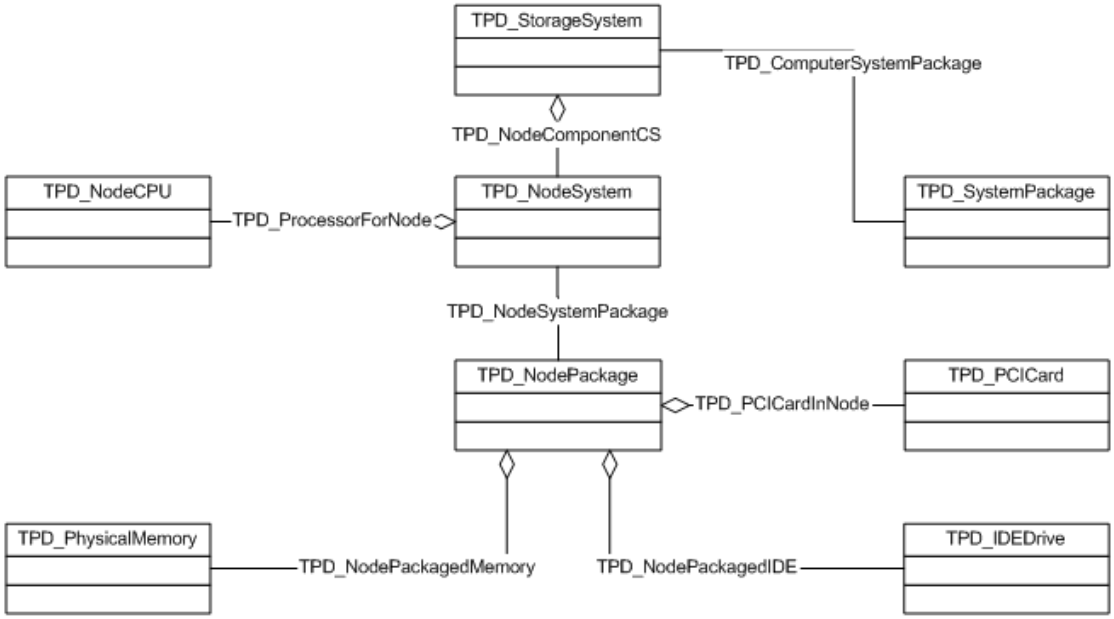
Supported Methods

Table 143 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 14 UML Diagram



Disk Enclosure Subsystem

The CIM Server provides the ability to retrieve inventory information for all of the components that comprise the backend Disk Enclosure portion of the storage system.

Disk Enclosure Subsystem CIM Classes

Table 144 Disk Enclosure Subsystem Key Classes

Class	Description
TPD_StorageSystem	The storage array.
TPD_DriveCage	Provides information about storage system drive cages.
TPD_CageInterfaceCard	Provides information about the Fibre Channel interface cards contained within a `TPD_DriveCage`.
TPD_Magazine	Provides information about drive magazines contained within a `TPD_DriveCage`.
TPD_DiskDrive	Provides information about disk drives.
TPD_DiskDrivePackage	Physical aspects of a disk drive. This class is used to gather information about the association of drive magazines to disk drives.

Properties for TPD_StorageSystem

Table 145 TPD_StorageSystem Properties

Property	Description
Name	The node WWN for the storage system.
ElementName	User-friendly name of the storage system.

Properties for TPD_DriveCage

Table 146 TPD_DriveCage Properties

Property	Description
Position	The position of the drive cage in the storage system (such as the cage number).
ElementName	User-friendly name of the Drive Cage.
Manufacturer	Manufacturer of the Drive Cage.
Model	Model of the Drive Cage.
SerialNumber	Serial number of the Drive Cage.

Properties for TPD_CageInterfaceCard

Table 147 TPD_CageInterfaceCard Properties

Property	Description
Position	The position of the `CageInterfaceCard` in the `DriveCage`.
Manufacturer	Manufacturer of the `CageInterfaceCard`.
Model	Model of the `CageInterfaceCard`.
SerialNumber	`SerialNumber` of the `CageInterfaceCard`.
FirmwareVersion	Revision of the firmware running on the `CageInterfaceCard`.

Properties for TPD_Magazine

Table 148 TPD_Magazine Properties

Property	Description
ElementName	User-friendly name of the drive magazine. The format vary depending on type of drive cage.
Position	The position of this drive magazine in the drive cage.
Manufacturer	Manufacturer of the magazine.
Model	Model of the magazine.
SerialNumber	Serial number of the magazine.

Properties for TPD_DiskDrive

Table 149 TPD_DiskDrive Properties

Property	Description
ElementName	User-friendly name of the `DiskDrive`. The format vary depending on the type of the `DriveCage`.
Name	The Fibre Channel WWN of this `DiskDrive`.
Position	The position of this `DiskDrive` in the magazine.
ID	ID assigned to this `DiskDrive` by the system.

Properties for TPD_DiskDrivePackage

Table 150 TPD_DiskDrivePackage Properties

Property	Description
Manufacturer	Manufacturer of the `DiskDrive`.
Model	Model of the `DiskDrive`.
SerialNumber	Serial number of the `DiskDrive`.

Properties for TPD_DiskSoftwareIdentity

Table 151 TPD_DiskSoftwareIdentity Properties

Property	Description
Manufacturer	Manufacturer of the `DiskDrive` firmware.
VersionString	Version of the firmware running on the `DiskDrive`.

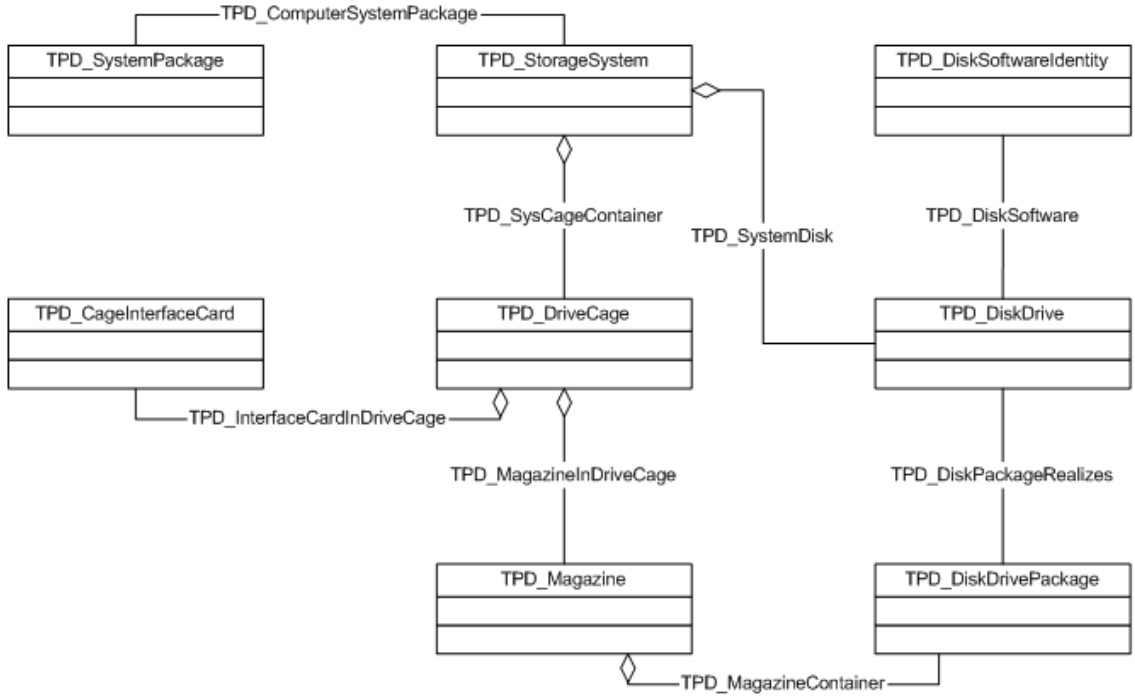
Supported Methods

Table 152 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 15 Disk Enclosure Subsystem



Power and Cooling

The CIM Server provides the ability to retrieve inventory information for the physical devices used to power and cool the storage system.

Power and Cooling CIM Classes

Table 153 Power and Cooling CIM Key Classes

Class	Description
TPD_StorageSystem	The storage array.
TPD_NodeSystem	Provides information about the controller nodes that make up the storage system.
TPD_DriveCage	Provides information about storage system drive cages.
TPD_PowerSupply	Provides information about power supplies for both controller nodes and disk cages.
TPD_Battery	Provides information about batteries attached to controller node power supplies.
TPD_Fan	Provides information about fans, including fans within each power supply and external fans connected to controller nodes.

Properties for TPD_StorageSystem

Table 154 TPD_StorageSystem Properties

Property	Description
Name	The node WWN for the storage system.
ElementName	User-friendly name of the storage system.

Properties for TPD_NodeSystem

Table 155 TPD_NodeSystem Properties

Property	Description
Position	The position of the controller node in the storage system (such as the node number).
ElementName	User-friendly name of the controller node.

Properties for TPD_DriveCage

Table 156 TPD_DriveCage Properties

Property	Description
Position	The position of the `DriveCage` in the storage system (such as the cage number).
ElementName	User-friendly name of the `DriveCage`.
Manufacturer	Manufacturer of the Drive Cage.
Model	Model of the Drive Cage.
SerialNumber	Serial number of the Drive Cage.

Properties for TPD_PowerSupply

Table 157 TPD_PowerSupply Properties

Property	Description
Position	The position of the `PowerSupply` in the storage system (such as the PS number).
Manufacturer	Manufacturer of the `PowerSupply`.
Model	Model of the `PowerSupply`.
SerialNumber	Serial number of the `PowerSupply`.

Properties for TPD_Battery

Table 158 TPD_Battery Properties

Property	Description
Position	The position of the Battery in the storage system (such as the battery number).
Manufacturer	Manufacturer of the Battery.
Model	Model of the Battery.
SerialNumber	Serial number of the Battery.

Properties for TPD_Fan

Table 159 TPD_Fan Properties

Property	Description
Position	The position of the Fan in the `PowerSupply` or Storage System (such as the fan number).
Manufacturer	Manufacturer of the Fan.
Model	Model of the Fan.
SerialNumber	Serial number of the Fan.

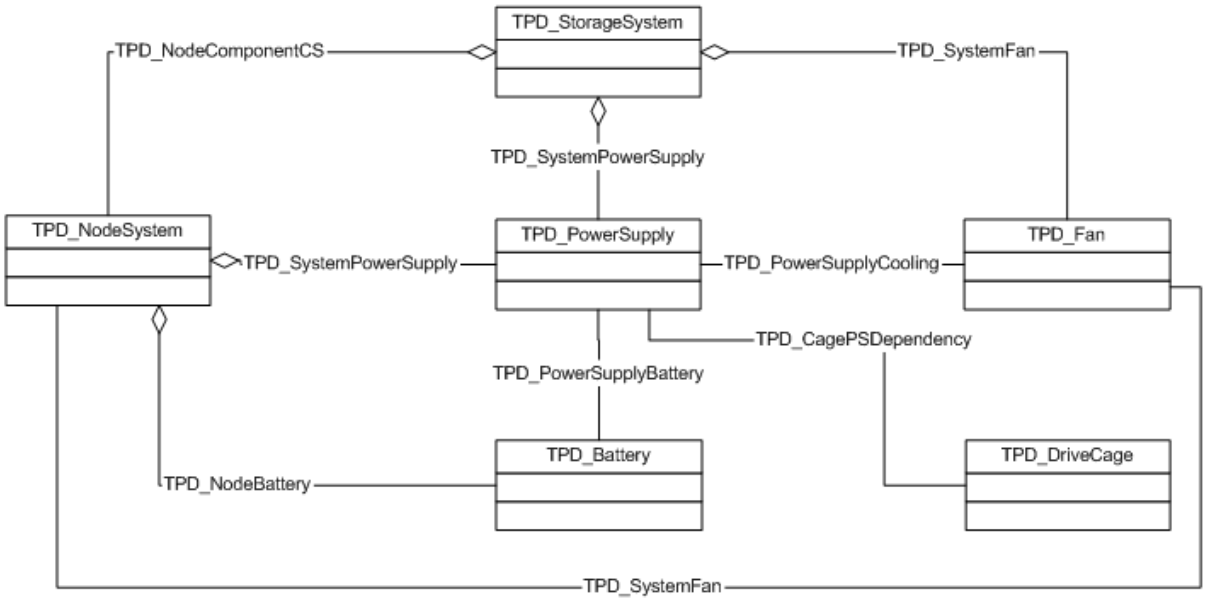
Supported Methods

Table 160 Supported Methods for TPD_StorageSystem

Method	Description
modifyInstance()	Provides the ability to set owner and contact information for the `StorageSystem`.

UML Diagram

Figure 16 UML Diagram



Domain, User and Licenses

Description

The 3PAR CIM Server provides the ability to retrieve domain and privilege information about the user. This information can be broken down into the following areas:

- Licenses
- User Domains
- User Privileges
- Domains
- Domain Sets

These informations are available in TPD_StorageSystem class and TPD_StorageDomainGroup.

CIM Classes

The key classes are as follows:

Class	Description
TPD_StorageSystem	The Storage array.
TPD_StorageDomainGroup	Models a domain set and provides information on the domain set and its members.
TPD_HostedStorageDomainGroup	Association between TPD_StorageSystem and TPD_StorageDomainGroup.
TPD_StorageHardwareIDCollection	Contains domain to which the host belongs
TPD_DynamicStoragePool	Contains domain to which the pool belongs
TPD_StorageVolume	Contains domain to which the volume belongs

Properties for TPD_StorageSystem

Property	Description
Licenses	Array of all the licenses enabled on the system.
LicenseIssueDate	Date on which the licenses were issued
LicenseExpirationDate	Date on which the licenses will expire.
CurrentUserDomains	Domains that the current user belongs to. If the user is not a domain user, value will show 'all'.
CurrentUserPrivileges	Privileges of the current user. Each element in this array specifies the privilege of the current user for the corresponding domain in the CurrentUserDomains array.
Domains	Array of domains visible to the current user.

Properties for TPD_StorageDomainGroup

Property	Description
ElementName	Name of the domain set
GroupMembers	Array of the member domains of this set.

Supported Methods

Methods for TPD_StorageHardwareIDCollection

Method	Description
modifyInstance()	Provides the ability to modify or unset host domain.

Thin Provisioning and TPVV Manipulations

See Chapter 4, "CIM API SMI-S Support" (page 14) for complete details on DynamicStoragePool (DSP) and TPVV manipulations.

A Managed Object Format Files

This appendix provides the contents of the Managed Object Format (MOF) files for the HP 3PAR Storage System. These MOF files derive from the standard MOF files contained in CIM, version 2.23. Note: The InServ Storage Server has been rebranded as HP 3PAR Storage System.There are instances in this document where menu items and command output refer to the HP 3PAR Storage System as InServ or InServ Storage Server.

For information regarding standard MOF files, refer to DMTF at www.dmtf.org.

3PAR_InterOp.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
// Copyright 2005-2006 3PAR, Inc. All Rights Reserved.
// This software is the property of 3PAR, Inc. Distribution
// or deployment of the source or derived binaries of this
// software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//
// File : 3PAR_InterOp.mof
//
// Purpose : This MOF contains 3PAR classes that will be loaded
// into root/PG_InterOp namespace.
//
// Date created: Apr./15/2005
//
////////////////////////////////////////////////////////////////////////////
// ==================================================================
// RegisteredProfile
// ==================================================================
[Description (
  "RegisteredProfile tells clients what profiles 3PAR InServ Storage "
  "Server supports. ")]
class TPD_RegisteredProfile : CIM_RegisteredProfile {
};

// ==================================================================
// RegisteredSubProfile
// ==================================================================
[Description (
  "A RegisteredSubProfile subclasses RegisteredProfile to "
  "indicate that a scoping profile is required to provide "
  "context. The latter is specified by the mandatory association, "
  "SubProfileRequiresProfile.")]
class TPD_RegisteredSubProfile : CIM_RegisteredSubProfile {
};

// ==================================================================
// SubProfileRequiresProfile - association between RegisteredProfile and
// RegisteredSubProfile
// ==================================================================
[Association,
 Description (
  "A subprofile requires another RegisteredProfile for context. "
  "This association mandates the scoping relationship between a "
  "subprofile and its scoping profile.")]
class TPD_SubProfileRequiresProfile : CIM_SubProfileRequiresProfile {
```

```
 [Override ( "Antecedent" ), Min ( 1 ), Description (
   "The RegisteredProfile that is referenced/required by the "
   "subprofile.")]
 TPD_RegisteredProfile REF Antecedent;

 [Override ( "Dependent" ), Description (
   "A RegisteredSubProfile that requires a scoping profile, for "
   "context.")]
 TPD_RegisteredSubProfile REF Dependent;
};

// ===================================================================
// RegisteredProfile and ComputerSystem association
// ===================================================================
[Association, Description (
  "The CIM_ElementConformsToProfile association defines the "
  "RegisteredProfiles to which the referenced ManagedElement is "
  "conformant. This association specifically ties RegisteredProfile "
  "with 3PAR InServ Storage Server (TPD_StorageSystem, subclass of "
  "CIM_ComputerSystem) if the profile is Array, and with the "
  "Object Manager if the profile is Server.")]
class TPD_ElementConformsToProfile : CIM_ElementConformsToProfile {

 [Override ("ConformantStandard"), Description (
   "The RegisteredProfile to which the InServ Storage Server conforms.")]
 TPD_RegisteredProfile REF ConformantStandard;
};

// ===================================================================
// ObjectManager
// ===================================================================
[Description (
  "TPD_ObjectManager defines the capabilities of the 3PAR InServ "
  "Storage System CIM Server in which this ObjectManager class "
  "resides. Details related to communicating with the ObjectManager, "
  "and the Manager's basic capabilities, are stored in instances of the "
  "associated TPD_CIMXMLCommunicationMechanism class available through "
  "the TPD_CommMechanismForManager association. ")]
class TPD_ObjectManager : CIM_ObjectManager {
};

// ===================================================================
// TPD_HostedOMService: Association between CIM_System and TPD_ObjectManager
// ===================================================================
[Association,
 Description (
  "TPD_HostedOMService is an association between TPD_ObjectManager and the "
  "System on which the functionality resides. The cardinality of "
  "this association is 1-to-many. A System may host many "
  "Services. Services are weak with respect to their hosting "
  "System. Heuristic: A Service is hosted on the System where the "
  "LogicalDevices or SoftwareFeatures that implement the Service "
  "are located. The model does not represent Services hosted "
  "across multiple systems. This is modeled as an "
  "ApplicationSystem that acts as an aggregation point for "
  "Services, that are each located on a single host.")]
class TPD_HostedOMService : CIM_HostedService {

 [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
   "The InServ StorageSystem.")]
 TPD_StorageSystem REF Antecedent;

 [Override ( "Dependent" ), Weak, Description (
  "The Object Manager hosted on the 3PAR InServ Storage System.")]
 TPD_ObjectManager REF Dependent;
};
```

```
// ===================================================================
// Namespace
// ===================================================================
[Description (
"Namespace provides a domain (in other words, a container), in "
  "which the instances [of a class] are guaranteed to be unique "
  "per the KEY qualifier definitions. It is named relative to the "
  "TPD_ObjectManager implementation that provides such a domain.")]
class TPD_Namespace : CIM_Namespace {
};

// ===================================================================
// NamespaceInManager - association between Namespace and Object Mgr
// ===================================================================
[Association,
 Description (
  "NamespaceInManager is an association describing the Namespaces "
  "hosted by the TPD_ObjectManager.")]
class TPD_NamespaceInManager: CIM_NamespaceInManager {

   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
    "The ObjectManager containing a Namespace.")]
 TPD_ObjectManager REF Antecedent;
   [Override ( "Dependent" ), Weak, Description (
    "The Namespace in an ObjectManager.")]
 TPD_Namespace REF Dependent;
};

// ===================================================================
// CIMXMLCommunicationMechanism
// ===================================================================
[Description (
  "This class specializes ObjectManagerCommunicationMechanism, "
  "adding properties specific to the CIM-XML protocol (XML "
  "encoding and CIM Operations) supported by the 3PAR CIM "
  "server Object Manager.")]
class TPD_CIMXMLCommunicationMechanism : CIM_CIMXMLCommunicationMechanism {
  [Description ( "Is this link encrypted or not. http or https. ")]
 String namespaceType;
  [Description ( "NamespaceType definition. Note that namespaceType"
  "is the official name of this field in the CIM specification "
  "Object Name definition."),
  ValueMap {"0", "2", "3", "4..100" },
  Values { "Unknown", "http", "https", "Pegasus Reserved" }]
 Uint16 namespaceAccessProtocol;

  [Description ( "IP Address for this comm mechanism. This MUST "
  "BE the complete address so that the CIM Server can be "
  "addressed from the network. It must include the port "
  "number unless the DMTF defined default ports are used." )]
 String IPAddress;
};

// ===================================================================
// TPD_CommMechanismForManager - Association between TPD_ObjectManager
// and TPD_CIMXMLCommunicationMechanism
// ===================================================================
[Association,
 Description (
  "CommMechanismForManager is an association between an "
  "ObjectManager and an TPD_CIMXMLCommunicationMechanism "
  "class. The latter describes a possible encoding/protocol/ set "
  "of operations for accessing the referenced ObjectManager.")]
class TPD_CommMechanismForManager : CIM_CommMechanismForManager {
```

```
  [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
   "The 3PAR CIM server ObjectManager whose communication mechanism is "
   "described.")]
 TPD_ObjectManager REF Antecedent;

  [Override ( "Dependent" ), Min ( 1 ), Description (
   "The encoding/protocol/set of operations that may be used to "
   "communicate with the referenced ObjectManager.")]
 TPD_CIMXMLCommunicationMechanism REF Dependent;
};

// ===================================================================
// TPD_HostedCIMXMLCommMechanism - Association between TPD_StorageSystem
// and TPD_CIMXMLCommunicationMechanism
// ===================================================================
[Association,
 Description (
  "HostedCIMXMLCommMechanism is an association between a Service "
  "AccessPoint and the System on which it is provided. The "
  "cardinality of this association is 1-to-many and is weak with "
  "respect to the System. Each System may host many "
  "ServiceAccessPoints. Heuristic: If the implementation of the "
  "ServiceAccessPoint is modeled, it must be implemented by a "
  "Device or SoftwareFeature that is part of the System hosting "
  "the ServiceAccessPoint.")]
class TPD_HostedCIMXMLCommMechanism : CIM_HostedAccessPoint {

  [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
 "The InServ StorageSystem.")]
 TPD_StorageSystem REF Antecedent;

  [Override ( "Dependent" ), Weak, Description (
   "The encoding/protocol/set of operations that may be used to "
   "communicate with the referenced ObjectManager.")]
 TPD_CIMXMLCommunicationMechanism REF Dependent;
};

// ===================================================================
// 3PAR ObjMgrSoftwareIdentity
// ===================================================================
[Description (
  "3PAR ObjMgrSoftwareIdentity describes the Object Manager"
  "software used to instrument the various profiles and subprofiles.")]
class TPD_ObjMgrSoftwareIdentity : CIM_SoftwareIdentity
{
};

// ===================================================================
// 3PAR Profiles and Subprofiles software identity
// ===================================================================
[Association,
 Description (
  "3PAR ObjMgrSoftwareIdentity and RegisteredProfile"
  "and RegisteredSubProfile mapping")]
class TPD_ProfileSoftwareIdentity : CIM_ElementSoftwareIdentity {

   [Override ( "Antecedent" ), Description (
   "The software that is needed to instrument the providers for "
 "the RegisterdProfiles and RegisteredSubProfiles.")]
 TPD_ObjMgrSoftwareIdentity REF Antecedent;

  [Override ( "Dependent" ), Description (
  "The RegisterdProfile or RegisteredSubProfile.")]
 CIM_RegisteredProfile REF Dependent;
};
```

3PAR_TPD.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
// Copyright 2007 3PAR, Inc. All Rights Reserved.
// This software is the property of 3PAR, Inc. Distribution
// or deployment of the source or derived binaries of this
// software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//
// File : 3PAR_TPD.mof
//
// Purpose : This MOF contains 3PAR classes that will be loaded
//      into root/tpd namespace.
//
// Date created: Apr./15/2005
//
////////////////////////////////////////////////////////////////////////////
#pragma include ("SNIA_StoragePool.mof")

// ===================================================================
// 3PAR SCSIController
// ===================================================================
[Description (
 "3PAR SCSI ProtocolController ")]
class TPD_SCSIController: CIM_SCSIProtocolController
{
};

// ===================================================================
// 3PAR StoragePool
// ===================================================================
[Description (
 "3PAR StoragePool ")]
class TPD_StoragePool: SNIA_StoragePool
{

 [Description (
  "Type of disk drives that are in this pool. Not applicable "
  "to primordial pool."),
 ValueMap {"0", "1", "2", "4"},
 Values {"Unknown", "FC", "Nearline", "SSD"} ]
 uint16 DiskDeviceType;
};

// ===================================================================
// 3PAR DynamicStoragePool:
// The space for this pool is dynamically allocated on demand.
// ===================================================================
[Description (
 "3PAR DynamicStoragePool. The space for this pool is dynamically"
 " allocated on demand. When the pool is initially created, a minimum"
 " size of disk space is allocated. More space is allocated on demand"
 " until it reaches the maximum size (SpaceLimit). Thin-provisioned"
 " StorageVolume draw space from this pool." )]
class TPD_DynamicStoragePool: SNIA_StoragePool
{
 [Description (
  "Type of disk drives that are in this pool. Not applicable "
  "to primordial pool."),
 ValueMap {"0", "1", "2", "4"},
```

```
 Values {"Unknown", "FC", "Nearline", "SSD"} ]
 uint16 DiskDeviceType;

  [Description (
  "The size of each allocation when the pool grows."),
  Units ( "Bytes" )]
uint64 AllocationUnit;

  [Description (
  "This is the nominal number of bytes allocated on disk for Snapshot "
  "data (SD) of this pool, not including RAID overhead."),
  Units ( "Bytes" )]
uint64 ConsumedSnapDataSpace;

  [Description (
   "This is the actual number of bytes allocated on disk for Snapshot "
   "data (SD) space of this pool, including RAID overhead. "),
   Units ( "Bytes" )]
uint64 ConsumedSnapDataSpacePlusMetaData;

  [Description (
  "This is the nominal number of bytes allocated on disk for Snapshot "
  "admin (SA) of this pool, not including RAID overhead."),
  Units ( "Bytes" )]
uint64 ConsumedSnapAdminSpace;

  [Description (
  "This is the actual number of bytes allocated on disk for Snapshot "
  "admin (SA) space of this pool, including RAID overhead. "),
  Units ( "Bytes" )]
uint64 ConsumedSnapAdminSpacePlusMetaData;

  [Description (
  "This is the number of bytes allocated on disk for "
  "this pool, not including RAID overhead. "),
  Units ( "Bytes" )]
uint64 ConsumedSpace;

  [Description (
  "This is the current number of bytes being used by all "
  "StorageVolumes and DeltaReplicaStoragePool (RAID "
  "overhead already taken into account) allocated from "
  "this pool. For example, a RAID-10 DynamicStoragePool with 50 GB "
  "being allocated to all volumes. The CurrentSpaceConsumed is 50 GB."),
  Units ( "Bytes" )]
uint64 CurrentSpaceConsumed;

  [Description (
   "Describe the RAID type of the pool. This value makes it easy"
   " for traversing CapabilitiesOfStoragePool association."),
 ValueMap {"0", "10", "50", "60"},
 Values {"RAID0", "RAID10", "RAID50", "RAID60"} ]
 uint16 RaidType;

 [Description (
  "Issue warning alert when space allocation exceeds this amount."
  "A size of 0 means no warning limit is enforced. Default is 0."),
  Units( "Bytes" )]
 uint64 SnapDataSpaceWarningLimit = 0;

 [Description (
  "Name of the administrative domain that this pool belongs "
  "to.\n")]
 String Domain;
};
```

```
// ===================================================================
// 3PAR StorageVolume
// ===================================================================
[Description (
 "3PAR StorageVolume ")]
class TPD_StorageVolume: SNIA_StorageVolume
{

  [Description (
   "Total number of logically contiguous blocks including overhead "
  "and metadata, of size Block Size, which form this Extent. "
  "The total size of the Extent can be calculated by multiplying "
  "BlockSize by NumberOfRawBlocks. If the BlockSize is 1, this "
  "property is the total size of the Extent." ),
   MappingStrings { "MIF.DMTF|Host Storage|001.5",
    "MIB.IETF|HOST-RESOURCES-MIB.hrStorageSize" }]
  uint64 NumberOfRawBlocks;

 [Description (
   "Describe the RAID type of the volume. This value makes it easy"
   " for traversing ElementSettingData association."),
 ValueMap {"0", "10", "50", "60"},
 Values {"RAID0", "RAID10", "RAID50", "RAID60"} ]
 uint16 RaidType;

 [Description (
  "Volume overall state. Possible value is a combination of following states: "
  "0x00000000, 0x00000001, 0x00000002, 0x00000004, 0x00000008, 0x00000010, "
  "0x00000020, 0x00000040, 0x00000080, 0x00000100, 0x00000200, 0x00000400, "
  "0x00000800, 0x00001000"),
 BitMap { "0","1","2","3","4","5","6","7","8","9","10","11","12","13" },
 BitValues {"Unknown", "Started", "Not Started", "Auto-Check",
 "Checking", "Need Check", "Need Disk", "Promoting",
  "Copy Failed", "Stale", "Copying", "Admin Failed", "Preserved",
 "Meta data corrupted"} ]
uint32 OtherOperationalStatus;
[Description (
 "Describe the type of the volume."),
ValueMap {"0", "1", "2", "3", "4"},
Values {"Unknown", "Base", "Physical Copy", "Virtual Copy", "Remote Copy"} ]
uint16 VolumeType;

[Description (
 "Total number of logically contiguous blocks, of size Block "
 "Size, which form Raw Size Admin. The total size "
 "can be calculated by multiplying BlockSize by RawAdminBlocks.")]
uint64 RawAdminBlocks;

[Description (
 "The maximum number of blocks, of size BlockSize, which are "
 "available for consumption in Admin space. The total size "
 "can be calculated by multiplying BlockSize by ConsumableAdminBlocks.")]
uint64 ConsumableAdminBlocks;

[Description (
 "Total number of logically contiguous blocks, of size Block "
 "Size, which form Raw Size Copy. The total size "
 "can be calculated by multiplying BlockSize by RawCopyBlocks.")]
uint64 RawCopyBlocks;

[Description (
 "The maximum number of blocks, of size BlockSize, which are "
 "available for consumption in Copy space. The total size "
 "can be calculated by multiplying BlockSize by ConsumableCopyBlocks.")]
uint64 ConsumableCopyBlocks;
```

```
[Description (
 "The number of blocks consumed by this virtual copy volume (snapshot) "
 "from the admin snapshot space of the parent base volume. "
 "This is applicable only if VolumeType is 'Virtual Copy'.")]
uint64 UsedSnapshotAdminSpaceBlocks;

[Description (
 "The number of blocks consumed by this virtual copy volume (snapshot) "
 "from the data snapshot space of the parent base volume. "
 "This is applicable only if VolumeType is 'Virtual Copy'.")]
uint64 UsedSnapshotDataSpaceBlocks;

[Description (
 "The date this volume was created.")]
datetime DateCreated;

[Description (
 "Master node ID.") ]
uint16 MasterNode;

[Description (
 "Backup 1 node ID.") ]
uint16 BackupNode1;

[Description (
 "Backup 2 node ID.") ]
uint16 BackupNode2;

[Description (
 "Parent volume ID.") ]
uint32 ParentID;

[Description (
 "Volume policy bits used to store policies information. "
 "The available policies are: "
 "DisallowStaleSnapshot - When copy-on-write fails, don't write to r/w "
 "   base so that r/o snapshots remain valid/consistent with r/w "
 "   base. This means failures to update snapshot will be "
 "   considered a failure to write to the base volume as well. "
 "AllowStaleSnapshot - When copy-on-write fails, allow write to r/w base "
 "   to succeed, but snapshot becomes invalid or stale. This means "
 "   failures to update snapshot will not affect the write to the "
 "   base volume, but the snapshot will then be considered invalid. "
 "   This is the default setting. "
 "System - Volume is intended to be used for internal system purposes."
 "NoSystem - Volume is not intended to be used for internal system "
 "    purposes."
 "TPZeroFill - For Thin Provisioned volumes, if a host write results in "
 "   allocation of a new data page that is only partially "
 "   filled by the host write, then do a zero-fill on the "
 "   unwritten portion of the page to ensure that the host can "
 "   never read old data from deleted volumes or snapshots. "
 "   Current allocation page size is 16kb. This is the "
 "   default setting. "
 "TPNoZeroFill - Bypass the zero-fill stage on allocation of partially "
 "    written data pages. This is a performance improvement "
 "    setting for Thin Provisioned volumes. "
 "OneHost - This constrains the export of a volume to one host or "
 "     one host cluster (when cluster names may be used as a "
 "     host name). Unless the volume is under the complete "
 "     control of a cluster aware application, multiple hosts "
 "      (with NoOneHost policy) will corrupt each other's data. "
 "NoOneHost - This policy should only be used when exporting a volume to "
 "      multiple hosts for use by a cluster-aware application, "
 "    or if your shop uses 'port presents' VLUNs. "
 "     This is the default policy setting."
```

```
 "Cache - enable caching for VV (default)."
 "NoCache - disable all caching for the VV plus read ahead."
 "ZeroDetect - Check If incoming data is zero. If so, reclaim the space."
 "NoZeroDetect - No need to check if incoming data is zero."
 "Possible value is a combination of following values: "
 "  0x80000000 - DisallowStaleSnapshot"
 "  0x40000000 - AllowStaleSnapshot"
 "  0x20000000 - System"
 "  0x10000000 - NoSystem"
 "  0x08000000 - TPZeroFill"
 "  0x04000000 - TPNoZeroFill"
 " 0x02000000 - OneHost"
 "  0x01000000 - NoOneHost"
 "  0x00800000 - Cache"
 "  0x00400000 - NoCache"
 "  0x00200000 - ZeroDetection"
 "  0x00100000 - NoZeroDetection"),
 BitMap { "31","30","29","28","27","26","25","24","23","22","21","20" },
 BitValues {"DisallowStaleSnapshot", "AllowStaleSnapshot",
  "System", "NoSystem", "TPZeroFill", "TPNoZeroFill",
  "OneHost", "NoOneHost", "Cache", "NoCache", "ZeroDetection",
  "NoZeroDetection"}]
uint32 Policy;

[Description (
 "Define the VV geometry sectors_per_track value that is "
 "reported to the hosts via the SCSI mode pages. The valid "
 "range is from 4 to 8192 and the default value is 304.\n"),
 MinValue ( 4 ), MaxValue ( 8192 )]
uint32 GeometrySectorsPerTrack = 304;

[ Description (
 "Define the VV geometry heads_per_cylinder value that is "
 "reported to the hosts via the SCSI mode pages. The valid "
 "range is from 1 to 1024 and the default value is 8.\n"),
 MinValue ( 1 ), MaxValue ( 1024 )]
uint32 GeometryHeadsPerCylinder = 8;

[ Description (
 "Defines the sector size presented to the host for this virtual volume. "
 "This must be a power of 2 from 512 to 16384, so the valid inputs are "
 "512, 1024, 2048, 4096, 8192, and 16384. The default size is 512.\n"),
ValueMap {"512", "1024", "2048", "4096", "8192", "16384"},
Values {"512", "1024", "2048", "4096", "8192", "16384"} ]
uint32 GeometrySectorSize = 512;

[Description (
 "Describes the preferred availability of the volume. "
 "This value might be different from CurrentAvailability."),
ValueMap {"0", "8", "9", "10", "11", "12"},
Values {"Unknown", "Port", "Cage", "Magazine", "Disk", "Chunklet"} ]
uint16 PreferredAvailability;

[Description (
 "Describes the current availability of the volume. "
 "This value might be different from PreferredAvailability."),
ValueMap {"0", "8", "9", "10", "11", "12"},
Values {"Unknown", "Port", "Cage", "Magazine", "Disk", "Chunklet"} ]
uint16 CurrentAvailability;

[Description (
 "Describes the export state of the volume."),
ValueMap {"0", "1", "2"},
Values {"None", "Active", "Template"} ]
uint16 ExportState;
```

```
[Description (
 "Type of disk drives that are used to create this volume."),
ValueMap {"0", "1", "2", "4"},
Values {"Unknown", "FC", "Nearline", "SSD"} ]
uint16 DiskDeviceType;

[Description (
 "Allocation warning. Generate a warning alert when SD space "
 "of the volume exceeds the specified percentage of the volume "
 "size."),
Units ( "Percent" )]
uint32 SnapSpaceAllocationWarning;

[Description (
 "Allocation limit. The SD space of the volume is prevented "
 "from growing beyond the specified percentage of the volume "
 "size."),
Units ( "Percent" )]
uint32 SnapSpaceAllocationLimit;

[Description (
 "Allocation warning. Generate a warning alert when user space "
 "of the volume exceeds the specified percentage of the volume "
 "size. Applicable only to a thin-provisioned volume."),
Units ( "Percent" )]
uint32 UserSpaceAllocationWarning;

[Description (
 "Allocation limit. The user space of the volume is prevented "
 "from growing beyong the specified percentage of the volume "
 "size. Applicable only to a thin-provisioned volume."),
Units ( "Percent" )]
uint32 UserSpaceAllocationLimit;

[Description (
 "The DynamicStoragePool (DSP) the snapshot data (SD) "
 "are provisioned from.\n")]
String SnapDSPName;

[Description (
 "The DynamicStoragePool (DSP) the user space "
 "is provisioned from.\n")]
String UserDSPName;

[Description (
 "The number of user blocks that are actually allocated to a "
 "thin-provisioned volume. "
 "This is applicable only for a thin-provisioned volume.")]
uint64 ProvisionedConsumableBlocks;

[Description (
 "Name of the administrative domain that this volume belongs "
 "to.\n")]
String Domain;

[Description (
 "Number of chunklets in a set. "
 "For RAID 0 volume, this is always 1. "
 "For RAID 10 volume, this is equal to StorageSetting.DataRedundancy. "
 "Minimum value for RAID 10 volume is 2, maximum value is 4. "
 "For RAID 50 volume, this is also known as the parity "
 "set size."),
MinValue ( 1 ) ]
uint16 SetSize;

[Description (
```

```
 "Number of chunklets in a set that contains data. "
 "Meaningful for RAID 60 volume only."),
MinValue ( 3 ) ]
uint16 SetData;

[Description ("The time specifies when the volume will expire and "
 "can be deleted.") ]
datetime ExpirationTime;

[Description ("The time specifies the volume needs to be retained and "
 "can not remove until the retention time is expired.") ]
datetime RetentionTime;

[Description (
 "Describes the provisioning for the volume. "
 "Value can be one of the following:"
 "   Full   Fully provisioned VV, either with no Snp (snapshot)"
 "     space or with statically allocated Snp space."
 "   TPVV   Thin provisioned VV, with space for the base volume"
 "      allocated from the Usr space that is associated with"
 "      the UsrCPG. Snapshots allocate space from the Snp"
 "      space associated with the SnpCPG (if any)."
 "   Peer   Volume reserved for data migration from a remote array."
 "   Unknown unknown type."),
ValueMap {"0", "1", "4", "0xffff"},
Values {"Full", "TPVV", "Peer", "Unknown"} ]
uint16 ProvisioningType;

[Description (
 "Bits showing the activities of the volume."
 "Meaning of the bits are:"
 " Promoting - volume is being promoted "
 " Copying - volume is the target of a physical copy "
 " Tuning - volume is being tuned "
 " Closing - volume is being shutdown "
 " Removing - volume is being removed "
 " Creating - volume is being created "
 " Copy Source - volume is the source of a physical copy "
 " Removing Expired - retrying a volume removal "
 " Resyncing - volume is currently the target of a resync copy "
 " Importing - volume is importing "
 "Possible value is a combination of following values: "
 "   0x00100000 - Importing"
 "   0x00020000 - Resyncing"
 "   0x00010000 - Removing Expired"
 "   0x00004000 - Copy Source"
 "   0x00002000 - Creating"
 "   0x00001000 - Removing"
 "   0x00000800 - Closing"
 "   0x00000400 - Tuning"
 "   0x00000200 - Copying"
 "   0x00000100 - Promoting"),
 BitMap { "20","17","16","14","13","12","11","10","9", "8" },
 BitValues {"Importing", "Resyncing",
  "Removing Expired", "Copy Source", "Creating", "Removing",
  "Closing", "Tuning", "Copying", "Promoting"}]
 uint32 Activity;
};

// ===================================================================
// 3PAR HostedStoragePool
// ===================================================================
[Association, Aggregation, Composition,
 Description (
  "3PAR HostedStoragePool (association between StoragePool and "
  "StorageSystem) ")]
```

```
class TPD_HostedStoragePool: CIM_HostedStoragePool
{

 [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
  Description ("The parent system in the association.") ]
 TPD_StorageSystem REF GroupComponent;
};

// ===================================================================
// 3PAR AllocatedFromStoragePool
// ===================================================================
[Association,
 Description (
  "TPD_AllocatedFromStoragePool is a superclass association describing how "
  "volumes or pools are allocated from underlying StoragePools.")]
class TPD_AllocatedFromStoragePool: CIM_AllocatedFromStoragePool
{
};

// ===================================================================
// 3PAR ConcretePoolAllocatedFromPrimordialPool
// ===================================================================
[Association,
 Description (
  "TPD_ConcretePoolAllocatedFromPrimordialPool is an association between "
  "concrete StoragePool and the primordial StoragePool from which "
  "the capacity of the concrete pool is allocated.")]
class TPD_ConcretePoolAllocatedFromPrimordialPool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
  Description ( "The primordial StoragePool." )]
 TPD_StoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary concrete StoragePool." )]
 TPD_StoragePool REF Dependent;
};

// ===================================================================
// 3PAR DynamicPoolAllocatedFromConcretePool
// ===================================================================
[Association,
 Description (
  "TPD_DynamicPoolAllocatedFromConcretePool is an association between "
  "DynamicStoragePool and the concrete StoragePool from which the "
  "the capacity of DynamicStoragePool is allocated.")]
class TPD_DynamicPoolAllocatedFromConcretePool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
  Description ( "The concrete StoragePool." )]
 TPD_StoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary DynamicStoragePool." )]
 TPD_DynamicStoragePool REF Dependent;
};

#pragma include ("3PAR_TPDCopySvcs.mof")
// ===================================================================
// 3PAR DeltaReplicaPoolAllocatedFromConcretePool
// ===================================================================
[Association,
 Description (
  "TPD_DeltaReplicaPoolAllocatedFromConcretePool is an association between "
```

```
  "DeltaReplicaStoragePool and concrete StoragePool from which the "
  "the capacity of DeltaReplicaStoragePool is allocated.")]
class TPD_DeltaReplicaPoolAllocatedFromConcretePool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
  Description ( "The concrete StoragePool." )]
 TPD_StoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary DeltaReplicaStoragePool." )]
 TPD_DeltaReplicaStoragePool REF Dependent;
};

// ===================================================================
// 3PAR DeltaReplicaPoolAllocatedFromDynamicPool
// ===================================================================
[Association,
  Description (
  "TPD_DeltaReplicaPoolAllocatedFromDynamicPool is an association between "
  "DeltaReplicaStoragePool and the DynamicStoragePool from which the "
  "the capacity of DeltaReplicaStoragePool is allocated.")]
class TPD_DeltaReplicaPoolAllocatedFromDynamicPool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
  Description ( "The DynamicStoragePool." )]
 TPD_DynamicStoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary DeltaReplicaStoragePool." )]
 TPD_DeltaReplicaStoragePool REF Dependent;
};

// ===================================================================
// 3PAR VolumeAllocatedFromConcretePool
// ===================================================================
[Association,
 Description (
  "TPD_VolumeAllocatedFromConcretePool is an association between "
  "StorageVolume and the concrete StoragePool from which the "
  "the capacity of StorageVolume is allocated.")]
class TPD_VolumeAllocatedFromConcretePool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
  Description ( "The concrete StoragePool." )]
 TPD_StoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary StorageVolume." )]
 TPD_StorageVolume REF Dependent;
};

// ===================================================================
// 3PAR VolumeAllocatedFromDynamicPool
// ===================================================================
[Association,
 Description (
  "TPD_VolumeAllocatedFromDynamicPool is an association between "
  "StorageVolume and the DynamicStoragePool from which the "
  "the capacity of StorageVolume is allocated.")]
class TPD_VolumeAllocatedFromDynamicPool : TPD_AllocatedFromStoragePool
{

  [Override ( "Antecedent" ),
```

```
  Description ( "The DynamicStoragePool." )]
 TPD_DynamicStoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary StorageVolume." )]
 TPD_StorageVolume REF Dependent;
};

// ===================================================================
// 3PAR VolumeAllocatedFromDeltaReplicaPool
// ===================================================================
[Association,
 Description (
  "TPD_VolumeAllocatedFromDeltaReplicaPool is an association between "
  "a snapshot StorageVolume and the DeltaReplicaStoragePool from which the "
  "the capacity of snapshot volume is allocated.")]
class TPD_VolumeAllocatedFromDeltaReplicaPool : TPD_AllocatedFromStoragePool
{
  [Override ( "Antecedent" ),
  Description ( "The DeltaReplicaStoragePool." )]
 TPD_DeltaReplicaStoragePool REF Antecedent;

  [Override ( "Dependent" ),
  Description ( "The subsidiary snapshot StorageVolume." )]
 TPD_StorageVolume REF Dependent;
};

// ===================================================================
// 3PAR StorageSetting
// ===================================================================
[Description (
 "3PAR parent StorageSetting class for TPD_StorageSetting and "
 "TPD_VolumeSetting.")]
class TPD_TopLevelStorageSetting : SNIA_StorageSetting
{
 [Write, Description (
  "The number of data chunklets in a RAID 50 or RAID 60 parity set. "
  "The system default is 4 chunklets: 3 for data and 1 for "
  "parity (3+1).\n"
  "This is meaningless for RAID 0 or RAID 10 volumes."),
 MinValue ( 3 ), MaxValue ( 9 ) ]
 uint16 ParitySetSize;

 [Write, Description (
  "Base ID of the volume. If NULL, the next available "
  "id will be used."),
 MinValue ( 1 ) ]
 uint32 BaseID;

 [Write, Description (
  "Whether a RAID 10 or RAID 50 volume supports a failure "
  "of one port pair, one cage or one magazine. For RAID 10 "
  "and RAID 50, the default for user, snap admin and snap "
  "data areas are cage. For RAID 0, the default for the "
  "snap admin area is cage.\n"),
 ValueMap { "8", "9", "10" },
 Values { "Port", "Cage", "Magazine" }]
 uint16 HighAvailability;

 [Write, Description (
  "Chunklet location preference. Allows chunklets to "
  "be placed physically close to other chunklets with "
  "the same location preference.\n"
  "  First - attempt to use the lowest numbered\n"
  "   available chunklets.\n"
  "  Last - attempt to use the highest numbered\n"
```

```
  "   available chunklets.\n"
  "The default value is First.\n"),
 ValueMap { "1", "2" },
 Values { "First", "Last" }]
 uint16 ChunkletLocationPreference;

 [Write, Description (
  "Size for the snap volume in bytes (maximum 1073741568). "
  "The default value is not set.\n"
  "If DSPName is specified, this value should be 0.\n"
  "Only one of SnapVolumePercentage and SnapVolumeSize "
  "can be set.\n"),
 Units ( "Bytes" )]
 uint64 SnapVolumeSize;

 [Write, Description (
  "Size for the snap volume in percentage of volume user "
  "size.\n"
  "The default value is 0.\n"
  "If DSPName is specified, this value should be 0.\n"
  "Only one of SnapVolumePercentage and SnapVolumeSize "
  "can be set.\n")]
 uint16 SnapVolumePercentage;

 [Write, Description (
  "Size for the admin volume in bytes (maximum 1073741568). "
  "The default value is not set.\n"
  "This property is valid only in a grow volume operation."),
 Units ( "Bytes" )]
 uint64 AdminVolumeSize;

 [Description (
  "Snapshot space allocation warning. Generate a warning alert "
  "when SD space of the volume exceeds the specified percentage "
  "of the volume size. This property can be set independent of "
  "SpaceLimit since, unlike SpaceLimitWarningThreshold, this is "
  "a percentage of the volume size and not of SpaceLimit. If "
  "this is set, then SpaceLimitWarningThreshold is ignored."),
 Units ( "Percent" )]
 uint32 SnapSpaceAllocationWarning;

 [Description (
  "User space allocation warning. Generate a warning alert "
  "when user space of the volume exceeds the specified percentage "
  "of the volume size. This property can be set independent of "
  "UserSpaceLimit since, unlike UserSpaceLimitWarningThreshold, this is "
  "a percentage of the volume size and not of UserSpaceLimit. If "
  "this is set, then UserSpaceLimitWarningThreshold is ignored. "
  "Applicable only to thin provisioned volumes."),
 Units  ( "Percent" )]
 uint32 UserSpaceAllocationWarning;

[Description (
  "UserSpaceLimit is the consumption limit for the allocated "
  "user space of the associated StorageVolumes. Please note "
  "that SpaceLimit refers to the consumption limit of the "
  "allocated space of a DynamicStoragePool or the snap space "
  "(DeltaReplicaStoragePool) of the associated StorageVolumes. "
  "Applicable only to thin provisioned volumes. "
  "If UserSpaceLimit = 0 (the default) then no limits are asserted "
  "on the amount of space consumed."),
 Units ( "Bytes" )]
 uint64 UserSpaceLimit = 0;

[Description (
  "If the associated storage volume may dynamically consume "
```

```
 "an increasing amount of user space and the user space limit "
 "is enforced on the volume, then a non-zero warning threshold "
 "percentage indicates when a warning indication should be "
 "generated based on the total amount of space consumed being "
 ">= (UserSpaceLimit*UserSpaceLimitWarningThreshold)/100."
 "Please note that SpaceLimitWarningThreshold refers to "
 "the threshold of allocated space of a DynamicStoragePool or "
 "the snap space (DeltaReplicaStoragePool) of the associated "
 "StorageVolumes.\n "
 "Applicable only to thin provisioned volumes."),
Units ( "Percentage" ), MinValue ( 0 ), MaxValue ( 100 )]
uint16 UserSpaceLimitWarningThreshold;

[Write, Description (
 "The DynamicStoragePool (DSP) the snapshot data (SD) "
 "are provisioned from.\n")]
 String SnapDSPName;

[Write, Description (
 "The DynamicStoragePool (DSP) the user space "
 "is provisioned from.\n")]
String UserDSPName;

[Write, Description (
 "If TRUE, the volume or pool is thinly provisioned.") ]
boolean ThinlyProvisioned;

[Write, Description (
 "Define volume policy. The available policies are: "
 "DisallowStaleSnapshot - When copy-on-write fails, don't write to r/w "
 "    base so that r/o snapshots remain valid/consistent with r/w "
 "    base. This means failures to update snapshot will be "
 "    considered a failure to write to the base volume as well. "
 "AllowStaleSnapshot - When copy-on-write fails, allow write to r/w base "
 "    to succeed, but snapshot becomes invalid or stale. This means "
 "    failures to update snapshot will not affect the write to the "
 "    base volume, but the snapshot will then be considered invalid. "
 "    This is the default setting. "
 "System - Volume is intended to be used for internal system purposes."
 "NoSystem - The volume is not intended to be used for internal system "
 "    purposes."
 "TPZeroFill - For Thin Provisioned volumes, if a host write results in "
 "    allocation of a new data page that is only partially "
 "    filled by the host write, then do a zero-fill on the "
 "    unwritten portion of the page to ensure that the host can "
 "    never read old data from deleted volumes or snapshots. "
 "    Current allocation page size is 16kb. This is the "
 "    default setting. "
 "TPNoZeroFill - Bypass the zero-fill stage on allocation of partially "
 "    written data pages. This is a performance improvement "
 "    setting for Thin Provisioned volumes. "
 "OneHost - This constrains the export of a volume to one host or "
 "    one host cluster (when cluster names may be used as a "
 "    host name). Unless the volume is under the complete "
 "    control of a cluster aware application, multiple hosts "
 "    (with NoOneHost policy) will corrupt each other's data. "
 "NoOneHost - This policy should only be used when exporting a volume to "
 "    multiple hosts for use by a cluster-aware application, "
 "    or if your shop uses 'port presents' VLUNs. "
 "    This is the default policy setting."
 "ZeroDetect - Check If incoming data is zero. If so, reclaim the space."
 "NoZeroDetect - No need to check if incoming data is zero."
 "Possible value is a combination of following values: "
 "   0x80000000 - DisallowStaleSnapshot"
 "   0x40000000 - AllowStaleSnapshot"
 "   0x20000000 - System"
```

```
 "   0x10000000 - NoSystem"
 "   0x08000000 - TPZeroFill"
 "   0x04000000 - TPNoZeroFill"
 "   0x02000000 - OneHost"
 "   0x01000000 - NoOneHost"
 "   0x00200000 - ZeroDetection"
 "   0x00100000 - NoZeroDetection"
 ),
BitMap { "31","30","29","28","27","26","25","24","21","20" },
BitValues {"DisallowStaleSnapshot", "AllowStaleSnapshot",
 "System", "NoSystem", "TPZeroFill", "TPNoZeroFill",
 "OneHost", "NoOneHost", "ZeroDetection", "NoZeroDetection"} ]
uint32 Policy;

[Write, Description (
 "Define the VV geometry sectors_per_track value that is "
 "reported to the hosts via the SCSI mode pages. The valid "
 "range is from 4 to 8192 and the default value is 304.\n"),
 MinValue ( 4 ), MaxValue ( 8192 )]
uint32 GeometrySectorsPerTrack;

[Write, Description (
 "Define the VV geometry heads_per_cylinder value that is "
 "reported to the hosts via the SCSI mode pages. The valid "
 "range is from 1 to 1024 and the default value is 8.\n"),
 MinValue ( 1 ), MaxValue ( 1024 )]
uint32 GeometryHeadsPerCylinder;

[Write, Description (
 "Defines the sector size presented to the host for this virtual volume. "
 "This must be a power of 2 from 512 to 16384, so the valid inputs are "
 "512, 1024, 2048, 4096, 8192, and 16384. The default size is 512.\n"),
ValueMap {"512", "1024", "2048", "4096", "8192", "16384"},
Values {"512", "1024", "2048", "4096", "8192", "16384"} ]
uint32 GeometrySectorSize;

[Write, Description (
 "Specifies the node number, the list of node numbers or "
 "the range of node numbers the disks must have their "
 "primary path on.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskPrimPathNodes[];

[Write, Description (
```

```
 "Specifies the PCI or PCIe slot, the list of PCI, PCIe slots or "
 "the range of PCI or PCIe slots the disks must have their "
 "primary path on.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskPrimPathPCISlots[];

[Write, Description (
 "Specifies the port number, the list of port numbers or "
 "the range of port numbers the disks must have their "
 "primary path on.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskPrimPathPorts[];

[Write, Description (
 "Specifies the cage number, the list of cage numbers or "
 "the range of cage numbers the disks must be in.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
```

```
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskCages[];

[Write, Description (
 "Specifies the magazine number, the list of magazine numbers or "
 "the range of magazine numbers the disks must be in. The '1.' "
 "or '0.' that indicates the side of the cage is omitted.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskMagazines[];
[Write, Description (
 "Specifies the magazine position, the list of magazine positions or "
 "the range of magazine positions the disks must be in.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskCages",
```

```
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskMagPositions[];
[Write, Description (
 "Specifies the ID, the list of IDs or the range of IDs "
 "the disks must have.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "Empty string means that the value is not specified for "
 "this particular set of patterns. \n"
 "The string must be in one of the following formats:\n"
 "    <nb>      - nb is an integer\n"
 "    <nb>,<nb> - a list of integers\n"
 "    <nb>-<nb> - a range of integers\n"
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
   "TPD_TopLevelStorageSetting.DiskPackageModels" }]
String DiskIDs[];

[Write, Description (
 "The total number of chunklets in the disk must be "
 "greater than this value.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
 "additional candidate disks that match the pattern.\n"
 "A value of -1 means that the value is not specified for "
 "this particular set of patterns. All other negative integers "
 "are invalid.\n"),
ArrayType ( "Indexed" ),
ModelCorrespondence {
   "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
   "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
   "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
   "TPD_TopLevelStorageSetting.DiskMagazines",
   "TPD_TopLevelStorageSetting.DiskMagPositions",
   "TPD_TopLevelStorageSetting.DiskIDs",
   "TPD_TopLevelStorageSetting.DiskCages",
   "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
   "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
   "TPD_TopLevelStorageSetting.DiskRPM",
"TPD_TopLevelStorageSetting.DiskPackageModels" }]
sint32 DiskTotalChunkletGT[];

[Write, Description (
 "The total number of chunklets in the disk must be "
 "less than this value.\n"
 "Each element in this array represents one instance of "
 "pattern for candidate disks; multiple array element adds "
```

```
  "additional candidate disks that match the pattern.\n"
  "A value of -1 means that the value is not specified for "
  "this particular set of patterns. All other negative integers "
  "are invalid.\n"),
 ArrayType ( "Indexed" ),
 ModelCorrespondence {
    "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
    "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
    "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
    "TPD_TopLevelStorageSetting.DiskMagazines",
    "TPD_TopLevelStorageSetting.DiskMagPositions",
    "TPD_TopLevelStorageSetting.DiskIDs",
    "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
    "TPD_TopLevelStorageSetting.DiskCages",
    "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
    "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
    "TPD_TopLevelStorageSetting.DiskRPM",
    "TPD_TopLevelStorageSetting.DiskPackageModels" }]
 sint32 DiskTotalChunkletLT[];

 [Write, Description (
  "The total number of free or initializing chunklets in "
  "the disk must be greater than this value.\n"
  "Each element in this array represents one instance of "
  "pattern for candidate disks; multiple array element adds "
  "additional candidate disks that match the pattern.\n"
  "A value of -1 means that the value is not specified for "
  "this particular set of patterns. All other negative integers "
  "are invalid.\n"),
 ArrayType ( "Indexed" ),
 ModelCorrespondence {
    "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
    "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
    "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
    "TPD_TopLevelStorageSetting.DiskMagazines",
    "TPD_TopLevelStorageSetting.DiskMagPositions",
    "TPD_TopLevelStorageSetting.DiskIDs",
    "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
    "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
    "TPD_TopLevelStorageSetting.DiskCages",
    "TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
    "TPD_TopLevelStorageSetting.DiskRPM",
    "TPD_TopLevelStorageSetting.DiskPackageModels" }]
 sint32 DiskFreeChunkletGT[];

[Write, Description (
  "The total number of free or initializing chunklets in "
  "the disk must be less than this value.\n"
  "Each element in this array represents one instance of "
  "pattern for candidate disks; multiple array element adds "
  "additional candidate disks that match the pattern.\n"
  "A value of -1 means that the value is not specified for "
  "this particular set of patterns. All other negative integers "
  "are invalid.\n"),
 ArrayType ( "Indexed" ),
 ModelCorrespondence {
    "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
    "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
    "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
    "TPD_TopLevelStorageSetting.DiskMagazines",
    "TPD_TopLevelStorageSetting.DiskMagPositions",
    "TPD_TopLevelStorageSetting.DiskIDs",
    "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
    "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
    "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
    "TPD_TopLevelStorageSetting.DiskCages",
```

```
    "TPD_TopLevelStorageSetting.DiskRPM",
    "TPD_TopLevelStorageSetting.DiskPackageModels" }]
 sint32 DiskFreeChunkletLT[];

 [Write, Description (
  "Disks must have package models that match one of the "
  "specified list of models.\n"
  "Each element in this array represents one instance of "
  "pattern for candidate disks; multiple array element adds "
  "additional candidate disks that match the pattern.\n"
  "Empty string means that the value is not specified for "
  "this particular set of patterns. \n"
  "The string must be in one of the following formats:\n"
  " <model>     -the disk drive model/n"
  " <model>,<model>   -a list of disk drive models/n")
 ArrayType ( "Indexed" ),
 ModelCorrespondence (
"TPD_DiskDrivePackage.Model",
"TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
"TPD_TopLevelStorageSetting.DiskPrimPathPorts",
"TPD_TopLevelStorageSetting.DiskPrimPathNodes",
"TPD_TopLevelStorageSetting.DiskMagazines",
"TPD_TopLevelStorageSetting.DiskMagPositions",
"TPD_TopLevelStorageSetting.DiskIDs",
"TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
"TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
"TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
"TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
"TPD_TopLevelStorageSetting.DiskRPM",
"TPD_TopLevelStorageSetting.DiskCages" }]
String DiskPackageModels[];
[Write, Description (
"The disk must be of the specified RPM, which could either mean "
"Revolution Per Minute in the case of FC or NL drives, or "
"Relative Performance Metric in the case of SSD.\n"),
ArrayType ( "Indexed" ),
ModelCorrespondence {
"TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
"TPD_TopLevelStorageSetting.DiskPrimPathPorts",
"TPD_TopLevelStorageSetting.DiskPrimPathNodes",
"TPD_TopLevelStorageSetting.DiskMagazines",
"TPD_TopLevelStorageSetting.DiskMagPositions",
"TPD_TopLevelStorageSetting.DiskIDs",
"TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
"TPD_TopLevelStorageSetting.DiskCages",
"TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
"TPD_TopLevelStorageSetting.DiskFreeChunkletLT",
"TPD_TopLevelStorageSetting.DiskPackageModels" }]
sint32 DiskRPM[];
[Write, Description (
       "Name of the administrative domain that the volume will "
       "belong to. Default is blank which means all.\n"),
       MaxLen ( 31 )]
    String Domain;

   [Write, Description (
       "The goal amount of disk space created on each auto-grow. If this value "
       "is non-zero  it must be 8192 or bigger. A size of 0 disables the "
       "auto-grow feature.\n"
       "The default auto-grow size is fixed at 32G but the minimum "
       "auto-grow is a function of the number of online nodes in the "
       "system:"
       "                 Number of Nodes       Default     Minimum "
       "                       1-2               32G          8G   "
       "                       3-4               64G         16G   "
       "                       5-6               96G         24G   "
```

```
       "                        7-8              128G         32G   "
       "Applicable only if TemplateType equals to 3, "
       "\"DynamicStoragePool\"."),
       Units("Bytes")]
    uint64 AllocationUnit;

[Write, Description (
       "The max amount of disk space created on each auto-grow. If this value "
       "is non-zero it must be 8192 or bigger. "
       "Applicable only if TemplateType equals to 3, "
       "\"DynamicStoragePool\", and AllocationUnit is not equal "
       "to zero."),
       Units("Bytes")]
    uint64 AllocationUnitMax;

   [Write, Description (
       "The min amount of disk space created on each auto-grow. If this value "
       "is non-zero it must be 8192 or bigger. "
       "Applicable only if TemplateType equals to 3, "
       "\"DynamicStoragePool\", and AllocationUnit is not equal "
       "to zero."),
       Units("Bytes")]
    uint64 AllocationUnitMin;

   [Write, Description (
       "Issue warning alert when space allocation for a DynamicStoragePool "
       "exceeds this amount."
       "A size of 0 means no warning limit is enforced. Default is 0."
       "Applicable only if TemplateType equals to 3, "
       "\"DynamicStoragePool\"."),
       Units( "Bytes" )]
    uint64 DSPSnapSpaceGrowWarning;

    [Write, Description (
       "If true, properties that are not specified in the template "
       "will be treated as read-only and cannot be overridden "
       "by the properties in Goal parameter when invoking the "
       "TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "If false, properties that are not specified in the template "
       "can be overridden by the properties in Goal parameter when "
       "invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "Default is true.\n"
       "Applicable only if ChangeableType equals to 2, "
       "\"Changeable-Persistent\".")]
    boolean TemplateDefaultReadOnly;
[Write, Description (
       "Element type that this template can be used to create."
       "Applicable only if ChangeableType equals to 2, "
       "\"Changeable-Persistent\"."),
    ValueMap { "1", "2" },
    Values { "StorageVolume",
             "DynamicStoragePool"}]
    uint16 TemplateType;

   [Write, Description (
       "The kind of volumes that can be created by this template."
       "Default is 3, \"Any\".\n"
       "Applicable only if TemplateType equals to 1, \"StorageVolume\", "
       "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
    ValueMap { "1", "2", "3" },
    Values { "Thin Provisioned StorageVolume",
             "Common Provisioned Virtual Volumes",
             "Any" }]
    uint16 TemplateForVolumeType;
```

```
[Write, Description (
    "Size of the volume in bytes. Used only as a template property."
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\".")]
 uint64 TemplateVolumeSize;

[Write, Description (
    "Device type of the volume. Either FC (Fibre-Channel) or NL "
    "(Nearline). Used only as a template property. "
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ValueMap { "1", "2", "4"},
 Values { "FC", "NL", "SSD"}]
 uint16 TemplateVolumeDeviceType;

[Write, Description (
    "RAID type of the volume. Used only as a template property. "
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ValueMap {"0", "10", "50", "60"},
 Values {"RAID0", "RAID10", "RAID50", "RAID60"} ]
 uint16 TemplateVolumeRaidType;

[Write, Description (
    "Percent of the volume size the SD space of the volume "
    "can grow to. "
    "SpaceLimit = vol size * (1 + percent/100)"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\".")]
 uint16 TemplateSnapSpaceAllocationLimit;

[Write, Description (
    "Percent of the volume size the user space of the volume "
    "can grow to. "
    "UserSpaceLimit = vol size * (1 + percent/100)"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\", and only to thin provisioned "
    "volumes.")]
 uint16 TemplateUserSpaceAllocationLimit;

[Write, Description (
    "If true, TemplateForVolumeType value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, TemplateForVolumeType value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "TemplateForVolumeType"}]
 boolean TemplateForVolumeTypeReadOnly;

[Write, Description (
    "If true, RAID type of the volume (combination of DataRedundancyGoal "
    "and PackageRedundancyGoal values) in this template are cannot be "
    "overriden by the properties in Goal parameter when invoking the "
    "TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, RAID type of the volume (combination of DataRedundancyGoal "
    "and PackageRedundancyGoal values) in this template can be "
    "overriden by the corresponding properties in the Goal parameter "
```

```
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "DataRedundancyGoal",
          "PackageRedundancyGoal"}]
 boolean TemplateRAIDReadOnly;

[Write, Description (
    "If true, DataRedundancyGoal (RAID-10) or ParitySetSize (RAID-50/RAID-60) "
    "values in this template are read-only and cannot be overriden "
    "by the properties in the Goal parameter when invoking the "
    "TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, DataRedundancyGoal (RAID-10) or ParitySetSize (RAID-50/RAID-60) "
    "in this template can be overriden by the corresponding "
    "properties in the Goal parameter when invoking the "
    "TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "DataRedundancyGoal",
          "ParitySetSize"}]
 boolean TemplateSetSizeReadOnly;

[Write, Description (
    "If true, ExtentStripeLength value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, ExtentStripeLength value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "ExtentStripeLength"}]
 boolean TemplateRowSizeReadOnly;

[Write, Description (
    "If true, UserDataStripeDepth value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, UserDataStripeDepth value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "UserDataStripeDepth"}]
 boolean TemplateStepSizeReadOnly;

[Write, Description (
    "If true, HighAvailability value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
```

```
    "If false, HighAvailability value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "HighAvailability"}]
 boolean TemplateHighAvailabilityReadOnly;

[Write, Description (
    "If true, ChunkletLocationPreference value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, ChunkletLocationPreference value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "ChunkletLocationPreference"}]
 boolean TemplateChunkletLocationPreferenceReadOnly;

[Write, Description (
    "If true, Domain value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, Domain value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "Domain"}]
 boolean TemplateDomainReadOnly;

[Write, Description (
    "If true, all the pattern values in this template are read-only "
    "and cannot be overriden by the corresponding properties in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, all the pattern values in this template can be "
    "overriden by the corresponding properties in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Consult ModelCorrespondence for the list of pattern properties."
    "Note that this applies to all sets of pattern specified.\n"
    "Default is true.\n"
    "Applicable only if ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "TPD_TopLevelStorageSetting.DiskPrimPathPCISlots",
          "TPD_TopLevelStorageSetting.DiskPrimPathPorts",
          "TPD_TopLevelStorageSetting.DiskPrimPathNodes",
          "TPD_TopLevelStorageSetting.DiskMagazines",
          "TPD_TopLevelStorageSetting.DiskMagPositions",
          "TPD_TopLevelStorageSetting.DiskIDs",
          "TPD_TopLevelStorageSetting.DiskTotalChunkletGT",
          "TPD_TopLevelStorageSetting.DiskTotalChunkletLT",
```

```
          "TPD_TopLevelStorageSetting.DiskFreeChunkletGT",
          "TPD_TopLevelStorageSetting.DiskCages",
          "TPD_TopLevelStorageSetting.DiskPackageModels",
          "TPD_TopLevelStorageSetting.DiskFreeChunkletLT"}]
 boolean TemplatePatternReadOnly;

[Write, Description (
    "If true, SnapDSPName value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, SnapDSPName value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "SnapDSPName"}]
 boolean TemplateSnapDSPNameReadOnly;

[Write, Description (
    "If true, UserDSPName value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, UserDSPName value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "SnapDSPName"}]
 boolean TemplateUserDSPNameReadOnly;

[Write, Description (
    "If true, TemplateVolumeSize value in this template is read-only "
    "and cannot be overriden by the value in the Size parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, TemplateVolumeSize value in this template can be "
    "overriden by the Size parameter when invoking the "
    "TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "TemplateVolumeSize"}]
 boolean TemplateVolumeSizeReadOnly;

[Write, Description (
    "If true, SnapVolumeSize value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, SnapVolumeSize value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
```

```
 ModelCorrespondence {
          "SnapVolumeSize"}]
 boolean TemplateSnapVolumeSizeReadOnly;

[Write, Description (
    "If true, DeltaReservationGoal value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, DeltaReservationGoal value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "DeltaReservationGoal"}]
 boolean TemplatePctReadOnly;

[Write, Description (
    "If true, SpaceLimitWarningThreshold value of the snap space of a "
    "StorageVolume or LowSpaceWarningThreshold in a DynamicStoragePool "
    "in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, SpaceLimitWarningThreshold value of the snap space of a "
    "StorageVolume or LowSpaceWarningThreshold in a DynamicStoragePool "
    "in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "or 3, \"Common Provisioning Groups\", and ChangeableType equals "
    "to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "SpaceLimitWarningThreshold", "LowSpaceWarningThreshold" }]
 boolean TemplateSnapSpaceAllocationWarningReadOnly;

[Write, Description (
    "If true, UserSpaceAllocationWarning or UserSpaceLimitWarningThreshold "
    "value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, UserSpaceAllocationWarning or UserSpaceLimitWarningThreshold "
    "value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "TemplateForVolumeType equals to 1, \"Thin Provisioned StorageVolumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "UserSpaceLimitWarningThreshold", "UserSpaceAllocationWarning" }]
 boolean TemplateUserSpaceAllocationWarningReadOnly;

[Write, Description (
    "If true, SpaceLimit or TemplateSnapSpaceAllocationLimit value "
    "in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
```

```
    "If false, TemplateSnapSpaceAllocationLimit or SpaceLimit value "
    "in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "SpaceLimit", "TemplateSnapSpaceAllocationLimit" }]
 boolean TemplateSnapSpaceAllocationLimitReadOnly;

[Write, Description (
    "If true, TemplateUserSpaceAllocationLimit or UserSpaceLimit value "
    "in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, TemplateUserSpaceAllocationLimit or UserSpaceLimit value "
    "in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "TemplateForVolumeType equals to 1, \"Thin Provisioned StorageVolumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "UserSpaceLimit", "TemplateUserSpaceAllocationLimit"}]
 boolean TemplateUserSpaceAllocationLimitReadOnly;

[Write, Description (
    "If true, Policy value in this template is read-only "
    "and cannot be overriden by the corresponding property in "
    "the Goal parameter when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "If false, Policy value in this template can be "
    "overriden by the corresponding property in the Goal parameter "
    "when invoking the TPD_StorageConfigurationService."
    "CreateStorageVolumeFromStoragePoolWithTemplate method."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
    "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
 ModelCorrespondence {
          "Policy"}]
 boolean TemplatePolicyReadOnly;

[Write, Description (
    "If true, AllocationUnit value in this template is read-only "
    "and cannot be overriden during the creation of DSP."
    "If false, AllocationUnit value in this template can be "
    "overriden during the creation of DSP."
    "Default is true.\n"
    "Applicable only if TemplateType equals to 3, "
    "\"DynamicStoragePool\", and ChangeableType equals to 2, "
    "\"Changeable-Persistent\"."),
 ModelCorrespondence {
          "AllocationUnit"}]
 boolean TemplateAllocationUnitReadOnly;

[Write, Description (
    "If true, DSPSnapSpaceGrowWarning or SpaceLimitWarningThreshold "
    "value in this template is read-only "
    "and cannot be overriden during the creation of DSP."
    "If false, DSPSnapSpaceGrowWarning or SpaceLimitWarningThreshold "
    "value in this template can be "
```

```
       "overriden during the creation of DSP."
       "Default is true.\n"
       "Applicable only if TemplateType equals to 3, "
       "\"DynamicStoragePool\", and ChangeableType equals to 2, "
       "\"Changeable-Persistent\"."),
    ModelCorrespondence {
             "SpaceLimitWarningThreshold", "DSPSnapSpaceGrowWarning"}]
    boolean TemplateDSPSnapSpaceGrowWarningReadOnly;

   [Write, Description (
       "If true, GeometrySectorsPerTrack value in this template is read-only "
       "and cannot be overriden by the corresponding property in "
       "the Goal parameter when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "If false, GeometrySectorsPerTrack value in this template can be "
       "overriden by the corresponding property in the Goal parameter "
       "when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "Default is true.\n"
       "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
       "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
    ModelCorrespondence {
             "GeometrySectorsPerTrack"}]
    boolean TemplateGeometrySectorsPerTrackReadOnly;
   [Write, Description (
       "If true, GeometryHeadsPerCylinder value in this template is read-only "
       "and cannot be overriden by the corresponding property in "
       "the Goal parameter when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "If false, GeometryHeadsPerCylinder value in this template can be "
       "overriden by the corresponding property in the Goal parameter "
       "when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "Default is true.\n"
       "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
       "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
    ModelCorrespondence {
             "GeometryHeadsPerCylinder"}]
    boolean TemplateGeometryHeadsPerCylinderReadOnly;

   [Write, Description (
       "If true, GeometrySectorSize value in this template is read-only "
       "and cannot be overriden by the corresponding property in "
       "the Goal parameter when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "If false, GeometrySectorSize value in this template can be "
       "overriden by the corresponding property in the Goal parameter "
       "when invoking the TPD_StorageConfigurationService."
       "CreateStorageVolumeFromStoragePoolWithTemplate method."
       "Default is true.\n"
       "Applicable only if TemplateType equals to 1, \"Virtual Volumes\", "
       "and ChangeableType equals to 2, \"Changeable-Persistent\"."),
    ModelCorrespondence {
             "GeometrySectorSize"}]
    boolean TemplateGeometrySectorSizeReadOnly;

    [Description ("The time specifies when the volume will expire and "
        "can be deleted.") ]
    datetime ExpirationTime;

    [Description ("The time specifies the volume needs to be retained and "
        "can not remove until the retention time is expired.") ]
    datetime RetentionTime;
};

// ===================================================================
```

```
// 3PAR StorageSetting
// ===================================================================
[Description (
  "3PAR StorageSetting.")]
class TPD_StorageSetting : TPD_TopLevelStorageSetting
{
};

// ===================================================================
// 3PAR StorageSetting
// ===================================================================
[Description (
  "3PAR StorageSetting that describes the current setting of a StorageVolume.")]
class TPD_VolumeSetting : TPD_TopLevelStorageSetting
{
};

// ===================================================================
// 3PAR ElementSettingData
// ===================================================================
[Association,
    Description (
        "TPD_ElementSettingData is an association between "
        "StorageVolume and StorageSetting."
        "This association is used to figure out the desired setting "
        "data is used when the volume is created.") ]
class TPD_ElementSettingData: CIM_ElementSettingData
{
    [Override ( "ManagedElement" ), Description (
          "The StorageVolume.")]
    TPD_StorageVolume REF ManagedElement;

    [Override ( "SettingData" ), Description (
          "The setting data used when volume is allocated")]
    TPD_VolumeSetting REF SettingData;
};

// ===================================================================
// 3PAR StorageConfigurationService
// ===================================================================
[Description
     ("The StorageConfigurationService provides methods that allow "
      "a client to create, modify and delete storage pools and "
      "volumes.")]
class TPD_StorageConfigurationService : SNIA_StorageConfigurationService
{
      [Override ("CreateOrModifyStoragePool"), Description (
          "Starts a job to create (or modify) a StoragePool. The "
          "StoragePool will be (or must be) scoped to the same System "
          "as this Service. One of the parameters for this method is "
          "Size. As an input parameter, Size specifies the desired "
          "size of the pool. As an output parameter, it specifies the "
          "size achieved. Space is taken from either or both of the "
          "specified input StoragePools and StorageExtents (InPools "
          "and InExtents). The capability requirements that the Pool "
          "must support are defined using the Goal parameter. If the "
          "requested pool size cannot be created, no action will be "
          "taken, the Return Value will be 4097/0x1001, and the output "
          "value of Size will be set to the nearest possible size. If "
          "0 is returned, then the task completed successfully and the "
          "use of ConcreteJob was not required. If the task will take "
          "some time to complete, a ConcreteJob will be created and "
          "its reference returned in the output parameter Job. "
     "For 3PAR InServ provider, only TPD_DynamicStoragePool "
     "can be created from a concrete pool."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
```

```
        "4097", "4098..32767", "32768..65535" },
     Values { "Job Completed with No Error", "Not Supported",
        "Unknown", "Timeout", "Failed", "Invalid Parameter",
        "In Use", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Size Not Supported", "Method Reserved", "Vendor Specific" }]
uint32 CreateOrModifyStoragePool (
      [IN, Description (
          "A end user relevant name for the pool being created. If "
          "NULL, then a system supplied default name can be used. "
          "The value will be stored in the 'ElementName' property "
          "for the created pool. If not NULL, this parameter will "
          "supply a new name when modifying an existing pool.")]
   string ElementName,
      [IN ( false ), OUT, Description (
          "Reference to the job (may be null if job completed).")]
   CIM_ConcreteJob REF Job,
      [IN, Description (
          "Reference to an instance of StorageSetting that defines "
          "the desired capabilities of the StoragePool. If set to a "
          "null value, the default configuration from the source "
          "pool will be used. If not NULL, this parameter will "
          "supply a new Goal setting when modifying an existing "
          "pool. ")]
   CIM_StorageSetting REF Goal,
      [IN, OUT, Description (
          "As an input parameter this specifies the desired pool "
          "size in bytes. As an output parameter this specifies the "
          "size achieved. "),
       Units ( "Bytes" )]
   uint64 Size,
      [IN, Description (
          "Array of strings containing representations of "
          "references to CIM_StoragePool instances, that are used "
          "to create the Pool or modify the source pools.")]
   string InPools[],
      [IN, Description (
          "Array of strings containing representations of "
          "references to CIM_StorageExtent instances, that are used "
          "to create the Pool or modify the source extents.")]
   string InExtents[],
      [IN, OUT, Description (
          "As an input parameter: if null, creates a new "
          "StoragePool. If not null, modifies the referenced Pool. "
          "When returned, it is a reference to the resulting "
          "StoragePool.")]
   CIM_StoragePool REF Pool);

   [Override ("CreateOrModifyElementFromStoragePool"), Description (
       "Start a job to create (or modify) a specified element (for "
       "example a StorageVolume or StorageExtent) from a "
       "StoragePool. One of the parameters for this method is Size. "
       "As an input parameter, Size specifies the desired size of "
       "the element. As an output parameter, it specifies the size "
       "achieved. Space is taken from the input StoragePool. The "
       "desired settings for the element are specified by the Goal "
       "parameter. If the requested size cannot be created, no "
       "action will be taken, and the Return Value will be "
       "4097/0x1001. Also, the output value of Size is set to the "
       "nearest possible size. If 0 is returned, the function "
       "completed successfully and no ConcreteJob instance was "
       "required. If 4096/0x1000 is returned, a ConcreteJob will be "
       "started to create the element. The Job's reference will be "
       "returned in the output parameter Job."
"A proprietary out parameter ResultString is added for 3PAR."),
    ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
```

```
        "4097", "4098..32767", "32768..65535" },
     Values { "Job Completed with No Error", "Not Supported",
        "Unknown", "Timeout", "Failed", "Invalid Parameter",
        "In Use", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Size Not Supported", "Method Reserved", "Vendor Specific" }]
  uint32 CreateOrModifyElementFromStoragePool(
       [IN, Description (
           "A end user relevant name for the element being created. "
           "If NULL, then a system supplied default name can be "
           "used. The value will be stored in the 'ElementName' "
           "property for the created element. If not NULL, this "
           "parameter will supply a new name when modifying an "
           "existing element.")]
     string ElementName,
       [IN, Description (
           "Enumeration indicating the type of element being created "
           "or modified. If the input parameter TheElement is "
           "specified when the operation is a 'modify', this type "
           "value must match the type of that instance."),
        ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
        "..", "32768..65535" },
        Values { "Unknown", "Reserved", "StorageVolume",
           "StorageExtent", "LogicalDisk",
           "ThinlyProvisionedStorageVolume",
           "ThinlyProvisionedLogicalDisk",
           "ThinlyProvisionedAllocatedStoragePool",
           "ThinlyProvisionedQuotaStoragePool",
           "ThinlyProvisionedLimitlessStoragePool",
           "DMTF Reserved","Vendor Specific" }]
     uint16 ElementType,
       [IN ( false ), OUT, Description (
           "Reference to the job (may be null if job completed).")]
     CIM_ConcreteJob REF Job,
       [IN, Description (
           "The requirements for the element to maintain. If set to "
           "a null value, the default configuration from the source "
           "pool will be used. This parameter should be a reference "
           "to a Setting or Profile appropriate to the element being "
           "created. If not NULL, this parameter will supply a new "
           "Goal when modifying an existing element.")]
     CIM_ManagedElement REF Goal,
       [IN, OUT, Description (
           "As an input parameter Size specifies the desired size. "
           "If not NULL, this parameter will supply a new size when "
           "modifying an existing element. As an output parameter "
           "Size specifies the size achieved."),
        Units ( "Bytes" )]
     uint64 Size,
       [IN, Description (
           "The Pool from which to create the element. This "
           "parameter must be set to null if the input parameter "
           "TheElement is specified (in the case of a 'modify' "
           "operation).")]
     CIM_StoragePool REF InPool,
       [IN, OUT, Description (
           "As an input parameter: if null, creates a new element. "
           "If not null, then the method modifies the specified "
           "element. As an output parameter, it is a reference to "
           "the resulting element.")]
     CIM_LogicalElement REF TheElement,
       [OUT, Description (
           "A descriptive text of the result of the operation.")]
     string ResultDescription);

     [Description (
```

```
       "A 3PAR-specific method that can be used to create a "
"thin-provisioned or full-provisioned volume from a "
"TPD_DynamicStoragePool. "
"To create a thin-provisioned volume, Goal.ThinlyProvisioned "
"property must be set to true. If Goal is NULL or if "
"Goal.ThinlyProvisioned is false, then a full-provisoned "
"volume will be created. "
       "One of the parameters for this method is Size. "
       "As an input parameter, Size specifies the desired size of "
       "the element. As an output parameter, it specifies the size "
       "achieved. Space is taken from the input StoragePool. The "
       "desired settings for the element are specified by the Goal "
       "parameter. If the requested size cannot be created, no "
       "action will be taken, and the Return Value will be "
       "4097/0x1001. Also, the output value of Size is set to the "
       "nearest possible size. If 0 is returned, the function "
       "completed successfully and no ConcreteJob instance was "
       "required. If 4096/0x1000 is returned, a ConcreteJob will be "
       "started to create the element. The Job's reference will be "
       "returned in the output parameter Job."),
    ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
       "4097", "4098..32767", "32768..65535" },
    Values { "Job Completed with No Error", "Not Supported",
       "Unknown", "Timeout", "Failed", "Invalid Parameter",
       "In Use", "DMTF Reserved",
       "Method Parameters Checked - Job Started",
       "Size Not Supported", "Method Reserved", "Vendor Specific" }]
uint32 TPD_CreateOrModifyElementFromStoragePools(
      [IN, Description (
          "A end user relevant name for the element being created. "
          "If NULL, then a system supplied default name can be "
          "used. The value will be stored in the 'ElementName' "
          "property for the created element. If not NULL, this "
          "parameter will supply a new name when modifying an "
          "existing element.")]
   string ElementName,
      [IN, Description (
          "Enumeration indicating the type of element being created "
          "or modified. If the input parameter TheElement is "
          "specified when the operation is a 'modify', this type "
          "value must match the type of that instance."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
       "..", "32768..65535" },
       Values { "Unknown", "Reserved", "StorageVolume",
          "StorageExtent", "LogicalDisk",
          "ThinlyProvisionedStorageVolume",
          "ThinlyProvisionedLogicalDisk",
          "ThinlyProvisionedAllocatedStoragePool",
          "ThinlyProvisionedQuotaStoragePool",
          "ThinlyProvisionedLimitlessStoragePool",
          "DMTF Reserved","Vendor Specific" }]
   uint16 ElementType,
      [IN ( false ), OUT, Description (
          "Reference to the job (may be null if job completed).")]
   CIM_ConcreteJob REF Job,
      [IN, Description (
          "The requirements for the element to maintain. If set to "
          "a null value, the default configuration from the source "
          "pool will be used. This parameter should be a reference "
          "to a Setting or Profile appropriate to the element being "
          "created. If not NULL, this parameter will supply a new "
          "Goal when modifying an existing element.")]
   CIM_StorageSetting REF Goal,
      [IN, OUT, Description (
          "As an input parameter Size specifies the desired size. "
          "If not NULL, this parameter will supply a new size when "
```

```
            "modifying an existing element. As an output parameter "
            "Size specifies the size achieved."),
         Units ( "Bytes" )]
     uint64 Size,
        [IN, Description (
            "Array of strings containing representations of "
            "references to CIM_StoragePool instances, that are used "
            "to create the element. "
     "The first element contains reference to a StoragePool from "
     "which the user space of the StorageVolume is allocated. The "
     "second element, if present, contains reference to a "
     "StoragePool from which the snap space of the StorageVolume "
     "is allocated.")]
     string InPools[],
        [IN, OUT, Description (
            "As an input parameter: if null, creates a new element. "
            "If not null, then the method modifies the specified "
            "element. As an output parameter, it is a reference to "
            "the resulting element.")]
     CIM_LogicalElement REF TheElement,
        [OUT, Description (
            "A descriptive text of the result of the operation.")]
     string ResultDescription);

     [Description (
         "Start a job to create a specified StorageVolume from a "
       "StoragePool using a persistent StorageSetting as a template, which reference
 "
         "is specified in Template. The Template parameter must be specified and
cannot "
         "be NULL. Goal is an embedded instance string of a StorageSetting. This "
         "contains properties which values should override those in Template. "
         "Another one of the parameters is Size. As an input parameter, Size "
         "specifies the desired size of the element. As an output parameter, "
         "it specifies the size achieved. Space is taken from the input "
       "StoragePool. The desired settings for the element are specified by the Goal
 "
         "parameter. If the requested size cannot be created, no "
         "action will be taken, and the Return Value will be "
         "4097/0x1001. Also, the output value of Size is set to the "
         "nearest possible size. If Size is not specified, then the "
         "Size property in Template is used. If 0 is returned, the function "
         "completed successfully and no ConcreteJob instance was "
         "required. If 4096/0x1000 is returned, a ConcreteJob will be "
         "started to create the element. The Job's reference will be "
         "returned in the output parameter Job."),
      ValueMap { "0", "1", "2","4", "4096",
         "4097", "32768", "32769", "32770" },
      Values { "Job Completed with No Error", "Not Supported",
         "Unknown", "Failed", "Invalid Parameter",
         "Size Not Supported", "Cannot override template parameters",
         "Size in Goal and Template are both NULL",
  "Template does not exist" }]
  uint32 CreateStorageVolumeFromStoragePoolWithTemplate(
        [IN, Description (
            "A end user relevant name for the element being created. "
            "If NULL, then a system supplied default name can be "
            "used. The value will be stored in the 'ElementName' "
            "property for the created element. If not NULL, this "
            "parameter will supply a new name when modifying an "
            "existing element.")]
     string ElementName,
        [IN, Description (
            "A reference to an existing instance of persistent "
            "TPD_StorageSetting to be used as the template for "
            "creation. This must be specified and cannot "
```

```
            "be NULL.")]
    CIM_StorageSetting REF Template,
       [IN, Description (
           "If provided, the StorageSetting properties to be created "
           "or modified for the volume. Properties specified here will "
           "override those specified in Template. If override is not "
           "possible, an error code of 32768 will be returned. This is the "
           "string representation of an embedded instance and does not exist "
           "in the provider infrastructure but is maintained by the client. "
           "If not provided, all properties specified in Template will be "
           "used to create the volume."),
        EmbeddedInstance("TPD_StorageSetting")]
    String Goal,
       [IN, OUT, Description (
           "As an input parameter Size specifies the desired size. "
           "If not NULL, this parameter will supply a new size when "
           "modifying an existing element. As an output parameter "
           "Size specifies the size achieved. If this is not provided, "
           "Size property in Template will be used. This and Template "
           "size property cannot both be NULL "),
        Units ( "Bytes" )]
    uint64 Size,
       [IN, Description (
           "The Pool from which to create the element. If "
           "null, the pool all-FC will be used.")]
    CIM_StoragePool REF InPool,
       [OUT, Description (
           "As an input parameter, this must be NULL. "
           "As an output parameter, it is a reference to "
           "the resulting StorageVolume.")]
    TPD_StorageVolume REF TheElement,
       [OUT, Description (
           "A descriptive text of the result of the operation.")]
    string ResultDescription);

    [Description (
        "3PAR specific method to remove one or more StorageVolumes. "
        "This method allows for multiple volumes to be removed "
        "in one call. The freed space is returned to the source "
        "StoragePool. If 0 is returned, the function completed "
        "successfully."
        "Return code of Partial Success (32768) means that only "
        "some, but not all, of the StorageVolumes are successfully "
        "removed."),
     ValueMap { "0", "1", "2", "3", "4", "5", "6", "32768" },
     Values { "Job Completed with No Error", "Not Supported",
        "Unknown", "Timeout", "Failed", "Invalid Parameter",
        "In Use", "Partial Success" }]
uint32 TPD_ReturnToStoragePool(
       [IN, Description (
           "If true, operation will not block until the StorageVolume "
           "is removed. Removal of StorageVolume is performed in "
           "the background. Only snapshot StorageVolume of CopyType "
           "UnSyncAssoc and ReplicaType of After Delta can be removed "
           "with this option; attempt to remove StorageVolume of "
           "any other type results in an error. Default is false")]
    boolean NoWait,
       [IN, Description (
         "References to one or more StorageVolumes to return to the StoragePool.")]

    TPD_StorageVolume REF TheElements[]);

    [Description (
        "Start a job to update the actual space used by delta "
"snapshot volumes. This will cause the system to start updating "
"the actual space allocated from TPD_DeltaReplicaStoragePool to "
```

```
"the specified delta snapshot(s). If one or more delta snapshots "
"are specified in the SnapshotVolumes array, then only the specified "
"volumes will be updated. If none are specified, all delta snapshot "
"volumes will be updated."),
    ValueMap { "4", "4096", "32768" },
    Values { "Failed","Method Parameters Checked - Job Started",
       "One or more volume do not exist or not delta snapshots" }]
uint32 UpdateDeltaSnapshotSpace(
      [IN, Description (
          "References to delta snapshot volumes that are to be "
   "updated. If NULL, then update all delta snapshots "
   "in the system.")]
   TPD_StorageVolume REF SnapshotVolumes[],
      [IN ( false ), OUT, Description (
          "Reference to the job created.")]
   TPD_ConcreteJob REF Job);

   [Override ("CreateReplica"), Description (
       "Start a job to create a new storage object which is a "
       "replica of the specified source storage object. "
       "(SourceElement). Note that using the input paramter, "
       "CopyType, this function can be used to instantiate the "
       "replica, and to create an ongoing association between the "
       "source and replica. If 0 is returned, the function "
       "completed successfully and no ConcreteJob instance is "
       "created. If 4096/0x1000 is returned, a ConcreteJob is "
       "started, a reference to which is returned in the Job output "
       "parameter."
"For 3PAR, ElementName parameter is required."),
    ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
       "4097..32767", "32768..65535" },
    Values { "Job Completed with No Error", "Not Supported",
       "Unknown", "Timeout", "Failed", "Invalid Parameter",
       "In Use", "DMTF Reserved",
       "Method Parameters Checked - Job Started", "Method Reserved",
       "Vendor Specific" }]
uint32 CreateReplica(
      [IN, Description (
          "A end user relevant name for the element being created. "
          "The value will be stored in the 'ElementName' "
          "property for the created element."
   "This parameter is required and cannot be NULL.")]
   string ElementName,
      [IN ( false ), OUT, Description (
          "Reference to the job (may be null if job completed).")]
   CIM_ConcreteJob REF Job,
      [Required, IN, Description (
          "The source storage object which may be a StorageVolume "
          "or storage object.")]
   CIM_LogicalElement REF SourceElement,
      [IN ( false ), OUT, Description (
          "Reference to the created target storage element (i.e., "
          "the replica).")]
   CIM_LogicalElement REF TargetElement,
      [IN, Description (
          "The definition for the StorageSetting to be maintained "
          "by the target storage object (the replica).")]
   CIM_StorageSetting REF TargetSettingGoal,
      [IN, Description (
          "The underlying storage for the target element (the "
          "replica) will be drawn from TargetPool if specified, "
          "otherwise the allocation is implementation specific.")]
   CIM_StoragePool REF TargetPool,
      [IN, Description (
          "CopyType describes the type of copy that will be made. "
          "Values are: \n"
```

```
            "Async: Create and maintain an asynchronous copy of the "
            "source. \n"
            "Sync: Create and maintain a synchronized copy of the "
            "source. \n"
            "UnSyncAssoc: Create an unsynchronized copy and maintain "
            "an association to the source. \n"
            "UnSyncUnAssoc: Create unassociated copy of the source "
            "element."),
         ValueMap { "2", "3", "4", "5", "..", "32768..65535" },
         Values { "Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc",
            "DMTF Reserved", "Vendor Specific" }]
     uint16 CopyType);
};

// ===================================================================
// 3PAR StorageConfigurationCapabilities
// ===================================================================
[Description
     ("3PAR StorageConfigurationCapabilities describes the storage "
      "elements and methods supported by StorageConfigurationService.")]
class TPD_StorageConfigurationCapabilities :
                           SNIA_StorageConfigurationCapabilities
{
      [Override ( "SupportedStorageElementFeatures" ),
       Description (
          "Enumeration indicating features supported by the Storage "
          "Element methods." ),
       ValueMap { "2", "3", "4", "5", "6", "7", "8", "9", "10",
          "11", "12", "13", "14", "15", "16", "17", "18", "..",
          "0x8000..0xFFFF" },
       Values { "StorageExtent Creation", "StorageVolume Creation",
          "StorageExtent Modification",
          "StorageVolume Modification", "Single InPool",
          "Multiple InPools", "LogicalDisk Creation",
          "LogicalDisk Modification", "InElements",
          "Storage Element QoS Change",
          "Storage Element Capacity Expansion",
          "Storage Element Capacity Reduction",
          "StorageVolume To StoragePool Relocation",
          "StoragePool To StoragePool Relocation",
          "StorageVolume To StorageExtent Relocation",
          "StoragePool To StorageExtent Relocation",
          "LogicalDisk To StorageExtent Relocation",
          "DMTF Reserved", "Vendor Specific" },
       ModelCorrespondence {

"CIM_StorageConfigurationService.CreateOrModifyElementFromStoragePool.ElementType",
        "CIM_StorageConfigurationService.CreateOrModifyElementFromStoragePool.InPool",

          "CIM_StorageConfigurationService",
          "CreateOrModifyElementFromElements.InElements",
         "CIM_StorageConfigurationService.CreateElementsFromStoragePool.ElementType"
 }]
   uint16 SupportedStorageElementFeatures[];
};

// ===================================================================
// 3PAR StoragePoolConfigurationCapabilities
// ===================================================================
[Description
     ("3PAR StoragePoolConfigurationCapabilities describes the storage "
      "elements and methods supported by a StoragePool.")]
class TPD_StoragePoolConfigurationCapabilities :
                           SNIA_StorageConfigurationCapabilities
{
};
```

```
// ===================================================================
// 3PAR CapabilitiesOfStorageConfigurationService
// ===================================================================
[Association,
    Description (
        "TPD_CapabilitiesOfStorageConfigurationService is an association "
        "between StorageConfigurationCapabilities or "
        "StorageReplicationCapabilities and StorageConfigurationService. "
        "This association is used to query the features and functions "
        "supported by StorageConfigurationService.") ]
class TPD_CapabilitiesOfStorageConfigurationService: CIM_ElementCapabilities
{
    [Override ( "ManagedElement" ), Description (
          "The StorageConfigurationService.")]
    TPD_StorageConfigurationService REF ManagedElement;

    [Override ( "Capabilities" ), Description (
          "The configuration capabilities that the "
   "TPD_StorageConfigurationService can support. This can "
          "be either TPD_StorageConfigurationCapabilities or "
          "TPD_StorageReplicationCapabilities.")]
    CIM_Capabilities REF Capabilities;
};

// ===================================================================
// 3PAR CapabilitiesOfStoragePoolConfiguration
// ===================================================================
[Association,
    Description (
        "TPD_CapabilitiesOfStoragePoolConfiguration is an association "
        "between StoragePoolConfigurationCapabilities and "
        "StoragePool.  This association is used to query "
        "the features and functions supported by "
        "StoragePool.") ]
class TPD_CapabilitiesOfStoragePoolConfiguration: CIM_ElementCapabilities
{
    [Override ( "ManagedElement" ), Description (
          "The StoragePool.")]
    CIM_StoragePool REF ManagedElement;

    [Override ( "Capabilities" ), Description (
          "The configuration capabilities (RAID types) that the StoragePool can
support.")]
    TPD_StoragePoolConfigurationCapabilities REF Capabilities;
};

// ===================================================================
// 3PAR System and ControllerConfigurationService association.
// ===================================================================
[Association,
    Description (
       "TPD_HostedStorageConfigurationService is an association "
       "between TPD_StorageSystem and TPD_StorageConfigurationService. "
       "The cardinality of this association is 1-to-1. A System may "
       "host only one StorageConfigurationService. ")]
class TPD_HostedStorageConfigurationService : CIM_HostedService {
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ StorageSystem.")]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ), Weak, Description (
          "The StorageConfigurationService hosted on the 3PAR InServ.")]
   TPD_StorageConfigurationService REF Dependent;
};
```

```
// ===================================================================
// 3PAR Frontend FC Port
// ===================================================================
[Description ("3PAR InServ FC front end Port ")]
class TPD_FCPort : CIM_FCPort
{
    [Description (" "),
        ValueMap {"0", "1", "2", "3", "4", "7", "8",
            "9", "10", "11", "12", "13"},
        Values {"Config Wait", "Alpa Wait", "Login Wait", "OK", "Loss Sync",
            "Nonparticipate", "Core Dump", "Offline", "FW Dead",
            "Link Idle For Reset", "DHCP In Progress", "Pending Reset"} ]
        uint16 OtherOperationalStatus;

    [Description ("Topology of the port"),
        ValueMap {"0", "1", "2", "3", "4"},
        Values {"Private loop", "Public loop", "Point to point",
            "Fabric", "Unknown"} ]
        uint16 Topology;

    [Description ("Daisy chain configuration of the port"),
        ValueMap {"0", "1", "2"},
        Values {"Unknown", "Valid", "Invalid"} ]
        uint16 Configuration;

    [Description ("Mode of the port"),
        ValueMap {"0", "1", "2"},
        Values {"Suspended", "Target", "Initiator"} ]
        uint16 Mode;

    [Description ("Shows if the mode change is allowed or not.")]
        boolean ModeChangeAllowed;

    [Description ("Node WWN of the port.") ]
      uint64 NodeWWN;

    [Description ("Describes device type this port is connected to."),
        ValueMap {"0", "1", "2", "3", "4"},
        Values {"Free", "Host", "Disk", "IPort", "FCRC"} ]
        uint16 DeviceTypeConnected;

    [Description ("List of devices, this port is connected to.") ]
        string ConnectedTo;

    [Description ("Persona of the port.") ]
        uint16 Persona;

    [Description ("Connection type."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Unknown", "Loop", "Point-to-Point", "Loop Point-to-Point"} ]
        uint16 ConnectionType;

    [Description ("Connection mode, can be one of: "
               "  Disk - disk connection "
               "  Host - host connection "
               "  RCFC - RCFC connection "
               "  Peer - data migration connection."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Disk", "Host", "RCFC", "Peer"} ]
        uint16 ConnectionMode;

    [Description ("Class 2 support."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Unknown", "ack0", "ack1", "Disable"} ]
        uint16 Class2Support;
```

```
    [Description ("VLUN Change Notification (VCN) support."),
        ValueMap {"0", "1", "2"},
        Values {"Unknown", "Enable", "Disable"} ]
        uint16 VCN;

    [Description ("Interrupt coalescing support."),
        ValueMap {"0", "1", "2"},
        Values {"Unknown", "Enable", "Disable"} ]
        uint16 InterruptCoalescing;
};

// ===================================================================
// 3PAR FCAL Port
// ===================================================================
// [Description ("3PAR InServ FCAL Port ")]
// class TPD_FCALPort : CIM_FCPort
//{
//    [Description (" "),
//        ValueMap {"0", "1", "2", "4", "7", "8",
//            "9", "10", "11"},
//        Values {"config wait", "alpa wait", "login wait", "loss sync",
//            "nonparticipate", "core dump", "offline",
//            "fw dead", "link idle for reset"} ]
//        uint16 OtherOperationalStatus;

//    [Description ("Node WWN the FCAL port is connected to.") ]
//      uint64 NodeWWN;
//};

// ===================================================================
// 3PAR SCSI Protocol Controller And FCPort association
// ===================================================================
[Association,
    Description (
    "3PAR InServ SCSI protocol controller and FC port"
    " association. ")]
class TPD_ControllerForPort : CIM_ProtocolControllerForPort
{
    [Override ("Antecedent"),
        Description ("The SCSI Protocol Controller.") ]
    TPD_SCSIController REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The FCPort controlled by the SCSI protocol controller.") ]
    TPD_FCPort REF Dependent;
    // CIM_FCPort REF Dependent; -- reenable this line if we support
TPD_SCSIInitTargetLUPath
};


// ===================================================================
// 3PAR SCSI Protocol Controller And StorageVolume association
// ===================================================================
[Association,
    Description (
    "3PAR InServ SCSI protocol controller and storage volume "
    "association. ")]
class TPD_ControllerForUnit : CIM_ProtocolControllerForUnit
{
    [Override ("Antecedent"),
        Description ("The SCSI Protocol Controller.") ]
    TPD_SCSIController REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The storage volume controlled by the SCSI protocol controller.") ]
    TPD_StorageVolume REF Dependent;
```

```
};

// ===================================================================
// 3PAR SCSIProtocolEndpoint (front-end FC port)
// ===================================================================
[Description
     ("3PAR InServ SCSIProtocolEndpoint (front-end FC port).  "
      "This class represent FC ports that are connecting to hosts or "
      "fabric or disk. ")]
class TPD_SCSIProtocolFCEndpoint : CIM_SCSIProtocolEndpoint
{
    [Description ("Connection mode, can be one of: "
                "  Disk - disk connection "
                "  Host - host connection "
                "  RCFC - RCFC connection "
                "  Peer - data migration connection."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Disk", "Host", "RCFC", "Peer"} ]
        uint16 ConnectionMode;
};

// ===================================================================
// 3PAR SCSIProtocolEndpoint on FC-AL card (connects to initiator
// endpoint on the node controller)
// ===================================================================
//[Description
//     ("3PAR InServ SCSIProtocolEndpoint that resides on the "
//      "FC-AL card. This endpoint is connected to the initiator FC "
//      "endpoint the node controller.")]
//class TPD_SCSIProtocolFCALEndpoint : CIM_SCSIProtocolEndpoint
//{
//};

// ===================================================================
// 3PAR SCSIProtocolController And SCSIProtocolEndpoint association.
// NOTE: In SMI-S 1.1, this association replaces ControllerForPort
// association (SMI-S 1.02).
// ===================================================================
[Association,
    Description (
    "3PAR InServ SCSI protocol controller and the front end "
    "target SCSIProtocolEndpoint association. ")]
class TPD_SAPAvailableForElement : CIM_SAPAvailableForElement
{
    [Override ("AvailableSAP"), Description (
        "The SCSIProtocolEndpoint (front end port).") ]
    TPD_SCSIProtocolFCEndpoint REF AvailableSAP;

    [Override ("ManagedElement"), Description (
      "The SCSI Protocol Controller (service).") ]
    TPD_SCSIController REF ManagedElement;
};

// ===================================================================
// DeviceSAPImplementation - association between SCSIEndpoint and
//  FCPort
// ===================================================================
[Association,
    Description (
       "An association between a 3PAR SCSI protocol endpoint and its "
       "implementation on FC port.")]
class TPD_FCPortSCSIEndpointImplementation: CIM_DeviceSAPImplementation {

      [Override ( "Antecedent" ), Description (
          "The FC port.")]
   TPD_FCPort REF Antecedent;
```

```
   // CIM_FCPort REF Antecedent; -- reenable this line if we support
TPD_SCSIInitTargetLUPath

      [Override ( "Dependent" ), Description (
          "The SCSIProtocolEndpoint implemented using the FC port.")]
   TPD_SCSIProtocolFCEndpoint REF Dependent;
   // CIM_SCSIProtocolEndpoint REF Dependent; -- reenable this line if we support
TPD_SCSIInitTargetLUPath
};

// ===================================================================
// ControllerConfigurationService
// ===================================================================
[Description
     ("The ControllerConfigurationService provides methods that allow "
      "a client to export and delete VLUNs in an InServ.")]
class TPD_ControllerConfigurationService : CIM_ControllerConfigurationService
{
      [Override ("ExposePaths"), Description (
          "3PAR version of the ExposePaths operation, adding an extra "
     "property ResultDescription to better describe the result "
     "of the expose path operation. \n"
     "\n"
          "Expose a list of SCSI logical units (such as RAID volumes "
          "or tape drives) to a list of initiators through a list of "
          "target ports, through one or more SCSIProtocolControllers "
          "(SPCs). \n"
          "\n"
          "The parameters for this method are: Job - null if no job "
          "created, otherwise this is a reference to the job. LUNames "
          "- the list of names of the logical units to use. "
          "InitiatorPortIDs - the names of the initiator ports to use. "
          "TargetPortIDs - the names of the target ports to use. "
          "DeviceNumbers - the device numbers (LUNs) to use. "
          "DeviceAccesses - permissions for the logical units. "
          "ProtocolControllers - SPCs involved in this operation. \n"
          "\n"
          "There are two modes of operation, create and modify. If a "
          "NULL value is passed in for the SPC, then the "
          "instrumentation will create at least one SPC that satisfies "
          "the request. Since "
     "CIM_ProtocolControllerMaskingCapabilities.OneHardwareIDPerView "
     "for the 3PAR InServ is true, more than one SPC could be created "
     "if more than one initiatorID was passed in. If an SPC is "
          "passed in, then the instrumentation attempts to add the new "
          "paths to the existing SPC. \n"
          "\n"
          "For creating an SPC, all parameters, except ProtocolControllers, "
     "MUST be specified since SPCAllowsNoLUs, SPCAllowsNoTargets and "
     "SPCAllowsNoInitiators properties are all false, and "
     "ClientSelectableDeviceNumbers property is true, for a 3PAR "
     "InServ array. \n"
          "\n"
          "The LUNames, DeviceNumbers, and DeviceAccesses parameters "
          "are mutually indexed arrays - any element in DeviceNumbers "
          "or DeviceAccesses will set a property relative to the "
          "LogicalDevice instance named in the corresponding element "
          "of LUNames. LUNames, DeviceNumbers and DeviceAccesses MUST "
     "have the same number of elements. If these conditions are "
     "not met, the instrumentation MUST return a 'Invalid Parameter' "
          "status or a CIM_Error. \n"
          "\n"
          "For modifying an SPC, the 3PAR InServ array supports only "
          "the Add LUs to a view use case."
          "Add LUs to a view requires that the LUNames and DeviceNumbers "
     "parameters not be null and that the InitiatorIDs and TargetPortIDs "
```

```
        "parameters be null. \n"
        "\n"
        "The relevant rules of SCSI semantics are: \n"
        "- an SPC MAY NOT be exposed through a particular "
        "host/target port pair that is in use by another SPC. (In "
        "other words, an SPC and its associated logical units and "
        "ports together correspond to the logical unit inventory "
        "provided by SCSI REPORT LUNS and INQUIRY commands) \n"
        "- each LogicalDevice associated to an SPC MUST have a "
        "unique ProtocolControllerForUnit DeviceNumber (logical unit "
        "number) \n"
        "The instrumentation MUST report an error if the client "
        "request would violate one of these rules. \n"
        "\n"
        "If the instrumentation provides PrivilegeManagementService, "
        "the results of setting DeviceAccesses MUST be synchronized "
        "with PrivilegeManagementService as described in the "
        "ProtocolControllerForUnit DeviceAccess description."),
     ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
        "4097", "4098", "4099", "4100", "4101", "4102", "4103",
        "4104..32767", "32768..65535" },
     Values { "Success", "Not Supported", "Unspecified Error",
        "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Invalid logical unit ID", "Invalid initiator port ID",
        "Invalid target port ID", "Invalid permission",
        "Target/initiator combination already exposed",
        "Requested logical unit number in use",
        "Maximum Map Count Exceeded", "Method Reserved",
        "Vendor Specific" }]
uint32 ExposePaths (
      [IN ( false ), OUT, Description (
          "Reference to the job if 'Method Parameters Checked - Job "
          "Started' is returned (MAY be null if job completed).")]
   CIM_ConcreteJob REF Job,
      [Required, IN, Description (
          "An array of IDs of logical unit instances. The LU "
          "instances MUST already exist. The members of this array "
          "MUST match the Name property of LogicalDevice instances "
          "that represent SCSI logical units. See the method "
          "description for conditions where this MAY be null."),
       ModelCorrespondence { "CIM_LogicalDevice.Name"}]
   string LUNames[],
      [IN, Description (
          "IDs of initiator ports. If existing StorageHardwareID "
          "instances exist, they MUST be used. Implicit creation "
     "of StorageHardwareID if no instance matches is "
          "NOT supported."),
       ModelCorrespondence { "CIM_StorageHardwareID.StorageID"}]
   string InitiatorPortIDs[],
      [IN, Description (
          "IDs of target ports. See the method description for "
          "conditions where this MAY be null."),
       ModelCorrespondence { "CIM_SCSIProtocolEndpoint.Name"}]
   string TargetPortIDs[],

      [IN, Description (
          "A list of logical unit numbers to assign to the "
          "corresponding logical unit in the LUNames parameter. "
          "(within the context of the elements specified in the "
          "other parameters). If the LUNames parameter is null, "
          "then this parameter MUST be null. Otherwise, if this "
          "parameter is null, all LU numbers are assigned by the "
          "hardware or instrumentation."),
       ModelCorrespondence {
          "CIM_ProtocolControllerForUnit.DeviceNumber"}]
```

```
    string DeviceNumbers[],

       [IN, Description (
           "A list of permissions to assign to the corresponding "
           "logical unit in the LUNames parameter. This specifies "
           "the permission to assign within the context of the "
           "elements specified in the other parameters. Setting this "
           "to 'No Access' assigns the DeviceNumber for the "
           "associated initiators, but does not grant read or write "
           "access. If the LUNames parameter is not null then this "
           "parameter MUST be specified."),
        ValueMap { "0", "2", "3", "4", "5..15999", "16000.." },
        Values { "Unknown", "Read Write", "Read-Only", "No Access",
           "DMTF Reserved", "Vendor Reserved" },
        ModelCorrespondence {
           "CIM_ProtocolControllerForUnit.DeviceAccess" }]
    uint16 DeviceAccesses[],

       [IN, OUT, Description (
           "An array of references to SCSIProtocolControllers "
           "(SPCs). On input, this can be null, or contain exactly "
           "one element; there MAY be multiple references on output. "
           "If null on input, the instrumentation will create one or "
           "more new SPC instances. If an SPC is specified, the "
           "instrumentation will attempt to add associations to one "
           "or more existing SPCs. If the first array element is a "
           "valid SPC reference and SCSI semantics can be preserved, "
           "the instrumentation MUST attach associations to the "
           "specified SPC. If multiple elements are non-null on "
           "input, the instrumentation MUST report an invalid "
           "parameter. On output, this is an array of references to "
           "SPCs created or modified as the result of processing the "
           "request.")]
    CIM_SCSIProtocolController REF ProtocolControllers[],

       [IN, Description (
           "Override existing lower priority VLUNs, if necessary. "
           "Can be used only when exporting to a specific host. "
      "This parameter is a vendor-specific extension for 3PAR. "
      "The default value is false.")]
    boolean Override,
       [IN, Description (
           "Do not issue a VLUN (Virtual Logical Unit Number) Change "
           "Notification (VCN) after export.  For direct connect or "
           "loop configuration, a VCN consists of a Fibre Channel "
           "Loop Initialization Primitive (LIP).  For fabric "
           "configuration a VCN consists of Registered State Change "
           "Notification (RSCN) being sent to the fabric controller. "
      "This parameter is a vendor-specific extension for 3PAR. "
      "The default value is false.")]
    boolean NoVCN,

       [OUT, Description (
           "An array of descriptive text of the result of the operation, "
      "with each entry containing the result of each "
      "expose path operation.")]
    string ResultDescriptions[]);

    [Override ("HidePaths"), Description (
        "3PAR version of the HidePaths operation, adding an extra "
    "property ResultDescription to better describe the result "
    "of the hide path operation, and DeviceNumbers for more "
    "efficient HidePaths operation.\n"
    "\n"
        "Hide a list of SCSI logical units (such as a RAID volume or "
        "tape drive) from a list of initiators and/or target ports "
```

```
        "on a SCSIProtocolController (SPC). \n"
        "\n"
        "The parameters for this method are: Job - null if no job "
        "created, otherwise this is a reference to the job. LUNames "
        "- the list of names of the logical units to use. "
        "InitiatorPortIDs - the names of the initiator ports to use. "
        "TargetPortIDs - the names of the target ports to use. "
        "DeviceNumbers (optional) - the device numbers (LUNs) to use. "
  "This is a 3PAR extension."
        "ProtocolControllers - SPCs involved in this operation. \n"
        "\n"
        "The LUNames and the optional DeviceNumbers parameters "
        "are mutually indexed arrays - any element in DeviceNumbers "
        "will set a property relative to the "
        "LogicalDevice instance named in the corresponding element "
        "of LUNames. DeviceNumbers MUST be null (the provider will "
  "deduce the DeviceNumbers of all the LU's in LUNames) or have "
  "the same number of elements as LUNames. If these conditions "
  "are not met, the instrumentation MUST return a 'Invalid Parameter' "
        "status or a CIM_Error. \n"
        "\n"
        "When hiding logical units, 3PAR InServ array supports only "
        "the Remove LUs from a view and Hide Full Paths from View. "
        "Remove LUs from a view requires that the "
        "LUNames parameter not be null and that the InitiatorIDs and "
        "TargetPortIDs parameters be null. \n"
        "\n"
  "The SPC is automatically deleted when the last logical unit "
  "is removed since SPCAllowsNoLUs is false."),
    ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
       "4097", "4098", "4099", "4100", "4101..32767",
    "32768..65535" },
    Values { "Success", "Not Supported", "Unspecified Error",
       "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
       "Method Parameters Checked - Job Started",
       "Invalid logical unit ID", "Invalid initiator port ID",
       "Invalid target port ID",
       "Target/initiator combination not exposed",
       "Method Reserved", "Vendor Specific" }]
uint32 HidePaths (
      [IN ( false ), OUT, Description (
          "Reference to the job if 'Method Parameters Checked - Job "
          "Started' is returned (MAY be null if job completed).")]
   CIM_ConcreteJob REF Job,

      [Required, IN, Description (
          "A list of IDs of logical units. Each LU instance MUST "
          "already exist. See the method description for conditions "
          "where this MAY be null."),
       ModelCorrespondence {
          "CIM_LogicalDevice.Name"}]
   string LUNames[],

      [IN, Description (
          "IDs of initiator ports. See the method description for "
          "conditions where this MAY be null."),
       ModelCorrespondence { "CIM_StorageHardwareID.StorageID" }]
   string InitiatorPortIDs[],

      [IN, Description (
          "IDs of target ports. See the method description for "
          "conditions where this MAY be null."),
       ModelCorrespondence { "CIM_SCSIProtocolEndpoint.Name" }]
   string TargetPortIDs[],

      [IN, Description (
```

```
        "A list of logical unit numbers that corresponds to "
        "logical unit in the LUNames parameter. "
        "(within the context of the elements specified in the "
        "other parameters). This parameter is a vendor-specific "
    "extension for 3PAR. Use of this parameter is optional, "
    "but its use will enhance efficiency of the HidePaths "
    "operation."),
      ModelCorrespondence {
        "CIM_ProtocolControllerForUnit.DeviceNumber"}]
string DeviceNumbers[],

    [IN, OUT, Description (
        "An array of references to SCSIProtocolControllers "
        "(SPCs). On input, this MUST contain exactly one element; "
        "there MAY be multiple references on output. The "
        "instrumentation will attempt to remove associations "
        "(LUNames, InitiatorPortIDs, or TargetPortIDs) from this "
        "SPC. Depending upon the specific implementation, the "
        "instrumentation MAY need to create new SPCs with a "
        "subset of the remaining associations. On output, this is "
        "an array of references to SPCs created or modified as "
        "the result of processing the request.")]
CIM_SCSIProtocolController REF ProtocolControllers[],

    [IN, Description (
        "Do not issue a VLUN (Virtual Logical Unit Number) Change "
        "Notification (VCN) after removal.  For direct connect or "
        "loop configuration, a VCN consists of a Fibre Channel "
        "Loop Initialization Primitive (LIP).  For fabric "
        "configuration a VCN consists of Registered State Change "
        "Notification (RSCN) being sent to the fabric controller. "
    "This parameter is a vendor-specific extension for 3PAR. "
    "The default value is false.")]
boolean NoVCN,
    [OUT, Description (
        "An array of descriptive text of the result of the operation, "
    "with each entry containing the result of the "
    "hide path operation.")]
string ResultDescriptions[]);

[Deprecated { "ExposePaths" },
  Override ("ExposeDefaultLUs"), Description (
     "3PAR version of the ExposeDefaultLUs operation, adding an extra "
     "property ResultDescription to better describe the result "
     "of the expose logical units operation. \n"
     "This method is deprecated in favor of ExposePaths; port-present "
     "vluns are created via ExposePaths by setting InitiatorPortIDs to "
     "NULL.\n"
     "Expose a list of SCSI logical units (such as RAID volumes "
     "or tape drives) through a 'default view' "
     "SCSIProtocolController (SPC) through a list of target "
     "ports. The 'default view' SPC exposes logical units to all "
     "initiators. This SPC is identified by an association to a "
     "StorageHardwareID with Name property set to the empty "
     "string. \n"
     "\n"
     "The parameters for this method are: Job - null if no job "
     "created, otherwise this is a reference to the job. LUNames "
     "- the list of names of the logical units to use. "
     "TargetPortIDs - the names of the target ports to use. "
     "DeviceNumbers - the device numbers (LUNs) to use. "
     "DeviceAccesses - permissions for the logical units. "
     "ProtocolControllers - SPCs involved in this operation. \n"
     "\n"
     "There are two modes of operation, create and modify. If a "
     "NULL value is passed in for the SPC, then the "
```

```
        "instrumentation will attempt to create a new default view. "
        "There could be multiple default view SPCs associated with "
        "each port. If an SPC is passed in, then the "
        "instrumentation adds the new paths to the existing SPC. \n"
        "\n"
        "For creating a default view SPC, all parameters, except "
    "ProtocolControllers, MUST be specified since SPCAllowsNoLUs, "
    "SPCAllowsNoTargets and SPCAllowsNoInitiators properties are "
    "all false, and ClientSelectableDeviceNumbers property is "
    "true, for a 3PAR InServ array. \n"
        "\n"
        "The LUNames, DeviceNumbers, and DeviceAccesses parameters "
        "are mutually indexed arrays - any element in DeviceNumbers "
        "or DeviceAccesses will set a property relative to the "
        "LogicalDevice instance named in the corresponding element "
        "of LUNames. LUNames, DeviceNumbers and DeviceAccesses MUST "
    "have the same number of elements. If these conditions are "
    "not met, the instrumentation MUST return a 'Invalid Parameter' "
        "status or a CIM_Error. \n"
        "\n"
        "For modifying an SPC, the 3PAR InServ array supports only "
        "the Add LUs to a view use case."
        "Add LUs to a view requires that the LUNames and DeviceNumbers "
    "parameters not be null and that the InitiatorIDs and TargetPortIDs "
        "parameters be null. \n"
        "\n"
        "The relevant rules of SCSI semantics are: \n"
        "- an SPC MAY NOT be exposed through a particular "
        "host/target port pair that is in use by another SPC. (In "
        "other words, an SPC and its associated logical units and "
        "ports together correspond to the logical unit inventory "
        "provided by SCSI REPORT LUNS and INQUIRY commands) \n"
        "- each LogicalDevice associated to an SPC MUST have a "
        "unique ProtocolControllerForUnit DeviceNumber (logical unit "
        "number) \n"
        "The instrumentation MUST report an error if the client "
        "request would violate one of these rules. \n"
        "\n"
        "If the instrumentation provides PrivilegeManagementService, "
        "the results of setting DeviceAccesses MUST be synchronized "
        "with PrivilegeManagementService as described in the "
        "ProtocolControllerForUnit DeviceAccess description."),
     ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
        "4097", "4098", "4099", "4100", "4101", "4102..32767",
        "32768..65535" },
     Values { "Success", "Not Supported", "Unspecified Error",
        "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Invalid logical unit ID", "Invalid target port ID",
        "Invalid permission", "Requested logical unit number in use",
        "Maximum Map Count Exceeded", "Method Reserved",
        "Vendor Specific" }]
  uint32 ExposeDefaultLUs (
       [IN ( false ), OUT, Description (
           "Reference to the job if 'Method Parameters Checked - Job "
           "Started' is returned (MAY be null if job completed).")]
    CIM_ConcreteJob REF Job,
       [Required, IN, Description (
           "An array of IDs of logical unit instances. The LU "
           "instances MUST already exist. The members of this array "
           "MUST match the Name property of LogicalDevice instances "
           "that represent SCSI logical units. See the method "
           "description for conditions where this MAY be null."),
        ModelCorrespondence { "CIM_LogicalDevice.Name"}]
    string LUNames[],
       [IN, Description (
```

```
        "IDs of target ports. See the method description for "
        "conditions where this MAY be null."),
     ModelCorrespondence { "CIM_SCSIProtocolEndpoint.Name" }]
string TargetPortIDs[],

    [IN, Description (
        "A list of logical unit numbers to assign to the "
        "corresponding logical unit in the LUNames parameter. "
        "(within the context of the elements specified in the "
        "other parameters). If the LUNames parameter is null, "
        "then this parameter MUST be null. Otherwise, if this "
        "parameter is null, all LU numbers are assigned by the "
        "hardware or instrumentation."),
     ModelCorrespondence {
        "CIM_ProtocolControllerForUnit.DeviceNumber"}]
string DeviceNumbers[],

    [IN, Description (
        "A list of permissions to assign to the corresponding "
        "logical unit in the LUNames parameter. This specifies "
        "the permission to assign within the context of the "
        "elements specified in the other parameters. Setting this "
        "to 'No Access' assigns the DeviceNumber for all "
        "initiators, but does not grant read or write access. If "
        "the LUNames parameter is not null then this parameter "
        "MUST be specified."),
     ValueMap { "0", "2", "3", "4", "5..15999", "16000.." },
     Values { "Unknown", "Read Write", "Read-Only", "No Access",
        "DMTF Reserved", "Vendor Reserved" },
     ModelCorrespondence {
        "CIM_ProtocolControllerForUnit.DeviceAccess" }]
uint16 DeviceAccesses[],

    [IN, OUT, Description (
        "An array of references to SCSIProtocolControllers "
        "(SPCs). On input, this can be null, or contain exactly "
        "one element; there MAY be multiple references on output. "
        "If null on input, the instrumentation will create one or "
        "more new SPC instances. If an SPC is specified, the "
        "instrumentation will attempt to add associations to one "
        "or more existing SPCs. If the first array element is a "
        "valid SPC reference and SCSI semantics can be preserved, "
        "the instrumentation MUST attach associations to the "
        "specified SPC. If multiple elements are non-null on "
        "input, the instrumentation MUST report an invalid "
        "parameter. On output, this is an array of references to "
        "SPCs created or modified as the result of processing the "
        "request.")]
CIM_SCSIProtocolController REF ProtocolControllers[],

    [IN, Description (
        "Override existing lower priority VLUNs, if necessary. "
        "Can be used only when exporting to a specific host. "
   "This parameter is a vendor-specific extension for 3PAR. "
   "The default value is false.")]
boolean Override,

    [IN, Description (
        "Do not issue a VLUN (Virtual Logical Unit Number) Change "
        "Notification (VCN) after export.  For direct connect or "
        "loop configuration, a VCN consists of a Fibre Channel "
        "Loop Initialization Primitive (LIP).  For fabric "
        "configuration a VCN consists of Registered State Change "
        "Notification (RSCN) being sent to the fabric controller. "
   "This parameter is a vendor-specific extension for 3PAR. "
   "The default value is false.")]
```

```
    boolean NoVCN,

       [OUT, Description (
           "An array of descriptive text of the result of the operation, "
      "with each entry containing the result of the "
      "expose Default LU operation.")]
    string ResultDescriptions[]);

    [Deprecated { "HidePaths" },
     Override ("HideDefaultLUs"), Description (
        "3PAR version of the HideDefaultLUs operation, adding an extra "
        "property ResultDescription to better describe the result "
        "of the hide default logical units operation\n"
        "\n"
        "This method is deprecated in favor of HidePaths."
        "\n"
        "Hide a list of SCSI logical units (such as RAID volumes or "
        "tape drives) through a list of target ports on a default "
        "view SCSIProtocolController (SPC). \n"
        "\n"
        "The parameters for this method are: Job - null if no job "
        "created, otherwise this is a reference to the job. LUNames "
        "- the list of names of the logical units to use. "
        "TargetPortIDs - the names of the target ports to use. "
        "DeviceNumbers (optional) - the device numbers (LUNs) to use. "
  "This is a 3PAR extension."
        "ProtocolControllers - SPCs involved in this operation. \n"
        "\n"
        "The LUNames and the optional DeviceNumbers parameters "
        "are mutually indexed arrays - any element in DeviceNumbers "
        "will set a property relative to the "
        "LogicalDevice instance named in the corresponding element "
        "of LUNames. DeviceNumbers MUST be null (the provider will "
  "deduce the DeviceNumbers of all the LU's in LUNames) or have "
  "the same number of elements as LUNames. If these conditions "
  "are not met, the instrumentation MUST return a 'Invalid Parameter' "
        "status or a CIM_Error. \n"
        "\n"
        "When hiding logical units, 3PAR InServ array supports only "
        "the Remove LUs from a default view use case. "
        "Remove LUs from a default view requires that the "
        "LUNames parameter not be null and that the TargetPortIDs "
  "parameters be null. \n"
        "\n"
  "The SPC is automatically deleted when the last logical unit "
  "is removed since SPCAllowsNoLUs is false."),
     ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
        "4097", "4098", "4099", "4100..32767", "32768..65535" },
     Values { "Success", "Not Supported", "Unspecified Error",
        "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Invalid logical unit ID", "Invalid target port ID",
        "Maximum Map Count Error", "Method Reserved",
        "Vendor Specific" }]
uint32 HideDefaultLUs (
      [IN ( false ), OUT, Description (
          "Reference to the job if 'Method Parameters Checked - Job "
          "Started' is returned (MAY be null if job completed).")]
   CIM_ConcreteJob REF Job,

       [Required, IN, Description (
          "A list of IDs of logical units. Each LU instance MUST "
          "already exist. See the method description for conditions "
          "where this MAY be null."),
        ModelCorrespondence {
          "CIM_LogicalDevice.Name"}]
```

```
    string LUNames[],

       [IN, Description (
           "IDs of target ports. See the method description for "
           "conditions where this MAY be null."),
        ModelCorrespondence { "CIM_SCSIProtocolEndpoint.Name" }]
    string TargetPortIDs[],

       [IN, Description (
           "A list of logical unit numbers that corresponds to "
           "logical unit in the LUNames parameter. "
           "(within the context of the elements specified in the "
           "other parameters). This parameter is a vendor-specific "
     "extension for 3PAR. Use of this parameter is optional, "
     "but its use will enhance efficiency of the HideDefaultLUs "
     "operation."),
        ModelCorrespondence {
           "CIM_ProtocolControllerForUnit.DeviceNumber" }]
    string DeviceNumbers[],

       [IN, OUT, Description (
           "An array of references to SCSIProtocolControllers "
           "(SPCs). On input, this MUST contain exactly one element; "
           "there MAY be multiple references on output. The "
           "instrumentation will attempt to remove associations "
           "(LUNames or TargetPortIDs) from this SPC. Depending upon "
           "the specific implementation, the instrumentation MAY "
           "need to create new SPCs with a subset of the remaining "
           "associations. On output, this is an array of references "
           "to SPCs created or modified as the result of processing "
           "the request.")]
    CIM_SCSIProtocolController REF ProtocolControllers[],

       [IN, Description (
           "Do not issue a VLUN (Virtual Logical Unit Number) Change "
           "Notification (VCN) after removal.  For direct connect or "
           "loop configuration, a VCN consists of a Fibre Channel "
           "Loop Initialization Primitive (LIP).  For fabric "
           "configuration a VCN consists of Registered State Change "
           "Notification (RSCN) being sent to the fabric controller. "
     "This parameter is a vendor-specific extension for 3PAR. "
     "The default value is false.")]
    boolean NoVCN,

       [OUT, Description (
           "An array of descriptive text of the result of the operation, "
     "with each entry containing the result of the "
     "hide default LU operation.")]
    string ResultDescriptions[]);

     [Description (
        "Expose a list of SCSI logical units (such as RAID volumes "
        "or tape drives) to a list of StorageHardwareIDCollections "
        "(host names), through a list of target ports if specified. "
        "If target port ids are not specified, the logical units "
"will be visible to all InitiatorPortIDs that are member "
"of the StorageHardwareIDCollections (host see vluns). If "
"target ports are specified, the logical units will be visible "
"to InitiatorPortIDs that are member of the "
"StorageHardwareIDCollections and that are connected to/zoned with "
"the target ports (matched set vluns).\n"
        "\n"
        "The parameters for this method are: Job - null if no job "
        "created, otherwise this is a reference to the job. LUNames "
        "- the list of names of the logical units to use. "
        "HostNames - the names of the StorageHardwareIDCollection to use. "
```

```
        "DeviceNumbers - the device numbers (LUNs) to use. "
        "DeviceAccesses - permissions for the logical units. "
        "ProtocolControllers - SPCs involved in this operation. \n"
        "\n"
        "There are two modes of operation, create and modify. If a "
        "NULL value is passed in for the SPC, then the "
        "instrumentation will attempt to create a new view. "
    "If an SPC is passed in, then the instrumentation adds the "
    "new paths to the existing SPC. \n"
        "\n"
        "For creating a SPC, all parameters, except "
    "ProtocolControllers, MUST be specified since SPCAllowsNoLUs, "
    "SPCAllowsNoTargets and SPCAllowsNoInitiators properties are "
    "all false, and ClientSelectableDeviceNumbers property is "
    "true, for a 3PAR InServ array. \n"
        "\n"
        "The LUNames, DeviceNumbers, and DeviceAccesses parameters "
        "are mutually indexed arrays - any element in DeviceNumbers "
        "or DeviceAccesses will set a property relative to the "
        "LogicalDevice instance named in the corresponding element "
        "of LUNames. LUNames, DeviceNumbers and DeviceAccesses MUST "
    "have the same number of elements. If these conditions are "
    "not met, the instrumentation MUST return a 'Invalid Parameter' "
        "status or a CIM_Error. \n"
        "\n"
        "For modifying an SPC, the 3PAR InServ array supports only "
        "the Add LUs to a view use case."
        "Add LUs to a view requires that the LUNames and DeviceNumbers "
    "parameters not be null and that the Hostnames "
        "parameter be null. \n"
        "\n"
        "The relevant rules of SCSI semantics are: \n"
        "- an SPC MAY NOT be exposed through a particular "
        "host/target port pair that is in use by another SPC. (In "
        "other words, an SPC and its associated logical units and "
        "ports together correspond to the logical unit inventory "
        "provided by SCSI REPORT LUNS and INQUIRY commands) \n"
        "- each LogicalDevice associated to an SPC MUST have a "
        "unique ProtocolControllerForUnit DeviceNumber (logical unit "
        "number) \n"
        "The instrumentation MUST report an error if the client "
        "request would violate one of these rules. \n"
        "\n"
        "The relevant rules of SCSI semantics are: \n"
        "- an SPC MAY NOT be exposed through a particular "
        "host/target port pair that is in use by another SPC. (In "
        "other words, an SPC and its associated logical units and "
        "ports together correspond to the logical unit inventory "
        "provided by SCSI REPORT LUNS and INQUIRY commands) \n"
        "- each LogicalDevice associated to an SPC MUST have a "
        "unique ProtocolControllerForUnit DeviceNumber (logical unit "
        "number) \n"
        "The instrumentation MUST report an error if the client "
        "request would violate one of these rules. \n"
        "\n"
        "If the instrumentation provides PrivilegeManagementService, "
        "the results of setting DeviceAccesses MUST be synchronized "
        "with PrivilegeManagementService as described in the "
        "ProtocolControllerForUnit DeviceAccess description."),
      ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
        "4097", "4098", "4099", "4100", "4101", "4102", "4103..32767",
        "32768..65535" },
      Values { "Success", "Not Supported", "Unspecified Error",
        "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
        "Method Parameters Checked - Job Started",
        "Invalid logical unit ID", "Invalid host name",
```

```
  "Invalid target port ID",
         "Invalid permission", "Requested logical unit number in use",
         "Maximum Map Count Exceeded", "Method Reserved",
         "Vendor Specific" }]
  uint32 ExposeLUsToStorageHardwareIDCollection (
        [IN ( false ), OUT, Description (
            "Reference to the job if 'Method Parameters Checked - Job "
            "Started' is returned (MAY be null if job completed).")]
     CIM_ConcreteJob REF Job,
        [Required, IN, Description (
            "An array of IDs of logical unit instances. The LU "
            "instances MUST already exist. The members of this array "
            "MUST match the Name property of LogicalDevice instances "
            "that represent SCSI logical units. See the method "
            "description for conditions where this MAY be null."),
         ModelCorrespondence { "CIM_LogicalDevice.Name"}]
     string LUNames[],

        [Required, IN, Description (
            "An array of names for the hosts (StorageHardwareIDCOllection). "
            "Instances MUST already exist. The members of this array "
            "MUST match the ElementName property of TPD_StorageHardwareIDCollection
"
       "instances."),
         ModelCorrespondence { "TPD_StorageHardwareIDCollection.ElementName" }]
     string HostNames[],

        [Required, IN, Description (
            "An array of target port LPIDs, in '<node>:<slot>:<port>' format."
     "If this is empty, then host-sees vlun will be created; otherwise "
     "matched set vlun(s) will be created. ")]
     string TargetPortLPIDs[],

        [IN, Description (
            "A list of logical unit numbers to assign to the "
            "corresponding logical unit in the LUNames parameter. "
            "(within the context of the elements specified in the "
            "other parameters). If the LUNames parameter is null, "
            "then this parameter MUST be null. Otherwise, if this "
            "parameter is null, all LU numbers are assigned by the "
            "hardware or instrumentation."),
         ModelCorrespondence {
            "CIM_ProtocolControllerForUnit.DeviceNumber"}]
     string DeviceNumbers[],

        [IN, Description (
            "A list of permissions to assign to the corresponding "
            "logical unit in the LUNames parameter. This specifies "
            "the permission to assign within the context of the "
            "elements specified in the other parameters. Setting this "
            "to 'No Access' assigns the DeviceNumber for all "
            "initiators, but does not grant read or write access. If "
            "the LUNames parameter is not null then this parameter "
            "MUST be specified."),
         ValueMap { "0", "2", "3", "4", "5..15999", "16000.." },
         Values { "Unknown", "Read Write", "Read-Only", "No Access",
            "DMTF Reserved", "Vendor Reserved" },
         ModelCorrespondence {
            "CIM_ProtocolControllerForUnit.DeviceAccess" }]
     uint16 DeviceAccesses[],

        [Required, IN, OUT, Description (
            "An array of references to SCSIProtocolControllers "
            "(SPCs). On input, this can be null, or contain exactly "
            "one element; there MAY be multiple references on output. "
            "If null on input, the instrumentation will create one or "
```

```
        "more new SPC instances. If an SPC is specified, the "
        "instrumentation will attempt to add associations to one "
        "or more existing SPCs. If the first array element is a "
        "valid SPC reference and SCSI semantics can be preserved, "
        "the instrumentation MUST attach associations to the "
        "specified SPC. If multiple elements are non-null on "
        "input, the instrumentation MUST report an invalid "
        "parameter. On output, this is an array of references to "
        "SPCs created or modified as the result of processing the "
        "request.")]
    CIM_SCSIProtocolController REF ProtocolControllers[],

       [IN, Description (
           "Override existing lower priority VLUNs, if necessary. "
           "Can be used only when exporting to a specific host. "
      "This parameter is a vendor-specific extension for 3PAR. "
      "The default value is false.")]
    boolean Override,

       [IN, Description (
           "Do not issue a VLUN (Virtual Logical Unit Number) Change "
           "Notification (VCN) after export.  For direct connect or "
           "loop configuration, a VCN consists of a Fibre Channel "
           "Loop Initialization Primitive (LIP).  For fabric "
           "configuration a VCN consists of Registered State Change "
           "Notification (RSCN) being sent to the fabric controller. "
      "This parameter is a vendor-specific extension for 3PAR. "
      "The default value is false.")]
    boolean NoVCN,

       [OUT, Description (
           "An array of descriptive text of the result of the operation, "
      "with each entry containing the result of the "
      "expose LUs to StorageHardwareIDCollection operation.")]
    string ResultDescriptions[]);

    [Description (
        "Hide a list of SCSI logical units (such as RAID volumes or "
        "tape drives) through a list of StorageHardwareIDCollections. \n"
        "\n"
        "The parameters for this method are: Job - null if no job "
        "created, otherwise this is a reference to the job. LUNames "
        "- the list of names of the logical units to use. "
        "DeviceNumbers (optional) - the device numbers (LUNs) to use. "
        "ProtocolControllers - SPCs involved in this operation. \n"
        "\n"
        "The LUNames and the optional DeviceNumbers parameters "
        "are mutually indexed arrays - any element in DeviceNumbers "
        "will set a property relative to the "
        "LogicalDevice instance named in the corresponding element "
        "of LUNames. DeviceNumbers MUST be null (the provider will "
   "hide all LU's in LUNames on the SPC) or have the same number "
        "of elements as LUNames. If these conditions are not met, "
        "the instrumentation MUST return a 'Invalid Parameter' "
        "status or a CIM_Error. \n"
        "\n"
        "When hiding logical units, 3PAR InServ array supports only "
        "the Remove LUs from a view use case. "
        "Remove LUs from a view requires that the "
        "LUNames parameter not be null.\n"
        "\n"
   "The SPC is automatically deleted when the last logical unit "
   "is removed since SPCAllowsNoLUs is false."),
      ValueMap { "0", "1", "2", "3", "4", "5", "6..4095", "4096",
        "4097", "4098", "4099", "4100..32767", "32768..65535" },
      Values { "Success", "Not Supported", "Unspecified Error",
```

```
          "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
          "Method Parameters Checked - Job Started",
          "Invalid logical unit ID", "Invalid host name",
          "Maximum Map Count Error", "Method Reserved",
          "Vendor Specific" }]
   uint32 HideLUsFromStorageHardwareIDCollection (
         [IN ( false ), OUT, Description (
             "Reference to the job if 'Method Parameters Checked - Job "
             "Started' is returned (MAY be null if job completed).")]
      CIM_ConcreteJob REF Job,
         [Required, IN, Description (
             "A list of IDs of logical units. Each LU instance MUST "
             "already exist. See the method description for conditions "
             "where this MAY be null."),
          ModelCorrespondence {
             "CIM_LogicalDevice.Name"}]
      string LUNames[],

         [IN, Description (
             "A list of logical unit numbers that corresponds to "
             "logical unit in the LUNames parameter. "
             "(within the context of the elements specified in the "
             "other parameters). This parameter is a vendor-specific "
        "extension for 3PAR. Use of this parameter is optional, "
        "but its use will enhance efficiency of the HideDefaultLUs "
        "operation."),
          ModelCorrespondence {
             "CIM_ProtocolControllerForUnit.DeviceNumber"}]
      string DeviceNumbers[],

         [IN, OUT, Description (
             "An array of references to SCSIProtocolControllers "
             "(SPCs). On input, this MUST contain exactly one element; "
             "there MAY be multiple references on output. The "
             "instrumentation will attempt to remove associations "
             "(LUNames or TargetPortIDs) from this SPC. Depending upon "
             "the specific implementation, the instrumentation MAY "
             "need to create new SPCs with a subset of the remaining "
             "associations. On output, this is an array of references "
             "to SPCs created or modified as the result of processing "
             "the request.")]
      CIM_SCSIProtocolController REF ProtocolControllers[],

         [IN, Description (
             "Do not issue a VLUN (Virtual Logical Unit Number) Change "
             "Notification (VCN) after removal.  For direct connect or "
             "loop configuration, a VCN consists of a Fibre Channel "
             "Loop Initialization Primitive (LIP).  For fabric "
             "configuration a VCN consists of Registered State Change "
             "Notification (RSCN) being sent to the fabric controller. "
        "This parameter is a vendor-specific extension for 3PAR. "
        "The default value is false.")]
      boolean NoVCN,

         [OUT, Description (
             "An array of descriptive text of the result of the operation, "
        "with each entry containing the result of the "
        "hide default LU operation.")]
      string ResultDescriptions[]);
};

// ===================================================================
// ProtocolControllerMaskingCapabilities
// ===================================================================
[Description
     ("A subclass of Capabilities that defines the Masking-related "
```

```
      "capabilities of the 3PAR InServ storage system.")]
class TPD_ProtocolControllerMaskingCapabilities :
                    CIM_ProtocolControllerMaskingCapabilities
{
   [Description (
       "If true, StorageHardwareIDs need to be set up before calling "
       "ExposePaths or ExposeLUsToStorageHardwareIDCollection. "
       "StorageHardwareID is setup by calling CreateStorageHardwareID "
       "and specifying the ElementName parameter, which is mandatory "
       "for 3PAR CIMOM.")]
      boolean RequiresHostNodePortRegistration;
};

// ===================================================================
// SystemControllerMaskingCapabilities
// ===================================================================
[Association,
    Description (
        "TPD_SystemControllerMaskingCapabilities is an association between "
        "StorageSystem and ProtocolControllerMaskingCapabilities.") ]
class TPD_SystemControllerMaskingCapabilities : CIM_ElementCapabilities
{
    [Override ( "ManagedElement" ), Description (
          "The StorageSystem.")]
    TPD_StorageSystem REF ManagedElement;

    [Override ( "Capabilities" ), Description (
          "The protocol controller masking capabilities.")]
    TPD_ProtocolControllerMaskingCapabilities REF Capabilities;
};

// ===================================================================
// ConfigServicesForSCSIController Association
// ===================================================================
[Association,
    Description (
        "TPD_ConfigServicesForSCSIController is an association between "
        "TPD_SCSIController and ControllerConfigurationService.") ]
class TPD_ConfigServicesForSCSIController : CIM_ConcreteDependency
{
    [Override ( "Antecedent" ), Description (
          "The controller configuration service which the protocol "
          "controller depends on for configuration.")]
    TPD_ControllerConfigurationService REF Antecedent;

    [Override ( "Dependent" ), Description (
          "The protocol controller which uses the controller "
          "configuration service.")]
    TPD_SCSIController REF Dependent;
};

// ===================================================================
// 3PAR Storage System and Storage Volume association
// ===================================================================
[Association, Aggregation, Composition,
    Description (
        "3PAR InServ Storage System and Storage Volume Association ") ]
class TPD_SystemVolume : CIM_SystemDevice
{
    [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
        Description ("The storage system.") ]
    TPD_StorageSystem REF GroupComponent;

    [Override ("PartComponent"), Weak, Description (
        "The storage volume on storage system.") ]
    TPD_StorageVolume REF PartComponent;
```

```
};

// ==================================================================
// 3PAR SoftwareIdentity
// ==================================================================
[Description (
    "3PAR SoftwareIdentity describes Inform OS version loaded "
    "on InServ")]
class TPD_SoftwareIdentity : CIM_SoftwareIdentity
{
};


// ==================================================================
// 3PAR InstalledSoftwareIdentity
// ==================================================================
[Association,
    Description (
        "3PAR InstalledSoftwareIdentity is an association between "
        "the system and Inform OS identity.")]
class TPD_InstalledSoftwareIdentity : CIM_InstalledSoftwareIdentity
{
    [Override ("System"), Description (
          "The system on which the Inform software is "
     "installed. ") ]
    TPD_StorageSystem REF System;

    [Override ("InstalledSoftware"), Description (
          "The SoftwareIdentity, e.g. Inform version and "
     "release number, that is installed.") ]
    TPD_SoftwareIdentity REF InstalledSoftware;
};


// ==================================================================
// 3PAR's system package describes InServ's system packaging attributes
// including model, serial number and OS version
// ==================================================================
[Description (
    "3PAR system package describes InServ's system packaging attributes "
    " including model, serial number and OS version. ")]
class TPD_SystemPackage : CIM_PhysicalPackage
{
};


// ==================================================================
// 3PAR Storage System and System Package association
// ==================================================================
[Association,
    Description (
        "3PAR InServ Storage System and System Package Association.") ]
class TPD_ComputerSystemPackage : CIM_ComputerSystemPackage
{
    [Override ("Antecedent"),
        Description ("The InServ's physical package.") ]
    TPD_SystemPackage REF Antecedent;

    [Override ("Dependent"), Description (
        "The InServ storage system.") ]
    TPD_StorageSystem REF Dependent;
};

// ==================================================================
// 3PAR StorageCapabilities
// ==================================================================
```

```
[Description (
  "3PAR StorageCapabilities.  This class defines the capabilities "
  "of a StoragePool.  An instance of StorageCapabilities could be "
  "associated with a StoragePool by using ElementCapabilities.")]
class TPD_StorageCapabilities: CIM_StorageCapabilities
{
   [Description ("True indicates space limits on allocation from "
   "DynamicStoragePools may be enforced. This is TRUE for capabilities"
   " of DynamicStoragePool and concrete pool.")]
   boolean SpaceLimitSupported;

      [Description (
       "This is the maximum grow increment.  When a StorageVolume or "
       "a DynamicStoragePool grows on demand, the system may allocates upto "
       "this number of bytes"),
       Units ( "Bytes" )]
    uint64 AllocationUnitMax;

      [Description (
       "This is the minimum grow increment.  When a StorageVolume or "
       "a DynamicStoragePool grows on demand, the system may allocates at least "
       "this number of bytes"),
       Units ( "Bytes" )]
    uint64 AllocationUnitMin;

      [Description (
       "This is the default grow increment.  When a StorageVolume or "
       "a DynamicStoragePool grows on demand, the system may allocates the "
       "default number of bytes"),
       Units ( "Bytes" )]
    uint64 AllocationUnitDefault;

      [Description (
       "Warning threshold for generating an indication for "
       " RemainingManagedSpace.  Value of zero means no warning generated.  "
       "Triggered when "
       " RemainingManagedSpace <=
(TotalManagedSpace*LowSpaceLimitWarningThreshold)/100."
       " MinValue = 0, MaxValue = 100 "
       "This is 0 for both concrete pool and DynamicStoragePool. This property"
       " is not populated for all other kinds of pools"),
       Units ( "Percent" )]
    uint16 LowSpaceWarningThresholdDefault;

      [Description (
       "Warning threshold for generating an indication for "
       "SpaceConsumed.  Value of zero means no warning generated.  "
       "Triggered when "
       " SpaceConsumed >= (SpaceLimit*SpaceLimitWarningThreshold)/100."
       " MinValue = 0, MaxValue = 100 "
       "This is 0 for both concrete pool and DynamicStoragePool. This property"
       " is not populated for all other kinds of pools"),
       Units ( "Percent" )]
    uint16 SpaceLimitWarningThresholdDefault;

};


// ===================================================================
// 3PAR CapabilitiesOfStoragePool
// ===================================================================
[Association,
    Description (
        "TPD_CapabilitiesOfStoragePool is an association between "
        "StoragePool and StorageCapabilities."
        "This association is used to figure out the service quality (RAID types)"
```

```
        " that a StoragePool can support.") ]
class TPD_CapabilitiesOfStoragePool: CIM_ElementCapabilities
{
    [Override ( "ManagedElement" ), Description (
          "TPD_StoragePool or TPD_DeltaReplicaStoragePool.")]
    CIM_StoragePool REF ManagedElement;

    [Override ( "Capabilities" ), Description (
          "The capabilities (RAID types) that the StoragePool can support.")]
    TPD_StorageCapabilities REF Capabilities;
};


// ===================================================================
// 3PAR StorageSettingsAssociatedToCapabilities
// ===================================================================
   [Association, Description (
       "This association define StorageSettings that reflect the "
       "capabilities of the associated StorageCapabilities. The "
       "associated StorageSetting MAY NOT define the operational "
       "characteristics (through settings properties) of any storage "
       "element. Certain StorageSetting instances can be defined as "
       "\"Fixed = Not Changeable\" by using the \"ChangeableType\" "
       "attribute. \"Fixed\" settings have this special association. "
       "This association SHALL be defined between \"Fixed - Not "
       "Changeable\" instances of StorageSetting with the "
       "StorageCapabilities instances that are associated with the "
       "StoragePools which support the storage characteristics "
       "described by the StorageSetting instance. \n"
       "Fixed settings MAY be associated to many StorageCapabilities.")]
class TPD_StorageSettingsAssociatedToCapabilities :
                          CIM_StorageSettingsAssociatedToCapabilities
{
      [Override ( "Antecedent" ), Description (
          "3PAR StorageCapabilities.")]
      TPD_StorageCapabilities REF Antecedent;

      [Override ( "Dependent" ), Description (
          "3PAR StorageSetting.")]
      TPD_StorageSetting REF Dependent;
};

// ===================================================================
// 3PAR StorageSettingsGeneratedFromCapabilities
// ===================================================================
   [Association, Description (
       "This association define StorageSettings that reflect the "
       "capabilities of the associated StorageCapabilities. The "
       "associated StorageSetting may not define the operational "
       "characteristics (through settings properties) of any storage "
       "element. StorageSettingsGeneratedFromCapabilities is the "
       "association between instances of StorageCapabilities and those "
       "instances of StorageSetting that have been created from the "
       "StorageCapabilities instance using the StorageCapabilities "
       "\"CreateSetting\" method. These settings have a "
       "StorageSetting.ChangeableType of \"Changeable - Transient\" or "
       "\"Changeable - Persistent\" The settings associated by this "
       "class reflect the capabilities from which they are generated. "
       "These setting SHALL be associated with one "
       "StorageCapabilities. \n"
       "A generated setting can be deleted by the implementation at "
       "any time if it is a a StorageSetting of \"Changed - "
       "Transient\" ChangeableType. \n"
       "A client should not use this association to find transient "
       "Settings to subsequently modify and/or use because that would "
       "increase the likelihood of setting contention across clients. "
```

```
       "Instead the implementation uses this association to define "
       "transient settings, which can be located through any means, "
       "that have special temporal based life cycle. \n"
       "DefaultSetting is meaningless in this class.")]
class TPD_StorageSettingsGeneratedFromCapabilities :
                      CIM_StorageSettingsGeneratedFromCapabilities
{
      [Override ( "Antecedent" ), Description (
          "3PAR StorageCapabilities.")]
      TPD_StorageCapabilities REF Antecedent;

      [Override ( "Dependent" ), Description (
          "3PAR StorageSetting.")]
      TPD_StorageSetting REF Dependent;
};

// ===================================================================
// 3PAR's product package describes InServ's product attributes
// including vendor name, product name, serial number and OS version
// ===================================================================
[Description (
    "3PAR product package describes InServ's product attributes"
    " including vendor name, product name, serial number and OS version. ")]
class TPD_Product : CIM_Product
{
};

// ===================================================================
// 3PAR Product and System Package association
// ===================================================================
[Association, Aggregation, Composition,
    Description (
        "3PAR InServ Product and System Package Association.") ]
class TPD_ProductPhysicalComponent : CIM_ProductPhysicalComponent
{
    [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
        Description ("The InServ product.") ]
    TPD_Product REF GroupComponent;

    [Override ("PartComponent"), Description (
        "The InServ's system packaging attributes.") ]
    TPD_SystemPackage REF PartComponent;

};


// ===================================================================
// StorageHardwareID
// ===================================================================
   [Description (
       "StorageHardwareID represents the host HBA.")]
class TPD_StorageHardwareID : CIM_StorageHardwareID
{
   [Description ( "IP Address for this HBA.") ]
   String IPAddress;
};

// ===================================================================
// AuthorizedPrivilege
// ===================================================================
   [Description (
       "AuthorizedPrivilege represents the access permission "
       "that a StorageHardwareID is allowed on a "
       "SCSIProtocolController.")]
class TPD_AuthorizedPrivilege : CIM_AuthorizedPrivilege {
};
```

```
// ===================================================================
// PrivilegeForStorageHardwareID
// ===================================================================
[Association,
    Description (
       "PrivilegeForStorageHardwareID associates StorageHardwareID "
       "to its access permission (AuthorizedPrivilege).")]
class TPD_PrivilegeForStorageHardwareID : CIM_AuthorizedSubject {
      [Override ( "Privilege" ), Description (
          "The AuthorizedPrivilege either granted or denied to an "
          "Identity, Role or Collection. Whether the privilege is "
          "granted or denied is defined by the inherited property, "
          "CIM_Privilege.PrivilegeGranted.")]
   TPD_AuthorizedPrivilege REF Privilege;

      [Override ( "PrivilegedElement" ), Description (
          "The host StorageHardwareID for which AuthorizedPrivileges are granted or
"
          "denied. Whether the privilege is granted or denied is "
          "defined by the property, CIM_Privilege.PrivilegeGranted.")]
   TPD_StorageHardwareID REF PrivilegedElement;
};

// ===================================================================
// PrivilegeForSCSIController
// ===================================================================
[Association, Description (
       "PrivilegeForSCSIController is an association used to tie the "
       "AuthorizedPrivileges to SCSIProtocolController resources.")]
class TPD_PrivilegeForSCSIController : CIM_AuthorizedTarget {

      [Override ( "Privilege" ), Description (
          "The AuthorizedPrivilege affecting the SCSIProtocolController "
     "resource.")]
   TPD_AuthorizedPrivilege REF Privilege;

      [Override ( "TargetElement" ), Description (
          "The SCSIProtocolController resources to which the "
          "AuthorizedPrivilege applies.")]
   TPD_SCSIController REF TargetElement;
};

// ===================================================================
// StorageHardwareIDCollection
// ===================================================================
   [Description (
       "A collection of StorageHardwareIDs (host WWN) within a host.")]
class TPD_StorageHardwareIDCollection : CIM_SystemSpecificCollection {
   [Description (
       "System assigned ID for this host collection.")]
   uint64 ID;

    [Description ("3PAR InServ host persona index number.") ]
    uint32 HostPersonaID;

   [Description (
       "Name of the administrative domain that this host collection belongs "
       "to.\n")]
    String Domain;
};

// ===================================================================
// PrivilegeForStorageHardwareIDCollection
// ===================================================================
[Association,
```

```
    Description (
       "PrivilegeForStorageHardwareIDCollection associates "
      "StorageHardwareIDCollection to its access permission (AuthorizedPrivilege).")]
class TPD_PrivilegeForStorageHardwareIDCollection : CIM_AuthorizedSubject {
      [Override ( "Privilege" ), Description (
          "The AuthorizedPrivilege either granted or denied to an "
          "Identity, Role or Collection. Whether the privilege is "
          "granted or denied is defined by the inherited property, "
          "CIM_Privilege.PrivilegeGranted.")]
   TPD_AuthorizedPrivilege REF Privilege;

      [Override ( "PrivilegedElement" ), Description (
          "The host for which AuthorizedPrivileges are granted or "
          "denied. Whether the privilege is granted or denied is "
          "defined by the property, CIM_Privilege.PrivilegeGranted.")]
   TPD_StorageHardwareIDCollection REF PrivilegedElement;
};

// ===================================================================
// MemberOfStorageHardwareIDCollection
// ===================================================================
   [Association, Aggregation, Description (
       "MemberOfStorageHardwareIDCollection is an aggregation used "
       "to establish membership of StorageHardIDs (host WWNs) in a "
       "host.")]
class TPD_MemberOfStorageHardwareIDCollection : CIM_MemberOfCollection {

      [Key, Aggregate, Description (
          "The MemberOfStorageHardwareIDCollection (host) that "
     "aggregates StorageHardwareID (host WWNs).")]
   TPD_StorageHardwareIDCollection REF Collection;

      [Key, Description (
          "The aggregated member of the "
     "MemberOfStorageHardwareIDCollection.")]
   TPD_StorageHardwareID REF Member;
};

// ===================================================================
// StorageHardwareIDManagementService
// ===================================================================
   [Description (
       "StorageHardwareIDManagementService provides methods for "
       "manipulating instances of StorageHardwareIDs and manipulating "
       "the trust of these IDs in the underlying storage system.")]
class TPD_StorageHardwareIDManagementService :
            CIM_StorageHardwareIDManagementService {

    [Override ("CreateHardwareIDCollection"), Description (
        "Create a group of StorageHardwareIDs as a new instance of "
        "SystemSpecificCollection. This is useful to define a set of "
        "authorized subjects that can access volumes in a disk "
        "array. This method allows the client to make a request of a "
        "specific Service instance to create the collection and "
        "provide the appropriate class name. When these capabilities "
        "are standardized in CIM/WBEM, this method can be deprecated "
        "and intrinsic methods used. In addition to creating the "
        "collection, this method causes the creation of the "
        "HostedCollection association (to this service's scoping "
        "system) and MemberOfCollection association to members of "
        "the IDs parameter."
        "For 3PAR, Setting is added as an optional parameter."),
      ValueMap { "0", "1", "2", "3", "4", "5", "6..0xFFF", "0x1000",
        "0x1001", "0x1002", "0x1003..0x7FFF", "0x8000.." },
      Values { "Success", "Not Supported", "Unspecified Error",
        "Timeout", "Failed", "Invalid Parameter", "DMTF Reserved",
```

```
        "Invalid HardwareID instance",
        "Implementation does not support hardware ID collections",
        "Input hardware IDs cannot be used in same collection",
        "Method Reserved", "Vendor Specific" }]
    uint32 CreateHardwareIDCollection (

        [IN, Description (
            "The ElementName to be assigned to the created "
            "collection.")]
        string ElementName,

        [IN, Description (
            "Array of strings containing representations of "
            "references to StorageHardwareID instances that will "
            "become members of the new collection.")]
        string HardwareIDs[],

        [IN, Description (
            "REF to the StorageClientSettingData containing the "
            "OSType appropriate for this initiator. If left NULL, the "
            "instrumentation assumes a standard OSType - i.e., that "
            "no OS-specific behavior for this initiator is defined.")]
        CIM_StorageClientSettingData REF Setting,

        [IN, Description (
            "Name of the administrative domain.")]
        String Domain,

        [IN ( false ), OUT, Description (
            "The new instance of SystemSpecificCollection that is "
            "created.")]
        CIM_SystemSpecificCollection REF Collection );
};
[Description (
        "Sets the initiator and/or target CHAP authentication information "
        "on the host as specified in Collection."),
      ValueMap { "0", "4", "5", "0x1000" },
      Values { "Success", "Failed", "Invalid Parameter",
        "Non-existent Collection" }]
    uint32 SetISCSICHAP(
        [IN, Description (
            "The initiator CHAP name. If this value is not "
            "specified, then the CHAP name defaults to the "
            "host name. Applicable only if InitiatorSecret is set.")]
        string InitiatorCHAPName,

        [IN, Description (
            "The target CHAP name. If this value is not "
            "specified, then the CHAP name defaults to the HP 3PAR "
            "System name. Applicable only if TargetSecret is set.")]
        string TargetCHAPName,

        [IN, Description (
            "Whether the initiator CHAP secret is treated as an ASCII "
            "string or as hex number. If this value is not specified then it "
            "defaults to ASCII. Applicable only if InitiatorSecret is set."),
          ValueMap { "0", "1" },
          Values { "ASCII", "Hexadecimal" }]
        Uint16 InitiatorSecretType,

        [IN, Description (
            "Whether the target CHAP secret is treated as an ASCII "
            "string or as hex number. If this value is not specified then it "
            "defaults to ASCII. Applicable only if TargetSecret is set."),
          ValueMap { "0", "1" },
          Values { "ASCII", "Hexadecimal" }]
```

```
        Uint16 TargetSecretType,

        [IN, Description (
            "The CHAP secret for the host. If hex is specified, it is "
            "treated as a 16-byte hex number. Otherwise it should be a "
            "printable ASCII string 12 to 16 characters in length with "
            "no spaces.")]
        string InitiatorSecret,

        [IN, Description (
            "The CHAP secret for the target. If hex is specified, it is "
            "treated as a 16-byte hex number. Otherwise it should be a "
            "printable ASCII string 12 to 16 characters in length with "
            "no spaces.")]
        string TargetSecret,

        [IN, Description (
            "Reference to the iSCSI StorageHardwareIDCollection.")]
        CIM_SystemSpecificCollection REF Collection );

    [Description (
        "Remove CHAP authentication on the host as specified in "
        "Collection."),
      ValueMap { "0", "4", "5", "0x1000" },
      Values { "Success", "Failed", "Invalid Parameter",
        "Non-existent Collection" }]
    uint32 RemoveISCSICHAP(

        [IN, Description (
            "Remove target CHAP only or both target and init CHAP. "
            "If this value is not specified then both target and "
            "init CHAP will be deleted."),
          ValueMap { "1", "2" },
          Values { "Target", "Both Target and Initiator" }]
        Uint16 RemovalScope,

        [IN, Description (
            "Reference to the iSCSI StorageHardwareIDCollection.")]
        CIM_SystemSpecificCollection REF Collection );
};


// ===================================================================
// HostedStorageHWIDManagementService
// ===================================================================
   [Association, Description (
       "TPD_HostedStorageHWIDManagementService is an association "
       "between TPD_StorageSystem and "
       "TPD_StorageHardwareIDManagementService. The cardinality of "
       "this association is 1-to-1. A System may host only one "
       "StorageHardwareIDManagementService. ")]
class TPD_HostedStorageHWIDManagementService : CIM_HostedService {
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ StorageSystem.")]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ), Weak, Description (
          "The StorageHardwareID management service hosted on the "
     "3PAR InServ.")]
   TPD_StorageHardwareIDManagementService REF Dependent;
};

// ===================================================================
// HostedStorageHardwareIDCollection
// ===================================================================
   [Association, Description (
```

```
       "HostedStorageHardwareIDCollection defines a "
       "TPD_StorageHardwareIDCollection in a context of a scoping "
       "TPD_StorageSystem.")]
class TPD_HostedStorageHardwareIDCollection : CIM_HostedCollection {

      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
          "The 3PAR StorageSystem.")]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The collection of StorageHardwareIDs in the context of "
     "the StorageSystem.")]
   TPD_StorageHardwareIDCollection REF Dependent;
};

// ===================================================================
// MgmtServicesForStorageHWIDCollection Association
// ===================================================================
[Association,
    Description (
        "TPD_MgmtServicesForStorageHWIDCollection is an association between "
        "TPD_StorageHardwareID or TPD_StorageHardwareIDCollection and "
        "TPD_StorageHardwareIDManagementService.") ]
class TPD_MgmtServicesForStorageHWIDCollection : CIM_ConcreteDependency
{
    [Override ( "Antecedent" ), Description (
          "The storage hardware ID management service which the "
          "StorageHardwareIDCollection depends on for configuration.")]
    TPD_StorageHardwareIDManagementService REF Antecedent;
};


// ===================================================================
//  PrivilegeManagementService
// ===================================================================
   [Description (
       "3PAR PrivilegeManagementService is responsible for creating, "
       "deleting, and associating AuthorizedPrivilege instances. "
       "References to 'subject' and 'target' define the entities that "
       "are associated with an AuthorizedPrivilege instance via the "
       "relationships, AuthorizedSubject and AuthorizedTarget, "
       "respectively. When created, an AuthorizedPrivilege instance is "
       "related to this (PrivilegeManagement)Service via the "
       "association, ConcreteDependency.  Note: 3PAR does not support"
       "changing or removing AuthorizedPrivilege.")]
class TPD_PrivilegeManagementService : CIM_PrivilegeManagementService
{
};

// ===================================================================
// 3PAR StorageSystem and PrivilegeManagementService association.
// ===================================================================
[Association,
    Description (
       "TPD_HostedPrivilegeManagementService is an association "
       "between TPD_StorageSystem and TPD_PrivilegeManagementService. "
       "The cardinality of this association is 1-to-1. A System may "
       "host only one PrivilegeManagementService. ")]
class TPD_HostedPrivilegeManagementService : CIM_HostedService {
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ StorageSystem.")]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The PrivilegeManagementService hosted on the 3PAR InServ.")]
   TPD_PrivilegeManagementService REF Dependent;
```

```
};


// ===================================================================
// ManagementServicesForAuthorizedPrivilege association
// ===================================================================
[Association,
    Description (
        "TPD_ManagementServiceForAuthorizedPrivilege is an association "
        "between TPD_PrivilegeManagementService and "
        "TPD_AuthorizedPrivilege.") ]
class TPD_ManagementServiceForAuthorizedPrivilege : CIM_ConcreteDependency
{
    [Override ( "Antecedent" ), Description (
          "The privilege management service which supports "
          "authorized privilege.")]
    TPD_PrivilegeManagementService REF Antecedent;

    [Override ( "Dependent" ), Description (
          "The authorized privilege which uses the management service.")]
    TPD_AuthorizedPrivilege REF Dependent;
};


// ===================================================================
// Indication
// ===================================================================
[Indication,
    Description (
        "Indication specifying that the operational status of an FC "
   "port has changed.")
]
class TPD_FCPortInstModification : CIM_InstModification
{
};


// ===================================================================
// ThinProvisioning indications
// ===================================================================

[Indication,
    Description (
        "Indication specifying that the state of an alert condition "
        "has changed. ID of the alert which state has changed is in "
        "CorrelatedIndications.")
]
class TPD_AlertStateChangeAlert : CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a storage pool  "
   "has been over-used.")
]
class TPD_StoragePoolOverUsedWarningAlert : CIM_AlertIndication
{
};


[Indication,
    Description (
        "Indication specifying that a storage pool has reached its "
 "allocation warning threshold.")
]
```

```
class TPD_StoragePoolGrowWarningAlert : CIM_AlertIndication
{
};


[Indication,
    Description (
        "Indication specifying that a storage pool has reached its "
 "allocation limit.")
]
class TPD_StoragePoolGrowLimitAlert : CIM_AlertIndication
{
};


[Indication,
    Description (
        "Indication specifying that a storage pool fails to grow.")
]
class TPD_StoragePoolGrowFailureAlert : CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a storage volume has failed to "
 "allocate more space.")
]
class TPD_StorageVolumeAllocationFailureAlert : CIM_AlertIndication
{
};


[Indication,
    Description (
        "Indication specifying that a storage volume has reached its "
 "allocation warning threshold..")
]
class TPD_StorageVolumeAllocationWarningAlert : CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a storage volume has reached its "
 "its allocation limit threshold.")
]
class TPD_StorageVolumeAllocationLimitAlert : CIM_AlertIndication
{
};


[Indication,
    Description (
        "Indication specifying that a delta replica storage pool "
 "has reached its allocation warning threshold.")
]
class TPD_DeltaReplicaStoragePoolGrowWarningAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a delta replica storage pool "
 "has reached its allocation limit threshold.")
]
```

```
class TPD_DeltaReplicaStoragePoolGrowLimitAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a delta replica storage pool "
 "has failed to grow.")
]
class TPD_DeltaReplicaStoragePoolGrowFailureAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication indicating that the previous alert condition "
 "regarding a dynamic storage pool has cleared.")
]
class TPD_StoragePoolCapacityClearAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication indicating that the previous alert condition "
 "regarding a storage volume has cleared.")
]
class TPD_StorageVolumeCapacityClearAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication indicating that the previous alert condition "
 "regarding a delta replica storage pool has cleared.")
]
class TPD_DeltaReplicaStoragePoolCapacityClearAlert: CIM_AlertIndication
{
};

[Indication,
    Description (
        "Indication specifying that a storage volume has has failed to "
 "allocate snapshot space for user data.")
]
class TPD_SnapshotUserSpaceAlert : CIM_AlertIndication
{
};




// ===================================================================
// 3PAR System and ControllerConfigurationService association.
// ===================================================================
[Association,
    Description (
       "TPD_SystemHostedCCS is an association between TPD_StorageSystem "
       "and TPD_ControllerConfigurationService. The cardinality of "
       "this association is 1-to-1. A System may host only one "
       "ControllerConfigurationService. ")]
class TPD_SystemHostedCCS : CIM_HostedService {
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ StorageSystem.")]
   TPD_StorageSystem REF Antecedent;
```

```
      [Override ( "Dependent" ), Weak, Description (
          "The ControllerConfigurationService hosted on the 3PAR InServ.")]
   TPD_ControllerConfigurationService REF Dependent;
};

// ===================================================================
//  3PAR SFP
// ===================================================================
[Description (
   "3PAR SFP for Node FC ports.")]
class TPD_SFP : CIM_PhysicalConnector {

   [Description ("Revision code.") ]
   String Revision;

   [Description (
       "The current bandwidth of the SFP in Bits per Second."),
    Units ( "Bits per Second" )]
   uint64 Speed;

   [Description ("Position of the TX disable pin.  If pin position is HI, "
      "TX laser is disabled"),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "LO", "HI"} ]
   uint16 DisableTXPinPosition;

   [Description ("Position of the RX loss pin.  If pin position is HI, "
      "RX loss of signal"),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "LO", "HI"} ]
   uint16 LossRXPinPosition;

   [Description ("Position of the TX fault pin.  If pin position is HI, "
      "TX fault"),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "LO", "HI"} ]
   uint16 FaultTXPinPosition;

   [Description ("If TRUE, SFP is not qualified.") ]
   boolean IsUnqualified;

   [Description ("If TRUE, RX Power is Low.") ]
   boolean IsRXPowerLow;

   [Description ("If TRUE, DDM information is available.") ]
   boolean IsDDMAvailable;
};

// ===================================================================
//  3PAR Port SFP
// ===================================================================
[Association, Description (
    "A relationship between FCPort and it's SFP.")]
class TPD_FCPortSFP : CIM_PortActiveConnection {

      [Override ( "Antecedent" ), Description (
          "The PhysicalConnector.")]
   TPD_SFP REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The NetworkPort that transmits using the Connector.")]
   TPD_FCPort REF Dependent;
};

// ===================================================================
// 3PAR StorageClientSettingData
```

```
// ===================================================================
[Description (
    "StorageClientSettingData contains host OS types.")]
class TPD_StorageClientSettingData : CIM_StorageClientSettingData {
    [Description ("3PAR InServ host persona index number.") ]
    uint32 HostPersonaID;

    [Description ("3PAR InServ host persona overall capabilities. "
        "Possible value is a combination of following values: "
        "0x00000000, 0x00000001, 0x00000002, "
        "0x00000008, 0x00000010, "
        "0x00000020, 0x00000040"),
        BitMap {"0","1","2","4","5","6","7"},
       BitValues  {"Unknown", "Unit Attention Report LUNs", "Volume Set Addressing",

        "Soft Inquiry Data", "Normal Auto Contingent Allegiance",
        "Report Target Port Groups", "Enable SES device"}]
    uint32 HostPersonaCapabilities;
};

// ===================================================================
// 3PAR SystemClientSettingData
// ===================================================================
[Association,
    Description (
        "TPD_SystemClientSettingData is an association between "
        "ComputerSystem and StorageClientSettingData.") ]
class TPD_SystemClientSettingData : CIM_ElementSettingData
{
    [Override ("ManagedElement"), Description (
          "The CmputerSystem.")]
    TPD_StorageSystem REF ManagedElement;

    [Override ("SettingData"), Description (
          "The StorageClientSettingData.")]
    TPD_StorageClientSettingData REF SettingData;
};

// ===================================================================
// 3PAR HardwareIDElementSettingData
// ===================================================================
[Association,
    Description (
        "TPD_HardwareIDElementSettingData is an association between "
        "StorageHardwareID and StorageClientSettingData.") ]
class TPD_HardwareIDElementSettingData: CIM_ElementSettingData
{
    [Override ("ManagedElement"), Description (
          "The StorageHardwareID.")]
    TPD_StorageHardwareID REF ManagedElement;

    [Override ("SettingData"), Description (
          "The StorageClientSettingData.")]
    TPD_StorageClientSettingData REF SettingData;
};

// ===================================================================
// 3PAR HWIDCollectionElementSettingData
// ===================================================================
[Association,
    Description (
        "TPD_HWIDCollectionElementSettingData is an association between "
        "StorageHardwareIDCollection and StorageClientSettingData.") ]
class TPD_HWIDCollectionElementSettingData : CIM_ElementSettingData
{
    [Override ("ManagedElement"), Description (
```

```
          "The StorageHardwareIDCollection.")]
    TPD_StorageHardwareIDCollection REF ManagedElement;

    [Override ("SettingData"), Description (
          "The StorageClientSettingData.")]
    TPD_StorageClientSettingData REF SettingData;
};
// ===================================================================
// StorageDomainGroup
// ===================================================================
[Description ("A collection of domains in groups.")]
class TPD_StorageDomainGroup : CIM_SystemSpecificCollection {
    [Description (
        "System assigned ID for this domain group.")]
    uint32 ID;

    [Description (
        "An array of domains in the group."),
    ArrayType ( "Indexed" )]
    string GroupMembers[];
};

// ===================================================================
// HostedStorageDomainGroup
// ===================================================================
[Association, Description (
       "HostedStorageDomainGroup defines a "

       "TPD_StorageDomainGroup in a context of a scoping "
       "TPD_StorageSystem.")]
class TPD_HostedStorageDomainGroup : CIM_HostedCollection {

    [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
        "The 3PAR StorageSystem.")]
    TPD_StorageSystem REF Antecedent;

    [Override ( "Dependent" ), Description (
        "The group of domain in the context of the StorageSystem.")]
    TPD_StorageDomainGroup REF Dependent;
};

#pragma include ("3PAR_TPDCage.mof")
#pragma include ("3PAR_TPDDisk.mof")
#pragma include ("3PAR_TPDNode.mof")
#pragma include ("3PAR_TPDEnv.mof")
#pragma include ("3PAR_TPDLocation.mof")
#pragma include ("3PAR_TPDEthPort.mof")
#pragma include ("3PAR_TPDiSCSI.mof")
#pragma include ("3PAR_TPDStats.mof")
#pragma include ("3PAR_TPDJob.mof")
#pragma include ("3PAR_TPDReplicationSvcs.mof")
```

3PAR_TPDCage.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
```

```
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDCage.mof
//
//  Purpose : This MOF contains 3PAR Cage classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// Base TPD Drive Cage
// ===================================================================
[Description (
   "Base 3PAR Drive Cage")]
class TPD_DriveCage : CIM_Chassis
{
   [Description ("Type of the cage."), Read,
      ValueMap {"0", "2", "3", "4", "5"},
      Values {"Unknown", "DC1", "DC2", "DC3", "DC4"} ]
   uint16 CageType;

   [Description ("Subtype of the cage."), Read,
      ValueMap {"0", "1", "2"},
      Values {"FC", "Nearline", "ALPHAFC"} ]
   uint16 CageSubType;

   [Description ("Position of the cage in the node."), Read ]
   uint16 Position;

   [Description ("Location of the cage.") ]
   string Location;

   [Description ("Shows if the loop is split."), Read ]
   boolean LoopSplit;

   [Description ("Chain position of the A Loop."), Read ]
   uint16 ChainPositionLoopA;

   [Description ("Chain position of the B Loop."), Read ]
   uint16 ChainPositionLoopB;

   [Description ("Shows if the locate is ON."), Read ]
   boolean LocateON;
};

// ===================================================================
// DC1 TPD Drive Cage
// ===================================================================
[Description (
   "DC1 3PAR Drive Cage")]
class TPD_DriveCageDC1Family : TPD_DriveCage
{
   [Description ("Is Left side of the cage."), Read ]
   boolean IsLeftSide;

   [Description ("Is Right side of the cage."), Read ]
   boolean IsRightSide;
};

// ===================================================================
// DC3 TPD Drive Cage
// ===================================================================
```

```
[Description (
   "DC3 3PAR Drive Cage")]
class TPD_DriveCageDC3Family : TPD_DriveCage
{
   [Description ("Status of the temperature sensor."), Read,
      ValueMap {"0", "1", "2", "3", "4", "5", "6", "7"},
      Values {"Unknown", "OK", "Over Threshold", "Over/Under Threshold",
         "Communication Error", "Not Present", "IM Sync failed",
         "Not Available"} ]
   uint16 TempSensorState;

   [Description ("Value of the temperature sensor."), Read ]
   uint16 TempSensorValue;

   [Description ("Status of the temperature sensor threshold."), Read,
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "OK", "Under Warning Threshold",
              "Under Failure Threshold", "Over Warning Threshold",
              "Over Failure Threshold"} ]
   uint16 TempSensorThreshold;

   [Description ("Status of the operators panel."), Read,
      ValueMap {"0", "1", "2", "3", "4", "5", "6"},
      Values {"Unknown", "OK", "Failed", "Warning",
         "Unrecoverable", "Not Present", "Not Available"} ]
   uint16 OperatorPanelState;

   [Description ("Status of the audible alarm."), Read,
      ValueMap {"0", "1", "2", "3"},
      Values {"Unknown", "OK", "Muted", "Unrecoverable"} ]
   uint16 AudibleAlarmState;

   [Description ("Enclosure ID."), Read ]
   uint16 EnclosureID;
};

// ===================================================================
// DC2 or DC4 TPD Drive Cage
// ===================================================================
[Description (
   "DC2 or DC4 3PAR Drive Cage")]
class TPD_DriveCageDC2Family : TPD_DriveCage
{
   [Description ("Firmware version on the midplane CPU."), Read ]
   string FirmwareVersion;

   [Description ("Status of the firmware on the midplane CPU."), Read,
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "Not Current", "Current"} ]
   uint16 FirmwareStatus;

   [Description ("State of midplane CPU."), Read,
      ValueMap {"0", "1", "2", "3", "4", "5", "6"},
      Values {"No Response", "Fatal Error", "OK",
      "Hotplug Ready", "Loop Down Recovery", "Updating Firmware",
      "Copying Firmware"} ]
   uint16 CPUState;
};

// ===================================================================
// Base TPD Cage Interface Card
// ===================================================================
[Description (
   "Base 3PAR Cage Interface Card ")]
class TPD_CageInterfaceCard : CIM_Card
{
```

```
   [Description ("Position of the Interface Card in the cage.") ]
   uint16 Position;

   [Description ("Type of the cage this Interface Card is in."),
      ValueMap {"0", "2", "3", "4", "5"},
      Values {"Unknown", "DC1", "DC2", "DC3", "DC4"} ]
   uint16 CageType;

   [Description ("WWN of the Loop A ports in Interface Card.  Array index 0"
      " represents Loop A0, array index 1 represents"
      " Loop A1, etc.") ]
   uint64 LoopAPortWWNs[];

   [Description ("WWN of the Loop B ports in Interface Card.  Array index 0"
      " represents Loop B0, array index 1 represents"
      " Loop B1, etc.") ]
   uint64 LoopBPortWWNs[];

   [Description ("WWN of the Node FC Port connected to A Loop(s).  Array index 0"
      " represents Loop A0, array index 1 represents"
      " Loop A1, etc.") ]
   uint64 NodePortsLoopA[];

   [Description ("WWN of the Node FC Port connected to B Loops.  Array index 0"
      " represents Loop B0, array index 1 represents"
      " Loop B1, etc.") ]
   uint64 NodePortsLoopB[];

   [Description ("Loop A link speed. Array index 0"
      " represents Loop A0, array index 1 represents"
      " Loop A1, etc."),
           ValueMap {"1", "2", "4", "10"},
          Values {"1G", "2G", "4G", "10G"} ]
   uint16 LoopALinkSpeed[];

   [Description ("Loop B link speed. Array index 0"
      " represents Loop B0, array index 1 represents"
      " Loop B1, etc."),
           ValueMap {"1", "2", "4", "10"},
          Values {"1G", "2G", "4G", "10G"} ]
   uint16 LoopBLinkSpeed[];

    [Description ("Location of the initiator ports connecting to the cage ports"
       " which are on loop A. Array index 0 represents port A0, index 1"
       " represents A1. If the value is string 0, it represents no connection.")]
    string PortsLoopA[];

    [Description ("Location of the initiator ports connecting to the cage ports"
       " which are on loop B. Array index 0 represents port B0, index 1"
       " represents B1. If the value is string 0, it represents no connection.")]
    string PortsLoopB[];
};

// ===================================================================
// TPD Cage Interface Card for DC1 cage
// ===================================================================
[Description (
   "3PAR Cage Interface Card for DC1 cage")]
class TPD_CageInterfaceCardDC1Family : TPD_CageInterfaceCard
{
   [Description ("Split LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
     Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SplitLED;
```

```
   [Description ("System LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SystemLED;

   [Description ("Hot-plug LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 HotplugLED;

   [Description ("State of Interface Card CPU."),
      ValueMap {"0", "1", "2", "3", "4", "5", "6"},
      Values {"Partner Not Responding", "Fatal Error", "OK",
         "Hotplug Ready", "Loop Down Recovery", "Updating Firmware",
         "Copying Firmware"} ]
   uint16 CPUSelfState;

   [Description ("State of partner Interface Card CPU."),
      ValueMap {"0", "1", "2", "3", "4", "5", "6"},
      Values {"Partner Not Responding", "Fatal Error", "OK",
         "Hotplug Ready", "Loop Down Recovery", "Updating Firmware",
         "Copying Firmware"} ]
   uint16 CPUPartnerState;

   [Description ("Status of the firmware on the Interface Card processor."),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "Not Current", "Current"} ]
   uint16 FirmwareStatus;

   [Description ("Firmware version on the Interface Card processor.") ]
   string FirmwareVersion;

   [Description ("Loop A link LED status."
      " Array index 0 is Loop A0, array index 1 is Loop A1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopALinkLED[];

   [Description ("Loop B link LED status."
      " Array index 0 is Loop B0, array index 1 is Loop B1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopBLinkLED[];

   [Description ("Shows if the locate is ON."
      " Array index 0 is Loop A0, array index 1 is Loop A1.") ]
   boolean LoopALocateON[];

   [Description ("Shows if the locate is ON."
      " Array index 0 is Loop B0, array index 1 is Loop B1.") ]
   boolean LoopBLocateON[];
};

// ===================================================================
// TPD Cage Interface Card for DC3 cage
// ===================================================================
[Description (
   "3PAR Cage Interface Card for DC3 cage.  DC3 cage has 2 FC-ALs, with one "
   "having four A loops, and the other four B loops."
   "Check LoopPosition to see if Cage Interface instance for A or B.")]
class TPD_CageInterfaceCardDC3Family : TPD_CageInterfaceCard
{
```

```
   [Description ("If master set to TRUE.") ]
   boolean IsMaster;

   [Description ("Loop A link speed. Array index 0"
      " represents Loop A0, array index 1 represents"
      " Loop A1, etc."),
            ValueMap {"0x0400", "0x0411", "0x0414",
                "0x0415", "0x0417", "0x0418",
                "0x0421", "0x0422", "0x0423", "0x0424",
                "0x0425", "0x0426", "0x0427", "0x0428",
                "0x0431", "0x0432",
                "0x0433", "0x0434",
                "0x0435", "0x0436",
                "0x0437", "0x0438",
                "0x0439", "0x043B",
                "0x043C", "0x043E",
                "0x043F", "0x0440",
                "0x0441", "0x0442",
                "0x044F", "0x0450",
                "0x0451", "0x04FF"},
           Values {"OK", "Warning,High LIP", "Warning,Burst Word Errors",
                "Warning,High Word Errors", "Warning,High CRC Errors", "Warning,High
 Clock Delta",
                "Bypass", "No SFP Present", "Bypass,Tx Fault", "Bypass,LIP",
                "Bypass,Data Timeout", "Bypass,RX Loss", "Bypass,Sync Loss",
"Bypass,PTFI",
                "Manual Bypass, System Manager", "Manual Bypass,Redundant Port
Connection",
                "Manual Bypass,Stall Threshold", "Manual Bypass,Burst Word Error
Threshold",
                "Manual Bypass,High Word Error Threshold", "Manual Bypass,Burst CRC
Error Threshold",
                "Manual Bypass,High CRC Error Threshold", "Manual Bypass,High Clock
Delta Threshold",
                "Manual Bypass,Mirror Configuration", "Manual Bypass,Host Management
 Rules",
                "Manual Bypass,Trunked Cabling Errors", "Manual Bypass,TBD",
                "Manual Bypass,Looped Back", "Manual Bypass,Insert Oscillation",
              "Manual Bypass,Burst LIP Threshold", "Manual Bypass,High LIP Threshold",

                "Manual Bypass, Diagnostic Transmit", "Manual Bypass,Drive Bypass",
                "Manual Bypass,Drive Fault", "Unknown"} ]
   uint16 LoopState[];

   [Description ("Status of the firmware on the Interface Card processor."),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "Not Current", "Current"} ]
        uint16 FirmwareStatus;

   [Description ("Firmware version on the Interface Card processor.") ]
        string FirmwareVersion;

   [Description ("State of Interface Card IFC."),
      ValueMap {"0x0401", "0x0402", "0x0403", "0x0404", "0x0405", "0x0406",
                "0x0407"},
     Values {"OK", "Failed", "Warning", "Unrecoverable", "Not Installed", "Unknown",

             "Not Available"} ]
   uint16 IFCState;

   [Description ("State of Interface Card ESH."),
      ValueMap {"0", "1", "2", "3", "4", "5", "6"},
      Values {"Unknown", "OK", "Failed", "Warning",
         "Unrecoverable", "Not Installed", "Not Available"} ]
   uint16 ESHState;
```

```
   [Description ("Extended State of Interface Card ESH.  Use this when ESHState"
            "is set to 2"),
      ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8"},
      Values {"Unknown", "OK", "Microcontroller Failed POST",
         "Microcontroller FATAL Error", "ASIC Not Functioning",
         "ASIC Failed POST", "ASIC Port Failed POST",
         "ASIC Clock Delta Beyond Threshold", "Unavailable"} ]
   uint16 ExtendedESHState;
};

// ===================================================================
// TPD Cage Interface Card for DC2 or DC4 cage
// ===================================================================
[Description (
   "3PAR Cage Interface Card for DC2 or DC4 cage")]
class TPD_CageInterfaceCardDC2Family : TPD_CageInterfaceCard
{
   [Description ("Split LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SplitLED;

   [Description ("System LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SystemLED;

   [Description ("Hot-plug LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 HotplugLED;

   [Description ("Loop A sfp link RX status."
      " Array index 0 is Loop A0, array index 1 is Loop A1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopALinkRX[];

   [Description ("Loop B sfp link RX status."
      " Array index 0 is Loop A0, array index 1 is Loop A1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopBLinkRX[];

   [Description ("Loop A sfp link TX status."
      " Array index 0 is Loop B0, array index 1 is Loop B1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopALinkTX[];

   [Description ("Loop B sfp link TX status."
      " Array index 0 is Loop B0, array index 1 is Loop B1."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 LoopBLinkTX[];

   [Description ("Shows if the locate is ON."
      " Array index 0 is Loop A0, array index 1 is Loop A1.") ]
   boolean LoopALocateON[];
```

```
   [Description ("Shows if the locate is ON."
      " Array index 0 is Loop B0, array index 1 is Loop B1.") ]
   boolean LoopBLocateON[];
};

// ===================================================================
// 3PAR Drive Cage and Cage Interface Card association
// ===================================================================
   [Association, Aggregation,
      Description (
       "3PAR Drive Cage and Interface Card association")]
class TPD_InterfaceCardInDriveCage : CIM_PackageInChassis {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The Drive Cage that contains FC-AL cards")]
   TPD_DriveCage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The FC-AL card that is contained in the Drive Cage.")]
   TPD_CageInterfaceCard REF PartComponent;
};

// ===================================================================
// Base 3PAR Magazine
// ===================================================================
[Description (
  "Base 3PAR Disk Magazine ")]
class TPD_Magazine : CIM_Magazine {

   [Description ("Position of the Magazine in the cage.") ]
   uint16 Position;

   [Description ("Type of the cage this magazine is in."),
      ValueMap {"0", "2", "3", "4", "5"},
      Values {"Unknown", "DC1", "DC2", "DC3", "DC4"} ]
   uint16 CageType;

   [Description ("State of Magazine on Loop A."),
      ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8"},
      Values {"Unknown", "Not Present", "OK", "Loop Failed",
         "I2C Transaction Failed", "Power Supply Failed",
         "Manual Bypass, System Manager", "Manual Bypass, Port",
         "Midplane I2C Transaction Failed"} ]
   uint16 StateOnLoopA;

   [Description ("State of Magazine on Loop B."),
      ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8"},
      Values {"Unknown", "Not Present", "OK", "Loop Failed",
         "I2C Transaction Failed", "Power Supply Failed",
         "Manual Bypass, System Manager", "Manual Bypass, Port",
         "Midplane I2C Transaction Failed"} ]
   uint16 StateOnLoopB;

   [Description ("Shows if the locate is ON."), Read ]
   boolean LocateON;

   [Description ("System LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
     Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SystemLED;

   [Description ("Hot-plug LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
```

```
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 HotplugLED;
};

// ===================================================================
// 3PAR Drive Cage and Magazine association
// ===================================================================
   [Association, Aggregation,
      Description (
       "3PAR Drive Cage and Magazine association")]
class TPD_MagazineInDriveCage : CIM_PackageInChassis {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The Drive Cage that contains Magazines")]
   TPD_DriveCage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The Magazine that is contained in the Drive Cage.")]
   TPD_Magazine REF PartComponent;
};

// ===================================================================
// TPD_SystemPackage and TPD_DriveCage association
// ===================================================================
[Association, Aggregation, Description (
       "The SysCageContainer association represents the relationship between "
       "a TPD_SystemPackage and TPD_DriveCage.")]
class TPD_SysCageContainer : CIM_Container {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The SystemPackage that contains DriveCage(s)")]
   TPD_SystemPackage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The TPD_DriveCage which is contained in the SystemPackage.")]
   TPD_DriveCage REF PartComponent;
};

// ===================================================================
// TPD Cage Sensor
// ===================================================================
[Description (
   "TPD Cage Sensor information")]
class TPD_CageSensor : CIM_Sensor
{
   [Description ("Position of the Sensor in the cage.") ]
   uint16 Position;

   [Description ("Sensor base unit."), Read ]
   string BaseUnit;

   [Description ("Current sensor reading (actual * 100)."), Read ]
   uint16 CurrentReading;

   [Description ("Low limit (actual * 100)."), Read ]
   uint16 LowLimit;

   [Description ("High limit (actual * 100)."), Read ]
   uint16 HighLimit;

   [Description ("Sensor state."),
           ValueMap {"0", "1", "2"},
           Values {"OK", "Under Limit", "Over Limit"} ]
```

```
   uint16 State;
};

// ===================================================================
//  TPD DriveCage Sensor
// ===================================================================
[Association, Description (
   "Many Devices include Sensors or have Sensors installed nearby, "
   "in order to measure critical input and output properties. This "
   "association indicates that relationship.")]
class TPD_SensorInDriveCage : CIM_PackageDependency {

   [Override ( "Antecedent" ), Description (
      "The Sensor.")]
   TPD_CageSensor REF Antecedent;

   [Override ( "Dependent" ), Description (
      "The LogicalDevice for which information is measured by the "
      "Sensor. It only applies for DC2 or DC4.")]
   TPD_DriveCage REF Dependent;
};
```

3PAR_TPDCopySvcs.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2005-2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDCopySvcs.mof
//
//  Purpose : This MOF contains 3PAR Copy Services classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 3/19/2007
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// 3PAR delta replica storage pool
// ===================================================================
[Description (
   "Storage Pool representing the SA/SD space of a volume that is "
   "used to create a virtual copy.")]
class TPD_DeltaReplicaStoragePool: SNIA_StoragePool
{
   [Description (
       "Type of disk drives that are in this pool."),
    ValueMap {"0", "1", "2", "3", "4"},
    Values {"Unknown", "FC", "Nearline", "Allocated from DynamicStoragePool", "SSD"}
 ]
    uint16 DiskDeviceType;

   [Description (
       "Describe the RAID type of the volume that is associated to this "
       "pool."),
```

```
    ValueMap {"0", "10", "50", "60"},
    Values {"RAID0", "RAID10", "RAID50", "RAID60"} ]
    uint16 RaidType;
};

// ===================================================================
// 3PAR ReplicaPoolForStorage
// ===================================================================
[Association,
 Description (
       "Association between TPD_DeltaReplicaStoragePool and "
       "the source TPD_StorageVolume.")]
class TPD_ReplicaPoolForStorage : CIM_ReplicaPoolForStorage {
      [Override ( "Antecedent" ), Min ( 0 ), Max ( 1 ), Description (
          "The source volume.")]
   TPD_StorageVolume REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The specialized replica pool.")]
   TPD_DeltaReplicaStoragePool REF Dependent;
};

// ===================================================================
// 3PAR StorageReplicationCapabilities
// ===================================================================
[Description (
       "This subclass defines the replication capabilities of a "
       "StorageConfigurationService. Multiple instances of "
       "StorageReplicationCapabilities may be associated with a "
       "StorageConfigurationService using ElementCapabilities. A "
       "provider should create one instance for each supported "
       "SynchronizationType.")]
class TPD_StorageReplicationCapabilities : CIM_StorageReplicationCapabilities
{
      [Description (
          "Maximum number of replicas that can be associated "
          "with one read-only source volume. This applies only "
   "to SupportedSynchronizationType of UnsyncAssoc-Delta "
   "since only snapshot volume can be read-only.")]
   uint16 MaximumReplicasPerReadOnlySource;

      [Description (
          "Maximum number of replicas that can be associated "
          "with one read-write source volume. This applies to "
   "both SupportedSynchronizationType of UnsyncAssoc-Delta "
   "and UnSyncAssoc-Full since InServ can have read-write "
   "snapshot or base volumes.")]
   uint16 MaximumReplicasPerReadWriteSource;
};

// ===================================================================
// 3PAR StorageSynchronized
// ===================================================================
[Association,
   Description (
       "Indicates that two Storage objects were replicated at the "
       "specified point in time.")]
class TPD_StorageSynchronized : CIM_StorageSynchronized {
      [Override ( "SystemElement" ), Description (
          "SystemElement represents the Volume that is the source of "
          "the replication.")]
   TPD_StorageVolume REF SystemElement;

      [Override ( "SyncedElement" ), Description (
          "SyncedElement represents the Volume that is the target of "
          "the replication.")]
```

```
   TPD_StorageVolume REF SyncedElement;
};
```

3PAR_TPDDisk.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDDisk.mof
//
//  Purpose : This MOF contains 3PAR Disk classes that will be loaded
//            into root/tpd namespace.  (Disk Drive Lite)
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////

[Description (
  "3PAR Disk Drive ")]
class TPD_DiskDrive : CIM_DiskDrive {

   [Description ("Position of the Drive in the Magazine.") ]
   uint16 Position;

   [Description ("Type of the cage this disk is in."),
        ValueMap {"0", "2", "3", "4", "5"},
        Values {"Unknown", "DC1", "DC2", "DC3", "DC4"} ]
    uint16 CageType;

   [Description ("System id number of the Drive.") ]
   uint32 ID;

   [Description ("Type of the Disk. Taken from SMI-S 1.5.0 draft."),
      ValueMap {"0", "1", "2", "3"},
           Values {"Unknown", "Other", "Hard Disk Drive", "Solid State Disk"} ]
   uint16 DiskType;

   [Description ("Device type of the Disk."),
      ValueMap {"0", "1", "2", "4", "5"},
           Values {"Unknown", "FC", "NL", "SSD", "Volume"} ]
   uint16 DeviceType;

   [Description ("System LED status."),
      ValueMap {"0", "1", "2", "3", "4", "5"},
      Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"}
 ]
   uint16 SystemLED;

   [Description ("Temperature of the Drive in C.") ]
   uint16 Temperature;

   [Description ("If Drive can be used to allocate LD.") ]
   boolean IsLDAllocateable;
```

```
[Description ("State of Disk on Loop A."),
   ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
   "12", "13", "14", "15", "16", "0x0411", "0x0414", "0x0415", "0x0417",
   "0x0418", "0x0421", "0x0423", "0x0424", "0x0425", "0x0426", "0x0427",
   "0x0428", "0x0432", "0x0433", "0x0434", "0x0435", "0x0436", "0x0437",
   "0x0438", "0x0439", "0x043B", "0x043C", "0x043E", "0x043F", "0x0440",
   "0x0441", "0x0442", "0x044F", "0x0450", "0x0451"},
   Values {"Unknown", "Not Present", "Offloop", "New Onloop", "Spinning Up",
   "OK", "Spin Up Failed", "Loop Failed", "Drive Error Bit On",
   "Bypass, System Manager", "Manual Bypass, Port", "Not Ready",
   "Spundown, System Manager", "Spundown, Port", "Spundown, Over Temperature",
   "Spinning Down Fail", "I2C Transaction Failed",
   "Warning,High LIP", "Warning,Word Error Burst", "Warning,Word Error High",
   "Warning,High CRC", "Warning,High Clock Delta", "Bypass",
   "Bypass,TX Fault", "Bypass,LIP", "Bypass,DTMOUT", "Bypass,RX Loss",
   "Bypass,Sync Loss", "Bypass,PTFI", "Manual Bypass,Redundant Port",
   "Manual Bypass,Stall Threshold", "Manual Bypass,Word Error Burst",
   "Manual Bypass,Word Error High", "Manual Bypass,Burst CRC",
   "Manual Bypass,High CRC", "Manual Bypass,High Clock Delta",
   "Manual Bypass,Mirror", "Manual Bypass,Host", "Manual Bypass,Trunked Cable",
   "Manual Bypass,TBD", "Loopback", "Manual Bypass,Ins_Osc",
   "Manual Bypass,Burst LIP", "Manual Bypass,High LIP",
   "Manual Bypass,Diag Transmit",
   "Manual Bypass,Drive Bypass", "Manual Bypass,Drive Fault"} ]
uint16 StateOnLoopA;

[Description ("State of Disk on Loop B."),
   ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
   "12", "13", "14", "15", "16", "0x0411", "0x0414", "0x0415", "0x0417",
   "0x0418", "0x0421", "0x0423", "0x0424", "0x0425", "0x0426", "0x0427",
   "0x0428", "0x0432", "0x0433", "0x0434", "0x0435", "0x0436", "0x0437",
   "0x0438", "0x0439", "0x043B", "0x043C", "0x043E", "0x043F", "0x0440",
   "0x0441", "0x0442", "0x044F", "0x0450", "0x0451"},
   Values {"Unknown", "Not Present", "Offloop", "New Onloop", "Spinning Up",
   "OK", "Spin Up Failed", "Loop Failed", "Drive Error Bit On",
   "Bypass, System Manager", "Manual Bypass, Port", "Not Ready",
   "Spundown, System Manager", "Spundown, Port", "Spundown, Over Temperature",
   "Spinning Down Fail", "I2C Transaction Failed",
   "Warning,High LIP", "Warning,Word Error Burst", "Warning,Word Error High",
   "Warning,High CRC", "Warning,High Clock Delta", "Bypass",
   "Bypass,TX Fault", "Bypass,LIP", "Bypass,DTMOUT", "Bypass,RX Loss",
   "Bypass,Sync Loss", "Bypass,PTFI", "Manual Bypass,Redundant Port",
   "Manual Bypass,Stall Threshold", "Manual Bypass,Word Error Burst",
   "Manual Bypass,Word Error High", "Manual Bypass,Burst CRC",
   "Manual Bypass,High CRC", "Manual Bypass,High Clock Delta",
   "Manual Bypass,Mirror", "Manual Bypass,Host", "Manual Bypass,Trunked Cable",
   "Manual Bypass,TBD", "Loopback", "Manual Bypass,Ins_Osc",
   "Manual Bypass,Burst LIP", "Manual Bypass,High LIP",
   "Manual Bypass,Diag Transmit",
   "Manual Bypass,Drive Bypass", "Manual Bypass,Drive Fault"} ]
uint16 StateOnLoopB;

[Description ("ALPA on Loop A.") ]
uint16 AlpaOnLoopA;

[Description ("ALPA on Loop B.") ]
uint16 AlpaOnLoopB;

[Description ("Size of the chunk.") ]
uint64 ChunkSize;

[Description ("Number of Normal Used OK chunks.") ]
uint32 ChunksNormalUsedOK;

[Description ("Number of Normal Used Fail chunks.") ]
uint32 ChunksNormalUsedFail;
```

```
[Description ("Number of Normal Unused Free chunks.") ]
uint32 ChunksNormalUnusedFree;

[Description ("Number of Normal Unused Uninitialized chunks.") ]
uint32 ChunksNormalUnusedUninit;

[Description ("Number of Normal Unused Fail chunks.") ]
uint32 ChunksNormalUnusedFail;

[Description ("Number of Spare Used OK chunks.") ]
uint32 ChunksSpareUsedOK;

[Description ("Number of Spare Used Fail chunks.") ]
uint32 ChunksSpareUsedFail;

[Description ("Number of Spare Unused Free chunks.") ]
uint32 ChunksSpareUnusedFree;

[Description ("Number of Spare Unused Uninitialized chunks.") ]
uint32 ChunksSpareUnusedUninit;

[Description ("Number of Spare Unused Fail chunks.") ]
uint32 ChunksSpareUnusedFail;

[Description ("Number of Correctable READ Errors.") ]
uint64 CorrectableReadErrors;

[Description ("Number of Uncorrectable READ Errors.") ]
uint64 UncorrectableReadErrors;

[Description ("Number of Correctable WRITE Errors.") ]
uint64 CorrectableWriteErrors;

[Description ("Number of Uncorrectable WRITE Errors.") ]
uint64 UncorrectableWriteErrors;

   [Description ("3PAR specific operational state for the disk"),
       ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11"},
    Values {"Unknown", "OK", "Error", "Not Ready",
   "Missing", "Named in table but invalid label", "New",
   "drive block size is not 512 bytes",
   "Invalid drive type, unknown vendor", "Unknown firmware",
   "Size reported does not allow the number of "
   "chunklets specified in scsi database", "No cage found"} ]
     uint16 OtherOperationalStatus;

 [Description ("Location of the initiator ports connecting to the ports of a"
     " cage where disks are attached. Array index 0 represents port A0, index 1"
     " represents A1. If the value is string 0, it represents no connection.")]
 string InitiatorPortA[];

 [Description ("Location of the initiator ports connecting to the ports of a"
     " cage where disks are attached. Array index 0 represents port B0, index 1"
     " represents B1. If the value is string 0, it represents no connection.")]
 string InitiatorPortB[];

 [Description ("Primary path of initiator ports connecting to A or B ports"
     " of the cage.")]
 string PrimaryPortPath;

[Description ("If Drive is used for ESI Channel.") ]
boolean IsESI;

[Description ("ESI State of Disk."),
   ValueMap {"0", "1", "2", "3"},
```

```
      Values {"Unknown", "Not Present", "OK", "Failed"} ]
   uint16 ESIState;

   [Description ("Disk RPM - revolutions per minute in unit of K. "
       "This property is null if SSD.") ]
   uint32 DiskSpeed;

};

// ===================================================================
// 3PAR DiskDrivePackage
// ===================================================================
[Description (
  "3PAR Disk Drive Package for Vendor, Model, and Serial Number")]
class TPD_DiskDrivePackage : CIM_PhysicalPackage {
};

// ===================================================================
// 3PAR DiskDrive and DiskDrivePackage mapping
// ===================================================================
[Association,
   Description (
   "3PAR DiskDrivePackage and DiskDrive mapping")]
class TPD_DiskPackageRealizes : CIM_Realizes {

         [Override ( "Antecedent" ), Description (
          "The physical component that implements the Device.")]
      TPD_DiskDrivePackage REF Antecedent;

         [Override ( "Dependent" ), Description (
          "The Disk Drive.")]
      TPD_DiskDrive REF Dependent;
};

// ===================================================================
// 3PAR Disk StorageExtent
// ===================================================================
[Description (
  "3PAR Disk Drive Storage Extent")]
class TPD_DiskStorageExtent : CIM_StorageExtent {
};

// ===================================================================
// 3PAR Disk StorageExtent
// ===================================================================
[Association,
   Description (
   "3PAR Disk Drive and Disk Storage Extent association")]
class TPD_DiskPresent : CIM_MediaPresent {
   [Override ( "Antecedent" ), Description (
     "The Disk Drive.")]
   TPD_DiskDrive REF Antecedent;

   [Override ( "Dependent" ), Description (
     "The storage extent accessed using the Disk Drive.")]
      TPD_DiskStorageExtent REF Dependent;
};

// ===================================================================
// 3PAR DiskDrive Software Identity
// ===================================================================
[Description (
  "3PAR Disk Drive Software Identity")]
class TPD_DiskSoftwareIdentity : CIM_SoftwareIdentity {
   [Description ("Status of the firmware on the disk drive."),
         ValueMap {"0", "1", "2"},
```

```
         Values {"Unknown", "Old", "Current"} ]
        uint16 FirmwareStatus;
};

// ===================================================================
// 3PAR DiskDrive and DiskSoftwareIdentity association
// ===================================================================
[Association,
   Description (
   "3PAR DiskSoftwareIdentity and DiskDrive mapping")]
class TPD_DiskSoftware : CIM_ElementSoftwareIdentity {

         [Override ( "Antecedent" ), Description (
          "The software that is installed on the Disk Drive.")]
      TPD_DiskSoftwareIdentity REF Antecedent;

         [Override ( "Dependent" ), Description (
          "The Disk Drive.")]
      TPD_DiskDrive REF Dependent;
};

// ===================================================================
// TPD_DiskDrivePackage and TPD_Magazine association
// ===================================================================
[Association, Aggregation, Description (
       "The MagazineContainer association represents the relationship between "
       "a TPD_Magazine and TPD_DiskDrivePackage.")]
class TPD_MagazineContainer : CIM_Container {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The Magazine that contains DiskDrivePackage(s)")]
   TPD_Magazine REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The DiskDrivePackage which is contained in the Magazine.")]
   TPD_DiskDrivePackage REF PartComponent;
};

// ===================================================================
// TPD_DiskDrive and TPD_StorageSystem association
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "The SystemDisk association represents a relationship between "
       "a TPD_DiskDrive and TPD_StorageSystem.")]
class TPD_SystemDisk : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent system in the Association.")]
   TPD_StorageSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The DiskDrive that is a component of a StorageSystem.")]
   TPD_DiskDrive REF PartComponent;
};

// ===================================================================
// StoragePoolComponent
// ===================================================================
   [Association, Aggregation, Description (
       "TPD_StoragePoolComponent is an association used to "
       "establish 'part of' relationships between StoragePool and "
       "Disk StorageExtent.")]
class TPD_StoragePoolComponent : CIM_ConcreteComponent {
```

```
      [Aggregate, Override ( "GroupComponent" ), Description (
          "The parent element in the association.")]
   CIM_StoragePool REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The child element in the association.")]
   TPD_DiskStorageExtent REF PartComponent;
};

// ===================================================================
// SCSIInitiatorTargetLogicalUnitPath
//    Comment this class out as it is still experimental
// ===================================================================

// [Association, Description (
//       "An association that models a host driver path to a SCSI "
//       "logical unit. Each permutation of initiator and target "
//       "ProtocolEndpoints and logical units is considered a separate "
//       "path. This class describes end-to-end path behavior such as "
//       "properties and operations commonly used in multipath "
//       "management.")]
//class TPD_SCSIInitTargetLUPath : CIM_SCSIInitiatorTargetLogicalUnitPath
//{
//      [Override ("Initiator"), Description (
//          "An initiator endpoint on a node that connects to the "
//     "target endpoint on a cage.")]
//   TPD_SCSIProtocolFCEndpoint REF Initiator;

//      [Override ("Target"), Description (
//          "The target endpoint residing on a cage that connects "
//     "to the initiator endpoint on a node.")]
//   TPD_SCSIProtocolFCALEndpoint REF Target;

//      [Override ("LogicalUnit"), Description (
//          "Storage extent of DiskDrive.")]
//   TPD_DiskStorageExtent REF LogicalUnit;
//};

// ===================================================================
// TPD_DiskStorageExtent and TPD_StorageSystem association
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "The SystemExtent association represents a relationship between "
       "a TPD_DiskStorageExtent and TPD_StorageSystem.")]
class TPD_SystemExtent : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent system in the Association.")]
   TPD_StorageSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The StorageExtent that is a component of a StorageSystem.")]
   TPD_DiskStorageExtent REF PartComponent;
};

// ===================================================================
// TPD_DiskStorageExtent and TPD_StorageVolume association
// ===================================================================
[Association, Description (
    "BasedOn is an association describing how StorageExtents can be "
    "assembled from lower level Extents.")]
class TPD_VolumeBasedOnDiskExtent : CIM_BasedOn {
```

```
      [Override ( "Antecedent" ), Description (
          "The lower level StorageExtent.")]
   TPD_DiskStorageExtent REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The higher level StorageExtent.")]
   TPD_StorageVolume REF Dependent;
};
```

3PAR_TPDNode.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDNode.mof
//
//  Purpose : This MOF contains 3PAR Node (MCS) classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////

// ====================================================================
// 3PAR NodeSystem
// ====================================================================
[Description (
   "3PAR Controller Node ")]
class TPD_NodeSystem : CIM_ComputerSystem
{
   [Description ("Position of the Node.") ]
   uint16 Position;

   [Description ("If this node is Master, set to TRUE.") ]
   boolean IsMaster;

   [Description ("If this node is Online, set to TRUE.") ]
   boolean IsOnline;

   [Description ("If this node is in Cluster, set to TRUE.") ]
   boolean IsInCluster;

   [Description ("Major part of kernel version.") ]
   uint16 KernelMajor;

   [Description ("Minor part of kernel version.") ]
   uint16 KernelMinor;

   [Description ("Revision part of kernel version.") ]
   uint16 KernelRevision;

   [Description ("Build part of kernel version.") ]
   uint16 KernelBuild;
```

```
   [Description ("Kernel Version in String format.") ]
   string KernelVersion;

   [Description ("Major part of BIOS version.") ]
   uint16 BiosMajor;

   [Description ("Minor part of BIOS version.") ]
   uint16 BiosMinor;

   [Description ("Revision part of BIOS version.") ]
   uint16 BiosRevision;

   [Description ("Build part of BIOS version.") ]
   uint16 BiosBuild;

   [Description ("BIOS Version in String format.") ]
   string BiosVersion;

   [Description ("System LED status."),
   ValueMap {"0", "1", "2", "3", "4", "5"},
   Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"} ]
   uint16 SystemLED;

   [Description ("Percentage of the node cache enabled.") ]
   uint16 CacheEnabled;

   [Description ("3PAR specific operational state for the disk"),
      ValueMap {"0", "1", "2", "3", "4", "5", "6", "7"},
      Values {"Unknown", "OK/Inited", "Error/Offline", "New",
      "Defunct, mismatched kernel revisions",
      "Defunct, node ide partition bad",
      "Defunct, node is rebooting",
      "Defunct, node is being shutdown"} ]
   uint16 OtherOperationalStatus;
};

// ===================================================================
// 3PAR NodePackage
// ===================================================================
[Description (
  "3PAR Controller Node Package for Vendor, Model, and Serial Number")]
class TPD_NodePackage : CIM_PhysicalPackage {
};

// ===================================================================
// 3PAR NodeSystem and NodePackage association
// ===================================================================
[Association,
    Description (
        "3PAR InServ NodeSystem and NodePackage Association.") ]
class TPD_NodeSystemPackage : CIM_ComputerSystemPackage
{
    [Override ("Antecedent"),
        Description ("The InServ's Node physical package.") ]
    TPD_NodePackage REF Antecedent;

    [Override ("Dependent"), Description (
        "The InServ Node system.") ]
    TPD_NodeSystem REF Dependent;
};

// ===================================================================
// 3PAR NodeRedundancySet
// ===================================================================
[Description (
  "3PAR Controller Node redundancy set.  It is associated with a max of "
```

```
  "2 NodeSystem objects")]
class TPD_NodeRedundancySet : CIM_RedundancySet {
};

// ==================================================================
// 3PAR NodePairSystem
// ==================================================================
[Description (
  "3PAR Controller Node Pair computer system. One for each of the "
  "NodeRedundancySet object.")]
class TPD_NodePairSystem : CIM_ComputerSystem {
};

// ==================================================================
// 3PAR NodePairRedundancySet
// ==================================================================
[Description (
  "3PAR Controller Node Pair redundancy set.  This object is associated "
  "to all the NodePairSystem objects.")]
class TPD_NodePairRedundancySet : CIM_RedundancySet {
};

// ==================================================================
// 3PAR NodeMemberOfSet
// ==================================================================
[Association, Aggregation, Description (
  "TPD_NodeMemberOfSet is an aggregation used to establish "
  "membership of TPD_NodeSystem in a TPD_NodeRedundancySet.")]
class TPD_NodeMemberOfSet : CIM_MemberOfCollection {

   [Override ( "Collection" ), Description (
          "The NodeRedundancySet that contains NodeSystems.")]
   TPD_NodeRedundancySet REF Collection;

   [Override ( "Member" ), Description (
          "The NodeSystem that is contained in the NodeRedundancySet.")]
   TPD_NodeSystem REF Member;
};

// ==================================================================
// 3PAR NodePairMemberOfSet
// ==================================================================
[Association, Aggregation, Description (
  "TPD_NodePairMemberOfSet is an aggregation used to establish "
  "membership of TPD_NodePairSystem in a TPD_NodePairRedundancySet.")]
class TPD_NodePairMemberOfSet : CIM_MemberOfCollection {

   [Override ( "Collection" ), Description (
          "The TPD_NodePairRedundancySet that contains TPD_NodePairSystem.")]
   TPD_NodePairRedundancySet REF Collection;

   [Override ( "Member" ), Description (
          "The TPD_NodePairSystem that is contained "
          "in the TPD_NodePairRedundancySet.")]
   TPD_NodePairSystem REF Member;
};

// ==================================================================
// 3PAR NodeSetIdentity
// ==================================================================
[Association, Description (
       "TPD_NodeSetIdentity associates two elements representing "
       "different aspects of the same underlying entity: TPD_NodePairSystem"
       " and TPD_NodeRedundancySet.")]
class TPD_NodeSetIdentity : CIM_ConcreteIdentity {
```

```
   [Override ( "SystemElement" ), Description (
          "NodePairSystem associated to the NodeRedundancySet")]
   TPD_NodePairSystem REF SystemElement;

      [Override ( "SameElement" ), Description (
          "NodeRedundancySet for a NodePairSystem.")]
   TPD_NodeRedundancySet REF SameElement;
};

// ===================================================================
// 3PAR NodePairSetIdentity
// ===================================================================
[Association, Description (
       "TPD_NodePairSetIdentity associates two elements representing "
       "different aspects of the same underlying entity: TPD_StorageSystem"
       " and TPD_NodePairRedundancySet.")]
class TPD_NodePairSetIdentity : CIM_ConcreteIdentity {

   [Override ( "SystemElement" ), Description (
          "StorageSystem associated to the NodePairRedundancySet")]
   TPD_StorageSystem REF SystemElement;

      [Override ( "SameElement" ), Description (
          "NodePairRedundancySet for a StorageSystem.")]
   TPD_NodePairRedundancySet REF SameElement;
};

// ===================================================================
// NodePairComponentCS
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "A ComputerSystem can aggregate another ComputerSystem. This"
       "class groups together TPD_StorageSystem and TPD_NodePairSystem.")]
class TPD_NodePairComponentCS : CIM_ComponentCS {

      [Aggregate, Override ( "GroupComponent" ), Description (
          "The ComputerSystem that contains and/or aggregates other "
          "Systems.")]
   TPD_StorageSystem REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The contained (Sub)ComputerSystem.")]
   TPD_NodePairSystem REF PartComponent;
};

// ===================================================================
// NodeComponentCS
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "A ComputerSystem can aggregate another ComputerSystem. This"
       "class groups together TPD_StorageSystem and TPD_NodeSystem.")]
class TPD_NodeComponentCS : CIM_ComponentCS {

      [Aggregate, Override ( "GroupComponent" ), Description (
          "The ComputerSystem that contains and/or aggregates other "
          "Systems.")]
   TPD_StorageSystem REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The contained (Sub)ComputerSystem.")]
   TPD_NodeSystem REF PartComponent;
};

// ===================================================================
```

```
// Node CPU
// ==================================================================
class TPD_NodeCPU : CIM_Processor
{
   [Description ("Position of the CPU in the node.") ]
   uint16 Position;

   [Description ("Bus speed in Mhz * 100") ]
   uint32 BusSpeed;

   [Description ("Vendor name of the CPU.") ]
   string Manufacturer;

   [Description ("Model type for CPU.") ]
   string Model;

   [Description ("Serial number of the CPU.") ]
   string SerialNumber;
};

// ==================================================================
// ProcessorForNode
// ==================================================================
[Association, Aggregation, Composition,
    Description (
       "NodeCPU aggregated by a NodeSystem.")]
class TPD_ProcessorForNode : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent Node in the Association.")]
   TPD_NodeSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The NodeCPU that is a component of a Node.")]
   TPD_NodeCPU REF PartComponent;
};

// ==================================================================
// TPD Physical Memory (DIMM)
// ==================================================================
[Description (
   "3PAR DIMM in Controller Node.")]
class TPD_PhysicalMemory : CIM_PhysicalMemory
{
   [Description ("Slot number, memory is inserted in.") ]
   uint16 Slot;

   [Description ("Slot ID, memory is inserted in.") ]
   string SlotID;

   [Description ("Type of the cache memory is used for."),
           ValueMap {"0", "1"},
           Values {"Control (CPU)", "Data (Cluster)"} ]
   uint16 CacheType;

   [Description ("Single or double sided.  Not supported yet."),
           ValueMap {"0"},
           Values {"Unknown"} ]
   uint16 SideType;

   [Description ("Standard or stacked chips. Not supported yet."),
           ValueMap {"0"},
           Values {"Unknown"} ]
   uint16 ChipsType;
```

```
   [Description ("CAS latency.  Format: CL<min>/<max>, in nS.") ]
   string CASLatency;

   [Description ("JEDEC ID.") ]
   string JedecID;

   [Description ("Revision code.") ]
   string Revision;
};

// ===================================================================
// TPD NodePackagedMemory
// ===================================================================
[Association, Aggregation, Description (
   "A Physical Memory contained by a Node PhysicalPackage.")]
class TPD_NodePackagedMemory : CIM_PackagedComponent {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The NodePackage that contains PhysicalMemory(s).")]
   TPD_NodePackage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The PhysicalMemory which is contained in the NodePackage.")]
   TPD_PhysicalMemory REF PartComponent;
};

// ===================================================================
// TPD IDE Drive
// ===================================================================
[Description (
   "3PAR IDE Drive in Controller Node.")]
class TPD_IDEDrive : CIM_PhysicalComponent
{
   [Description ("Position of the IDE Drive in the Node.") ]
   uint16 Position;

   [Description ("Firmware Revision code.") ]
   string FirmwareVersion;

   [Description ("Max Logical Block Address.") ]
   uint64 MaxLBA;

   [Description ("Capacity of the drive in bytes.") ]
   uint64 Capacity;
};

// ===================================================================
// TPD NodePackagedIDE
// ===================================================================
[Association, Aggregation, Description (
   "An IDE Drive contained by a Node PhysicalPackage.")]
class TPD_NodePackagedIDE : CIM_PackagedComponent {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The NodePackage that contains IDE Drive(s).")]
   TPD_NodePackage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The IDE Drive which is contained in the NodePackage.")]
   TPD_IDEDrive REF PartComponent;
};

// ===================================================================
// TPD PCI or PCIe Card
```

```
// ===================================================================
[Description (
   "3PAR PCI or PCIe Card.")]
class TPD_PCICard : CIM_Card
{
   [Description ("Node slot the card is in.") ]
   uint16 Slot;

   [Description ("Firmware version on the PCI or PCIe Card.") ]
   string FirmwareVersion;

   [Description ("Revision code.") ]
   String Revision;

   [Description ("Type of the PCI or PCIe Card."),
      ValueMap {"0", "1", "2", "3", "4"},
      Values {"Unknown", "FC", "Ethernet", "iSCSI", "CNA"} ]
   uint16 Type;
};

// ===================================================================
// 3PAR Node Package and PCI or PCIe Card association
// ===================================================================
   [Association, Aggregation,
      Description (
       "3PAR Node Package and PCI or PCIe Card association")]
class TPD_PCICardInNode : CIM_Container {

      [Aggregate, Override ( "GroupComponent" ), Max ( 1 ),
       Description (
          "The Node Package that contains PCI or PCIe cards")]
   TPD_NodePackage REF GroupComponent;

      [Override ( "PartComponent" ), Description (
          "The PCI or PCIe card that is contained in the Node Package.")]
   TPD_PCICard REF PartComponent;
};

// ===================================================================
// 3PAR SystemEndpoint: association between NodeSystem and
// SCSIProtocolEndpoint
// ===================================================================
[Association,
    Description
     ("3PAR InServ NodeSystem and SCSIProtocolEndpoint association."
      "This is a subclass of CIM_HostedAccessPoint.  There are many "
      "subclasses of CIM_HostedAcessPoint in the system.  This class is "
      " named TPD_SystemEndpoint to distinguish with other subclasses. ")]
class TPD_SystemEndpoint : CIM_HostedAccessPoint
{
    // Indicate that this is one-to-many association
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ node controller that hosted the protocol endpoint.")]
    TPD_NodeSystem REF Antecedent;

      [Override ( "Dependent" ), Weak, Description (
          "The SCSIProtocolEndpoint representing front end port "
     "on the system.")]
    TPD_SCSIProtocolFCEndpoint REF Dependent;
    // CIM_SCSIProtocolEndpoint REF Dependent; -- reenable this line if we support
TPD_SCSIInitTargetLUPath
};

// ===================================================================
// 3PAR NodeSystem and FCPort association
// ===================================================================
```

```
[Association, Aggregation, Composition,
    Description (
        "3PAR InServ Node System and FC Port Association ") ]
class TPD_SystemFCPort : CIM_SystemDevice
{
    [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
   Description("The InServ node controller that hosted the FC port.")]
    TPD_NodeSystem REF GroupComponent;

    [Override ("PartComponent"), Weak, Description (
        "The front end FCPort port.") ]
    TPD_FCPort REF PartComponent;
};

// ===================================================================
// 3PAR Storage System and SCSI Controller association
// ===================================================================
[Association, Aggregation, Composition,
    Description (
        "3PAR InServ Node System and SCSI Controller Association ") ]
class TPD_SystemController : CIM_SystemDevice
{
    [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
   Description("The InServ system or node controller that associates with "
   "the SCSI Protocol controller.")]
    CIM_ComputerSystem REF GroupComponent;

    [Override ("PartComponent"), Weak, Description (
        "The SCSI controller on the node system.") ]
    TPD_SCSIController REF PartComponent;
};
```

3PAR_TPDEnv.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDEnv.mof
//
//  Purpose : This MOF contains 3PAR Environment classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// TPD Power Supply
// ===================================================================
[Description (
       "Capabilities and management of the 3PAR PowerSupply.")]
class TPD_PowerSupply : CIM_PowerSupply
{
   [Description ("Manufacturer name of the PS.") ]
```

```
   string Manufacturer;

   [Description ("Model type for PS.") ]
   string Model;

   [Description ("Serial number of the PS.") ]
   string SerialNumber;

   [Description ("Position of the PS.") ]
   uint16 Position;

   [Description ("State of the AC for the power supply."),
      ValueMap {"0", "1", "2"},
      Values {"Unknown", "OK", "Error"} ]
   uint16 ACStatus;

};

// ==================================================================
// TPD Cage Power Supply
// ==================================================================
[Description (
       "Capabilities and management of the 3PAR Cage PowerSupply.")]
class TPD_CagePowerSupply : TPD_PowerSupply
{
   [Description ("Type of the cage this Power Supply is in."),
      ValueMap {"0", "2", "3", "4", "5"},
      Values {"Unknown", "DC1", "DC2", "DC3", "DC4"} ]
   uint16 CageType;

   [Description ("If TRUE, user can not modify model.") ]
   boolean ModelReadOnly;
};

// ==================================================================
// TPD Node Power Supply
// ==================================================================
[Description (
       "Capabilities and management of the 3PAR Node PowerSupply.")]
class TPD_NodePowerSupply : TPD_PowerSupply
{
    [Description ("State of the DC for the node power supply."),
        ValueMap {"0", "1", "2"},
        Values {"Unknown", "OK", "Error"} ]
    uint16 DCStatus;
};

// ==================================================================
// CagePSDependency
// ==================================================================
   [Association, Description (
       "A PowerSupply is installed in a DriveCage, not for a specific Device, but "
       "to function with the DriveCage in general. This relationship is "
       "described by the TPD_CagePSDependency association.")]
class TPD_CagePSDependency : CIM_PackageDependency {

      [Override ( "Antecedent" ), Description (
          "The PowerSupply for the DriveCage.")]
   TPD_PowerSupply REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The DriveCage whose environment is affected.")]
   TPD_DriveCage REF Dependent;
};

// ==================================================================
```

```
// TPD Fan
// Based on the CIM schema, we need to use CIM_NumericSensor (Tachometer)
// to show the current speed of the fan.  This is way too much for what
// we need, so I am going to add a new variable for speed.  If
// we ever need to comply with the requirements, I will change it.
// ===================================================================
[Description (
   "3PAR Fan ")]
class TPD_Fan : CIM_Fan
{
   [Description ("Position of the Fan.") ]
   uint16 Position;

   [Description ("Fan speed."),
           ValueMap {"0", "1", "2", "3", "4"},
           Values {"Unknown", "Hi", "Low", "Medium", "Stopped"} ]
   uint16 Speed;

   [Description ("3PAR specific operational state for the Fan"),
          ValueMap {"0", "1", "2", "4", "5", "6", "7" },
       Values {"Unknown", "OK", "Error", "Critical", "Non Critical",
             "Not Installed", "Not Available"} ]
   uint16 OtherOperationalStatus;

   [Description ("Fan LED status.  This is ONLY available for system fans."),
   ValueMap {"0", "1", "2", "3", "4", "5"},
   Values {"Unknown", "Off", "Green", "Green Blinking", "Amber", "Amber Blinking"} ]
   uint16 SystemFanLED;
};

// ===================================================================
// PowerSupply Cooling
// ===================================================================
   [Association, Description (
       "Many Devices, such as processors or power supplies, require "
       "their own cooling devices. This association indicates where "
       "fans or other CoolingDevices are specific to a Device, versus "
       "providing enclosure or cabinet cooling.")]
class TPD_PowerSupplyCooling : CIM_AssociatedCooling {

      [Override ( "Antecedent" ), Description (
          "The Fan.")]
   TPD_Fan REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The PowerSupply being cooled.")]
   TPD_PowerSupply REF Dependent;
};

// ===================================================================
// Node Cooling
// ===================================================================
   [Association, Description (
       "Describes association between SystemFan and "
       "the Node Package it is cooling.")]
class TPD_NodePkgCooling : CIM_PackageDependency {

      [Override ( "Antecedent" ), Description (
          "The System Fan.")]
   TPD_Fan REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The Node being cooled.")]
   TPD_NodePackage REF Dependent;
};
```

```
// ===================================================================
// TPD Battery
// ===================================================================
[Description (
   "3PAR Battery connected to the PowerSupply in Node.")]
class TPD_Battery : CIM_Battery
{
   [Description ("Position of the Battery.") ]
   uint16 Position;

   [Description ("Maximum battery life in minutes.") ]
   uint32 MaxLife;

   [Description ("If TRUE, MAX life is low and battery is failed.") ]
   boolean MaxLifeLow;

   [Description ("Manufacturing date of the Battery.") ]
   datetime ManufacturingDate;

   [Description ("Expiration date of the Battery.") ]
   datetime ExpirationDate;

   [Description ("Charge start date of the Battery.") ]
   datetime ChargeStartDate;

   [Description ("Vendor name of the Battery.") ]
   string Manufacturer;

   [Description ("Model type for Battery.") ]
   string Model;

   [Description ("Serial number of the Battery.") ]
   string SerialNumber;

   [Description ("TRUE if the test is running.") ]
   boolean TestInProgress;

   [Description ("3PAR specific operational state for the Battery"),
          ValueMap {"0", "1", "2"},
       Values {"Unknown", "OK", "Error"} ]
   uint16 OtherOperationalStatus;

   // ===================================================================
   // The next 3 attributes describe log data for tests.  Each index in the
   // arrays represents single test log.  All 3 attributes must be of the
   // same length.
   // ===================================================================
   [Description ("The date the test has started."), ArrayType ( "Indexed" ),
         ModelCorrespondence {"TPD_Battery.TestDuration",
                     "TPD_Battery.TestPassed"}]
   datetime TestStartDate[];

   [Description ("The duration of the test in seconds."), ArrayType ( "Indexed" ),
         ModelCorrespondence {"TPD_Battery.TestStartDate",
                     "TPD_Battery.TestPassed"} ]
   uint16 TestDuration[];

   [Description ("TRUE if the test passed, FALSE otherwise."), ArrayType ( "Indexed"
 ),
         ModelCorrespondence {"TPD_Battery.TestStartDate",
                     "TPD_Battery.TestDuration"} ]
   boolean TestPassed[];
};

// ===================================================================
// TPD_PowerSupplyBattery
```

```
// ===================================================================
   [Association, Description (
       "A Node PowerSupply may use or require one or more Batteries. This "
       "relationship is described by the PowerSupplyBattery dependency.")]
class TPD_PowerSupplyBattery : CIM_AssociatedBattery
{
      [Override ( "Antecedent" ), Description (
          "The Battery.")]
   TPD_Battery REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The Node PowerSupply needing or associated with the Battery.")]
   TPD_PowerSupply REF Dependent;
};

// ===================================================================
// NodePkgPSDependency
// ===================================================================
   [Association, Description (
       "A PowerSupply is installed in a NodePackage, not for a specific "
       "Device, but to function with the NodePackage in general. This "
       "relationship is described by the TPD_NodePkgPSDependency "
       "association.")]
class TPD_NodePkgPSDependency : CIM_PackageDependency {

      [Override ( "Antecedent" ), Description (
          "The PowerSupply for the NodePackage.")]
   TPD_PowerSupply REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The NodePackage whose environment is affected.")]
   TPD_NodePackage REF Dependent;
};

// ===================================================================
// TPD_Battery and TPD_NodeSystem association
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "The NodeBattery association represents a relationship between "
       "a TPD_Battery and TPD_NodeSystem.")]
class TPD_NodeBattery : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent system in the Association.")]
   TPD_NodeSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The Battery that is a component of a NodeSystem.")]
   TPD_Battery REF PartComponent;
};

// ===================================================================
// TPD_PowerSupply and CIM_ComputerSystem association
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "The SystemPowerSupply association represents a relationship between "
       "a TPD_PowerSupply and CIM_ComputerSystem (TPD_NodeSystem or "
       "TPD_StorageSystem).")]
class TPD_SystemPowerSupply : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent system in the Association.  Could be either "
```

```
          "StorageSystem or NodeSystem.")]
   CIM_ComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The PowerSupply that is a component of a ComputerSystem.")]
   TPD_PowerSupply REF PartComponent;
};

// ===================================================================
// TPD_Fan and CIM_ComputerSystem association
// ===================================================================
   [Association, Aggregation, Composition,
    Description (
       "The SystemPowerSupply association represents a relationship between "
       "a TPD_Fan and CIM_ComputerSystem (TPD_NodeSystem or "
       "TPD_StorageSystem).")]
class TPD_SystemFan : CIM_SystemDevice {

      [Aggregate, Override ( "GroupComponent" ), Min ( 1 ), Max ( 1 ),
       Description (
          "The parent system in the Association.  Could be either "
          "StorageSystem or NodeSystem.")]
   CIM_ComputerSystem REF GroupComponent;

      [Override ( "PartComponent" ), Weak, Description (
          "The Fan that is a component of a ComputerSystem.")]
   TPD_Fan REF PartComponent;
};

// ===================================================================
//  TPD Cage PowerSupply Sensor
// ===================================================================
[Association, Description (
   "Many Devices include Sensors or have Sensors installed nearby, "
   "in order to measure critical input and output properties. This "
   "association indicates that relationship.")]
class TPD_SensorInCagePS : CIM_AssociatedSensor {

   [Override ( "Antecedent" ), Description (
      "The Sensor.")]
   TPD_CageSensor REF Antecedent;

   [Override ( "Dependent" ), Description (
      "The LogicalDevice for which information is measured by the "
      "Sensor.")]
   TPD_CagePowerSupply REF Dependent;
};
```

3PAR_TPDLocation.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////
//

////////////////////////////////////////////////////////////////////////
```

```
//
//
//  File    : 3PAR_TPDLocation.mof
//
//  Purpose : This MOF contains 3PAR Location classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////
//

// ===================================================================
// TPD System location
// ===================================================================
[Description (
 "Location of the InServ array.")]
class TPD_SystemLocation : CIM_Location
{
};

// ===================================================================
// TPD System Package and Location association
// ===================================================================
   [Association, Description (
       "SystemPackageLocation associates a SystemPackage with a "
       "Location object for inventory or replacement purposes.")]
class TPD_SystemPackageLocation : CIM_PhysicalElementLocation {

      [Override ( "Element" ), Description (
          "The PhysicalElement whose Location is specified.")]
   TPD_SystemPackage REF Element;

      [Override ( "PhysicalLocation" ), Max ( 1 ), Description (
          "The PhysicalElement's Location.")]
   TPD_SystemLocation REF PhysicalLocation;
};
```

3PAR_TPDEthPort.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDEthPort.mof
//
//  Purpose : This MOF contains 3PAR Ethernet Port classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 8/10/2005
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// 3PAR EthernetPort
```

```
// ===================================================================
[Description (
       "Capabilities and management of an EthernetPort.")]
class TPD_EthernetPort : CIM_EthernetPort {

      [Description ("TPD specific operational status"),
      ValueMap {"3", "4", "9"},
      Values {"ready", "loss sync", "offline"} ]
        uint16 OtherOperationalStatus;
};

// ===================================================================
// 3PAR IPProtocolEndpoint
// ===================================================================
[Description (
       "A ProtocolEndpoint that is dedicated to running IP.")]
class TPD_IPProtocolEndpoint : CIM_IPProtocolEndpoint {

         [Description (
          "The Gateway address for this ProtocolEndpoint.")]
      string Gateway;
};

// ===================================================================
// 3PAR EthPortSAPImplementation
// ===================================================================
[Association, Description (
       "An association between a ServiceAccessPoint and how it is "
       "implemented.")]
class TPD_EthPortSAPImplementation : CIM_DeviceSAPImplementation {

   [Override ( "Antecedent" ), Description (
      "The Ethernet Port.")]
   TPD_EthernetPort REF Antecedent;

   [Override ( "Dependent" ), Description (
      "The IP Protocol Endpoint implemented using the Ethernet Port.")]
   TPD_IPProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR SystemIPEndpoint: association between NodeSystem and
// IPProtocolEndpoint
// ===================================================================
[Association,
    Description
     ("3PAR InServ NodeSystem and IPProtocolEndpoint association."
      "This is a subclass of CIM_HostedAccessPoint.  There are many "
      "subclasses of CIM_HostedAcessPoint in the system.  This class is "
      " named TPD_SystemIPEndpoint to distinguish with other subclasses. ")]
class TPD_SystemIPEndpoint : CIM_HostedAccessPoint
{
    // Indicate that this is one-to-many association
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ node controller that hosted the protocol endpoint.")]
    TPD_NodeSystem REF Antecedent;

      [Override ( "Dependent" ), Weak, Description (
          "The IPProtocolEndpoint representing ethernet port "
     "on the system.")]
    TPD_IPProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR NodeSystem and EthernetPort association
// ===================================================================
```

```
[Association, Aggregation, Composition,
   Description (
      "3PAR InServ Storage System and Ethernet Port Association ") ]
class TPD_SystemEthPort : CIM_SystemDevice
{
    [Override ("GroupComponent"), Aggregate, Max (1), Min (1),
   Description("The InServ node controller that hosted the Ethernet port.")]
    TPD_NodeSystem REF GroupComponent;

    [Override ("PartComponent"), Weak, Description (
        "The Ethernet Port.") ]
    TPD_EthernetPort REF PartComponent;
};

// ===================================================================
// 3PAR TCPProtocolEndpoint
// ===================================================================
[Description (
   "A TPD protocol endpoint that is dedicated to running TCP.")]
class TPD_TCPProtocolEndpoint : CIM_TCPProtocolEndpoint {

};

// ===================================================================
// 3PAR BindsToIPEndpoint
// ===================================================================
[Association, Description (
   "This association makes explicit the dependency of a "
   "TCPProtocolEndpoint on an underlying IPProtocolEndpoint, on the same "
   "system.")]
class TPD_BindsToIPEndpoint : CIM_BindsTo {

      [Override ( "Antecedent" ), Description (
          "The underlying IPProtocolEndpoint, which is depended upon.")]
   TPD_IPProtocolEndpoint REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The TCPProtocolEndpoint dependent on the IPProtocolEndpoint.")]
   TPD_TCPProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR SystemTCPEndpoint: association between NodeSystem and
// TCPProtocolEndpoint
// ===================================================================
[Association,
    Description
     ("3PAR InServ NodeSystem and IPProtocolEndpoint association."
      "This is a subclass of CIM_HostedAccessPoint.  There are many "
      "subclasses of CIM_HostedAcessPoint in the system.  This class is "
      " named TPD_SystemIPEndpoint to distinguish with other subclasses. ")]
class TPD_SystemTCPEndpoint : CIM_HostedAccessPoint
{
   // Indicate that this is one-to-many association
   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
      "The InServ node controller that hosted the protocol endpoint.")]
   TPD_NodeSystem REF Antecedent;

   [Override ( "Dependent" ), Weak, Description (
      "The TCPProtocolEndpoint representing ethernet port "
      "on the system.")]
   TPD_TCPProtocolEndpoint REF Dependent;
};
```

3PAR_TPDiSCSI.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDiSCSI.mof
//
//  Purpose : This MOF contains 3PAR iSCSI classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 1/31/2006
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
//  3PAR iSCSIController
// ===================================================================
[Description (
   "iSCSIController is a type of SCSIProtocolController, "
   "managing an iSCSI interface.")]
class TPD_iSCSIController : CIM_SCSIProtocolController {

};

[Description (
   "The iSCSI Node represents a single iSCSI Target. "
   "There are one or more iSCSI Nodes within a Network "
   "Entity. The iSCSI Node is accessible via one or more "
   "Network Portals. An iSCSI Node is identified by its iSCSI "
   "Name.")]
class TPD_iSCSINode : CIM_SCSIProtocolController {

};

// ===================================================================
// 3PAR iSCSIProtocolEndpoint
// ===================================================================
class TPD_iSCSIProtocolEndpoint : CIM_iSCSIProtocolEndpoint {

};

// ===================================================================
// 3PAR BindsToTCPEndpoint
// ===================================================================
[Association, Description (
   "This association makes explicit the dependency of a "
   "iSCSIProtocolEndpoint on an underlying TCPProtocolEndpoint, on the same "
   "system.")]
class TPD_BindsToTCPEndpoint : CIM_BindsTo {

      [Override ( "Antecedent" ), Description (
          "The underlying TCPProtocolEndpoint, which is depended upon.")]
   TPD_TCPProtocolEndpoint REF Antecedent;

      [Override ( "Dependent" ), Description (
```

```
          "The iSCSIProtocolEndpoint dependent on the TCPProtocolEndpoint.")]
   TPD_iSCSIProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR SystemiSCSIEndpoint: association between NodeSystem and
// iSCSIProtocolEndpoint
// ===================================================================
[Association, Description (
   "3PAR InServ NodeSystem and IPProtocolEndpoint association."
   "This is a subclass of CIM_HostedAccessPoint.  There are many "
   "subclasses of CIM_HostedAcessPoint in the system.  This class is "
   " named TPD_SystemIPEndpoint to distinguish with other subclasses. ")]
class TPD_SystemiSCSIEndpoint : CIM_HostedAccessPoint
{
   // Indicate that this is one-to-many association
   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
      "The InServ node controller that hosted the protocol endpoint.")]
   TPD_NodeSystem REF Antecedent;

   [Override ( "Dependent" ), Weak, Description (
      "The iSCSIProtocolEndpoint representing ethernet port "
      "on the system.")]
   TPD_iSCSIProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR iSCSISAPImplementation
// ===================================================================
[Association, Description (
   "An association between a ServiceAccessPoint and how it is "
   "implemented.")]
class TPD_iSCSISAPImplementation : CIM_DeviceSAPImplementation {

   [Override ( "Antecedent" ), Description (
      "The Ethernet Port.")]
   TPD_EthernetPort REF Antecedent;

   [Override ( "Dependent" ), Description (
      "The iSCSI Protocol Endpoint implemented using the Ethernet Port.")]
   TPD_iSCSIProtocolEndpoint REF Dependent;
};

// ===================================================================
// 3PAR PrivilegeForiSCSIController
// ===================================================================
[Association, Description (
       "PrivilegeForiSCSIController is an association used to tie the "
       "AuthorizedPrivileges to iSCSIController resources.")]
class TPD_PrivilegeForiSCSIController : CIM_AuthorizedTarget {

      [Override ( "Privilege" ), Description (
          "The AuthorizedPrivilege affecting the iSCSIProtocolController "
     "resource.")]
   TPD_AuthorizedPrivilege REF Privilege;

      [Override ( "TargetElement" ), Description (
          "The iSCSIProtocolController resources to which the "
          "AuthorizedPrivilege applies.")]
   TPD_iSCSIController REF TargetElement;
};

// ===================================================================
// 3PAR ConfigServicesForiSCSIController Association
// ===================================================================
[Association,
```

```
    Description (
        "TPD_ConfigServicesForiSCSIController is an association between "
        "TPD_iSCSIController and ControllerConfigurationService.") ]
class TPD_ConfigServicesForiSCSIController : CIM_ConcreteDependency
{
    [Override ( "Antecedent" ), Description (
          "The controller configuration service which the protocol "
          "controller depends on for configuration.")]
    TPD_ControllerConfigurationService REF Antecedent;

    [Override ( "Dependent" ), Description (
          "The iSCSI protocol controller which uses the controller "
          "configuration service.")]
    TPD_iSCSIController REF Dependent;
};

// ===================================================================
// 3PAR iSCSI Protocol Controller And StorageVolume association
// ===================================================================
[Association,
    Description (
    "3PAR InServ iSCSI protocol controller and storage volume "
    "association. ")]
class TPD_iSCSIControllerForUnit : CIM_ProtocolControllerForUnit
{
    [Override ("Antecedent"),
        Description ("The iSCSI Protocol Controller.") ]
    TPD_iSCSIController REF Antecedent;

    [Override ("Dependent"), Weak, Description (
        "The storage volume controlled by the iSCSI protocol controller.") ]
    TPD_StorageVolume REF Dependent;
};

// ===================================================================
//  3PAR iSCSISessionSettings
// ===================================================================
[Description (
   "The default negotiation settings for an iSCSI Session. These "
   "properties are used as the starting position for login "
   "negotiations between initiator and target nodes. The "
   "properties describing the resulting outcome of the login "
   "negotiation are in the iSCSISession class.")]
class TPD_iSCSISessionSettings : CIM_iSCSISessionSettings {

};

// ===================================================================
//  3PAR iSCSISessionSystemSetting
// ===================================================================
[Association, Description (
   "ElementSettingData represents the association between "
   "ManagedElements and applicable setting data. This association "
   "also describes whether this is a default or current setting.")]
class TPD_iSCSISessionSystemSetting : CIM_ElementSettingData {

      [Override ( "ManagedElement" ), Description (
          "The managed element.")]
   TPD_StorageSystem REF ManagedElement;

      [Override ( "SettingData" ), Description (
          "The SettingData object associated with the element.")]
   TPD_iSCSISessionSettings REF SettingData;
};

// ===================================================================
```

```
//  3PAR iSCSIConnectionSettings
// ===================================================================
[Description (
   "The settings for the usage of an iSCSI NetworkPortal by an "
   "iSCSIProtcolEndpoint. These settings are the starting point "
   "for negotiation for connection establishment. "
   "The properties that reflect the actual outcome "
   "of the negotiation are found in the iSCSIConnection class.")]
class TPD_iSCSIConnectionSettings : CIM_iSCSIConnectionSettings {

};

// ===================================================================
//  3PAR iSCSIConnectionTCPSetting
// ===================================================================
[Association, Description (
   "ElementSettingData represents the association between "
   "ManagedElements and applicable setting data. This association "
   "also describes whether this is a default or current setting.")]
class TPD_iSCSIConnectionTCPSetting : CIM_ElementSettingData {

      [Override ( "ManagedElement" ), Description (
          "The managed element.")]
   TPD_TCPProtocolEndpoint REF ManagedElement;

      [Override ( "SettingData" ), Description (
          "The SettingData object associated with the element.")]
   TPD_iSCSIConnectionSettings REF SettingData;
};

// ===================================================================
//  3PAR iSCSIConnectionSettingEndpoint
// ===================================================================
[Association, Description (
   "ElementSettingData represents the association between "
   "ManagedElements and applicable setting data. This association "
   "also describes whether this is a default or current setting.")]
class TPD_iSCSIConnectionSettingEndpoint : CIM_ElementSettingData {

      [Override ( "ManagedElement" ), Description (
          "The managed element.")]
   TPD_iSCSIProtocolEndpoint REF ManagedElement;

      [Override ( "SettingData" ), Description (
          "The SettingData object associated with the element.")]
   TPD_iSCSIConnectionSettings REF SettingData;
};

// ===================================================================
//  3PAR iSCSISession
// ===================================================================
[Description (
   "iSCSISession is a network pipe between an initiator and target "
   "SCSIProtocolEndpoints. An iSCSISession is composed of one or "
   "more TCP connections which MUST be selected from a "
   "SystemSpecificCollection representing an iSCSI Portal Group. "
   "NetworkPipeComposition aggregates NetworkPipe instances "
   "representing iSCSI connections, which are associated to "
   "TCPProtocolEndpoints. Only an iSCSI initiator can create an "
   "iSCSI Session, an iSCSI Target MUST accept (or reject) a "
   "session request. EndOfNetworkPipe associates iSCSISession with "
   "SCSIProtocolEndpoint.")]
class TPD_iSCSISession : CIM_iSCSISession {

};
```

```
// ===================================================================
//  3PAR EndpointOfiSCSISession
// ===================================================================
[Association, Description (
   "EndpointOfNetworkPipe describes the endpoints between which a "
   "pipe transfers information. Whether an endpoint is a source or "
   "sink is indicated by a property of the association, "
   "SourceOrSink.")]
class TPD_EndpointOfiSCSISession : CIM_EndpointOfNetworkPipe {

      [Override ( "Antecedent" ), Min ( 2 ), Max ( 2 ), Description (
          "One of the endpoints of the pipe.")]
   TPD_iSCSIProtocolEndpoint REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The pipe which is dependent on the endpoints as the source "
          "or sink of the transferred information.")]
   TPD_iSCSISession REF Dependent;
};

// ===================================================================
//  3PAR iSCSIConnection
// ===================================================================
[Description (
   "This class contains the attributes of and negotiated values "
   "for, an iSCSI Connection which is modeled as a subclass of "
   "NetworkPipe. The original settings that are a starting point "
   "for negotiation are found in the class "
   "iSCSIConnectionSettings.")]
class TPD_iSCSIConnection : CIM_iSCSIConnection {

};

// ===================================================================
//  3PAR EndpointOfiSCSIConnection
// ===================================================================
[Association, Description (
   "EndpointOfNetworkPipe describes the endpoints between which a "
   "pipe transfers information. Whether an endpoint is a source or "
   "sink is indicated by a property of the association, "
   "SourceOrSink.")]
class TPD_EndpointOfiSCSIConnection : CIM_EndpointOfNetworkPipe {

      [Override ( "Antecedent" ), Min ( 2 ), Max ( 2 ), Description (
          "One of the endpoints of the pipe.")]
   TPD_TCPProtocolEndpoint REF Antecedent;

      [Override ( "Dependent" ), Description (
          "The pipe which is dependent on the endpoints as the source "
          "or sink of the transferred information.")]
   TPD_iSCSIConnection REF Dependent;
};

// ===================================================================
//  3PAR iSCSIPipeComposition
// ===================================================================
[Association, Aggregation, Composition, Description (
   "NetworkPipeComposition describes the makeup a pipe, based on "
   "lower-level ones. If the pipe is not composed of lower-level "
   "entities (i.e., its AggregationBehavior property is set to 2), "
   "then no instances of this association should be defined where "
   "the pipe has the role of GroupComponent.")]
class TPD_iSCSIPipeComposition : CIM_NetworkPipeComposition {

   [Aggregate, Override ( "GroupComponent" ), Description (
       "The higher level pipe that is composed of lower-level "
```

```
       "parts/pipes.")]
   TPD_iSCSISession REF GroupComponent;

   [Override ( "PartComponent" ), Description (
       "A pipe which is a part of a higher-level one.")]
   TPD_iSCSIConnection REF PartComponent;
};

// ===================================================================
//  3PAR iSCSISessionStatistics
// ===================================================================
[Description (
   "Traffic and error statistics for an iSCSI Session. An instance "
   "of this class will be associated by ElementStatisticalData to "
   "the instance of iSCSISession.")]
class TPD_iSCSISessionStatistics : CIM_iSCSISessionStatistics {
};

// ===================================================================
//  3PAR iSCSISessionStatisticalData
// ===================================================================
[Association, Description (
   "CIM_ElementStatisticalData is an association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_iSCSISessionStatisticalData : CIM_ElementStatisticalData {

      [Key, Min ( 1 ), Max ( 1 ), Description (
          "The ManagedElement for which statistical or metric data is "
          "defined.")]
   TPD_iSCSISession REF ManagedElement;

      [Key, Description (
          "The statistic information/object.")]
   TPD_iSCSISessionStatistics REF Stats;
};

// ===================================================================
//  3PAR iSCSICapabilities
// ===================================================================
[Description (
   "The capabilites for an iSCSI Network Entity. An instance of "
   "this class will be associated by ElementCapabilities to a "
   "instance of ComputerSystem that represents the Network Entity. "
   "These capability properties are associated to a Network "
   "Entity/ComputerSystem since they affect all login negotiations "
   "on all iSCSI ProtocolEndpoints aggregated to the system.")]
class TPD_iSCSICapabilities : CIM_iSCSICapabilities {

};

// ===================================================================
//  3PAR iSCSIElementCapabilities
// ===================================================================
[Association, Description (
   "ElementCapabilities represents the association between "
   "ManagedElements and their Capabilities. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementCapabilities association for the referenced instance of "
   "Capabilities. ElementCapabilities describes the existence "
```

```
   "requirements and context for the referenced instance of "
   "ManagedElement. Specifically, the ManagedElement MUST exist "
   "and provides the context for the Capabilities.")]
class TPD_iSCSIElementCapabilities : CIM_ElementCapabilities {

      [Key, Min ( 1 ), Max ( 1 ), Description (
          "The managed element.")]
   TPD_StorageSystem REF ManagedElement;

      [Key, Description (
          "The Capabilities object associated with the element.")]
   TPD_iSCSICapabilities REF Capabilities;
};

// ===================================================================
//  3PAR iSCSISAPAvailableForElement
// ===================================================================
[Association, Description (
   "CIM_SAPAvailableForElement conveys the semantics of a Service "
   "Access Point that is available for a ManagedElement. When "
   "CIM_SAPAvailableForElement is not instantiated, then the SAP "
   "is assumed to be generally available. If instantiated, the SAP "
   "is available only for the associated ManagedElements. For "
   "example, a device might provide management access through a "
   "URL. This association allows the URL to be advertised for the "
   "device.")]
class TPD_iSCSISAPAvailableForElement : CIM_SAPAvailableForElement {

   [Key, Description (
       "The Service Access Point that is available.")]
   TPD_iSCSIProtocolEndpoint REF AvailableSAP;

   [Key, Description (
       "The ManagedElement for which the SAP is available.")]
   CIM_SCSIProtocolController REF ManagedElement;
};

// ===================================================================
// 3PAR NodeSystem and iSCSIController association
// ===================================================================
[Association, Aggregation, Composition,
   Description (
      "3PAR InServ Storage System and iSCSI Controller Association ") ]
class TPD_SystemiSCSIController : CIM_SystemDevice
{
   [Override ("GroupComponent"), Aggregate, Max (1), Min (1), Description(
      "The InServ system or node controller that hosted the iSCSI Node.")]
   CIM_ComputerSystem REF GroupComponent;

   [Override ("PartComponent"), Weak, Description (
      "The iSCSI Controller or iSCSI Node.") ]
   CIM_SCSIProtocolController REF PartComponent;
};

// ===================================================================
//  3PAR iSCSISessionFailures
// ===================================================================
[Description (
   "Failure Statistics for Sessions associated with a iSCSI Node. "
   "An instance of this class will be associated by "
   "ElementStatisticalData to an instance of "
   "SCSIProtocolController representing an iSCSI Node. These "
   "statistics are associated to a Node since they describe the "
   "aggregated Session data for all failed Sessions associated to "
   "iSCSI protocol endpoints used by the Node.")]
class TPD_iSCSISessionFailures : CIM_iSCSISessionFailures {
```

```
};

// ===================================================================
//  3PAR iSCSILoginStatistics
// ===================================================================
[Description (
   "Statistics for Logins and Logouts to or from an iSCSI Node. An "
   "instance of this class will be associated by "
   "ElementStatisticalData to an instance of "
   "SCSIProtocolController that represents the Node. The Node can "
   "be either an Initiator or Target and so the interpretation of "
   "the properties in this class varies accordingly.")]
class TPD_iSCSILoginStatistics : CIM_iSCSILoginStatistics {

};

// ===================================================================
//  3PAR iSCSISessionFailuresData
// ===================================================================
[Association, Description (
   "CIM_ElementStatisticalData is an association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_iSCSISessionFailuresData : CIM_ElementStatisticalData {

      [Key, Min ( 1 ), Max ( 1 ), Description (
          "The ManagedElement for which statistical or metric data is "
          "defined.")]
   TPD_iSCSINode REF ManagedElement;

      [Key, Description (
          "The statistic information/object.")]
   TPD_iSCSISessionFailures REF Stats;
};

// ===================================================================
//  3PAR iSCSILoginStatisticsData
// ===================================================================
[Association, Description (
   "CIM_ElementStatisticalData is an association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_iSCSILoginStatisticsData : CIM_ElementStatisticalData {

      [Key, Min ( 1 ), Max ( 1 ), Description (
          "The ManagedElement for which statistical or metric data is "
          "defined.")]
   TPD_iSCSINode REF ManagedElement;

      [Key, Description (
          "The statistic information/object.")]
   TPD_iSCSILoginStatistics REF Stats;
};
```

3PAR_TPDJob.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//
//  Copyright 2005-2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////
//

////////////////////////////////////////////////////////////////////////////
//
//
//  File    : 3PAR_TPDJob.mof
//
//  Purpose : This MOF contains 3PAR Job Control classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 3/19/2007
//
////////////////////////////////////////////////////////////////////////////
//

// ===================================================================
// 3PAR Concrete Job
// ===================================================================
[Description (
   "3PAR Concrete Job")]
class TPD_ConcreteJob : CIM_ConcreteJob
{
   [Description ("Type of the job."),
   ValueMap {"7", "8", "10", "11"},
   Values {"Create Physical Copy", "Promote Virtual Copy",
           "Update Delta Snapshot Space", "Resync Physical Copy"}]
   uint16 JobType;

   [Description (
      "Source element that was affected by this job. For example, "
      "if this is a job to create a physical copy, then this "
      "will contain the ElementName of the base StorageVolume "
      "from which the replica StorageVolume is copied. Another example "
      "is if this is a job to promote virtual copy, then this "
      "will contain the ElementName of the replica StorageVolume."),
     ModelCorrespondence {
             "TPD_StorageVolume.ElementName"}]
    string SourceElementName;

   [Description (
      "Target element that was affected by this job. For example, "
      "if this is a job to create a physical copy, then this "
      "will contain the ElementName of the replica StorageVolume. "
      "Another example is if this is a job to promote virtual "
      "copy, then this will contain the ElementName of the "
      "base StorageVolume to which the replica StorageVolume "
      "is copied."),
     ModelCorrespondence {
             "TPD_StorageVolume.ElementName"}]
    string TargetElementName;
};
```

```
// ===================================================================
// 3PAR OwningJobElement
// ===================================================================
[Association,
    Description (
       "OwningJobElement represents an association between a "
       "TPD_ConcreteJob and the ManagedElement responsible for the "
       "creation of the Job, for example, TPD_StorageConfigurationService "
       "when a job is created as a side-effect of physical copy.")]
class TPD_OwningJobElement : CIM_OwningJobElement {
};

// ===================================================================
// 3PAR AffectedJobElement
// ===================================================================
[Association,
    Description (
       "AffectedJobElement represents an association between a "
       "TPD_ConcreteJob and the ManagedElement(s) that may be "
       "affected by its execution, for example, a TPD_StorageVolume "
       "when a job is created as a side-effect of physical copy.")]
class TPD_AffectedJobElement : CIM_AffectedJobElement
{
};

// ===================================================================
// 3PAR MethodResult
// ===================================================================
[Description (
       "Jobs are sometimes used to represent extrinsic method "
       "invocations that execute for times longer than the length of "
       "time is reasonable to require a client to wait. The method "
       "executing continues beyond the method return to the client. "
       "The class provides the result of the execution of a Job that "
       "was itself started by the side-effect of this extrinsic method "
       "invocation. \n"
       "The indication instances embedded an instance of this class "
       "MUST be the same indications delivered to listening clients or "
       "recorded, all or in part, to logs. Basically, this approach is "
       "a corollary to the functionality provided by an instance of "
       "ListenerDestinationLog (as defined in the Interop Model). The "
       "latter provides a comprehensive, persistent mechanism for "
       "recording Job results, but is also more resource-intensive and "
       "requires supporting logging functionality. Both the extra "
       "resources and logging MAY NOT be available in all environments "
       "(for example, embedded environments). Therefore, this "
       "instance-based approach is also provided. \n"
       "The MethodResult instances MUST NOT exist after the associated "
       "ConcreteJob is deleted.")]
class TPD_MethodResult : CIM_MethodResult {
};

// ===================================================================
// 3PAR AssociatedJobMethodResult
// ===================================================================
[Association,
    Description (
       "AssociatedJobMethodResult represents an association between a "
       "ConcreteJob and the MethodResult expressing the parameters for "
       "the Job when the job was created by side-effect of the "
       "execution of an extrinsic method.")]
class TPD_AssociatedJobMethodResult : CIM_AssociatedJobMethodResult {
};

// ===================================================================
// 3PAR Job Control Indication
```

```
// ===================================================================
[Description (
       "Indication sent when a property of a TPD_ConcreteJob changes "
       "value. This indication will only be sent if OperationlStatus "
       "of a TPD_ConcreteJob instance is changed.")]
class TPD_ConcreteJobInstModification : CIM_InstModification {
};
```

3PAR_TPDReplicationSvcs.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2010 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDReplicationSvcs.mof
//
//  Purpose : This MOF contains 3PAR Replication Services classes that will be loaded
//            into root/tpd namespace.
//
//  Date created: 1/19/2010
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// 3PAR Replication Services
// ===================================================================
[Description (
   "The ReplicationService class provides methods to allow a "
   "client to manage copy operations on storage objects, including "
   "management of replication groups, manipulation of replication "
   "operations, and retrieval of replication relationships.")]
class TPD_ReplicationService : CIM_ReplicationService
{
      [Description (
          "Create (or start a job to create) a new group of storage "
          "objects which are replicas of the specified source "
          "storage or a group of source storage objects "
          "(SourceElements). Note that using the input parameter, "
          "SyncType, this function can be used to instantiate the "
          "replicas, and to create ongoing associations between "
          "the source(s) and replicas. If 0 is returned, the "
          "function completed successfully and no ConcreteJob "
          "instance is created. If 4096/0x1000 is returned, a "
          "ConcreteJob is started, references to which is returned "
          "in the Jobs output parameter."
   "An array of references to StorageSynchronized associations, "
   "one for each pair of members in the source and target group, "
   "are returned in Synchronizations. GroupSynchronization "
   "asociation is not supported." ),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
          "4097..32767", "0x8000.." },
       Values { "Completed with No Error", "Not Supported",
          "Unknown", "Timeout", "Failed", "Invalid Parameter",
          "In Use", "DMTF Reserved",
          "Method Parameters Checked - Job Started",
```

```
        "Method Reserved", "Vendor Specific" }]
   uint32 TPD_CreateGroupReplica(
         [IN, Description (
             "A user relevant name for the relationship between "
             "the source and target groups or between a source "
             "element and a target group (i.e. one-to-many). If "
             "NULL, the implementation assigns a name. If the "
             "individual target elements require an ElementName, "
             "the implementation constructs an appropriate "
             "ElementName using the RelationshipName. For "
             "example, RelationshipName as a prefix followed by "
             "\"_n\" sequence number, where n is a number "
             "beginning with 1." )]
      string RelationshipName,
         [Required, IN, Description (
             "SyncType describes the type of copy that will be made."
              ),
          ValueMap { "..", "6", "7", "8", "..", "0x8000.." },
          Values { "DMTF Reserved", "Mirror", "Snapshot",
             "Clone", "DMTF Reserved", "Vendor Specific" },
          ModelCorrespondence { "CIM_Synchronized.SyncType" }]
      uint16 SyncType,
         [IN, Description (
             "Mode describes whether the target elements will be "
             "updated synchronously or asynchronously. If NULL, "
             "implementaton decides the mode." ),
          ValueMap { "2", "3", "..", "0x8000.." },
          Values { "Synchronous", "Asynchronous",
             "DMTF Reserved", "Vendor Specific" },
          ModelCorrespondence { "CIM_Synchronized.Mode" }]
      uint16 Mode,
         [IN, Description (
             "A group of source storage objects of StorageVolume." )]
      CIM_ReplicationGroup REF SourceGroup,
         [IN, Description (
             "As an input, refers to a target group "
             "to use. This is valid only if SyncType is "
      "Asynchronous, i.e., physical copy")]
      CIM_ReplicationGroup REF TargetGroup,
         [IN ( false), OUT, Description (
             "Array of references to the created target volumes, matched with "
      "with SourceElements, valid only for Snapshot/Synchronous "
      "replicas (virtual copy).")]
      TPD_StorageVolume REF TargetElements[],
         [IN ( false ), OUT, Description (
             "Reference to the list of jobs (may be NULL if job is completed)."
              )]
      CIM_ConcreteJob REF Jobs[],
         [IN ( false ), OUT, Description (
             "References to the created StorageSynchronized associations between "
             "the source and the target elements. If a job is "
             "created, this parameter may be NULL until the "
             "association is actually formed." )]
      TPD_StorageSynchronized REF Synchronizations[]);

      [Description (
          "Modify all StorageSynchronized in a ReplicationGroup. Output "
   "Jobs, if present, is an array consisting of ConcreteJobs "
   "of each volume pair between the source and target "
   "ReplicationGroup."),
       ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
          "4097..32767", "0x8000.." },
       Values { "Completed with No Error", "Not Supported",
          "Unspecified Error", "Timeout", "Failed",
          "Invalid Parameter", "In Use", "DMTF Reserved",
          "Method Parameters Checked - Job Started",
```

```
        "Method Reserved", "Vendor Specific" }]
  uint32 TPD_ModifyGroupReplicaSynchronization(
        [Required, IN, Description (
            "Operation describes the type of modification to be "
            "made to the ReplicationGroup and/or to the related "
            "associations. \n"
            "Abort: Abort the copy operation if it is possible. "
            "Activate Consistency: if consistency was not "
            "requested when CreateGroupReplica was called. If "
            "Consistency is already active, no modification is "
            "made. \n"
            "Activate: Activate an inactive or prepared "
            "Synchronized association. \n"
            "AddSyncPair: Add pairs of elements already in a "
            "relationship to source and target groups -- see "
            "SyncPair parameter. \n"
            "Deactivate Consistency: Deactivate consistency. If "
            "consistency was not enabled, this operation has no "
            "effect. \n"
            "Deactivate: Stop the data flow. Writes to source "
            "element are not copied to target. For Snapshots, "
            "writes to target are lost as the pointers to "
            "changed data are deleted. \n"
            "Detach: \'Forget\' the synchronization between two "
            "storage objects. Start to treat the objects as "
            "independent. \n"
            "Dissolve: Dissolve the synchronization between two "
            "storage objects, however, the target element "
            "continues to exist. \n"
            "Failover: Use the target element as the source "
            "elements. \n"
            "Failback: Reverse the effect of failback. \n"
            "Fracture: Suspend the synchronization between two "
            "storage objects. The association and (typically) "
            "changes are remembered to allow a fast "
            "resynchronization. This may be used during a "
            "backup cycle to allow one of the objects to be "
            "copied while the other remains in production. \n"
            "RemoveSyncPair: Remove the pair associated via "
            "StorageSynchronized from the source and target "
            "groups. The pair continue to remain associated but "
            "not in the groups. \n"
            "Resync Replica: Re-establish the synchronization. "
            "This will negate the action of a previous "
            "Fracture/Split operation. Recreate a Point In Time "
            "image for a Snapshot or a Clone replication. "
            "Restart a Broken or Aborted synchronization "
            "relationship. \n"
            "Restore from Replica: Renew the contents of the "
            "original storage object from a replica. \n"
            "Resume: Continue the copy operation of a suspended "
            "association. \n"
            "Reset To Sync: Change the Mode of the copy "
            "operation to Synchronous (e.g., from the "
            "Asynchronous Mode). \n"
            "Reset To Async: Change the Mode of the copy "
            "operation to Asynchronous (e.g., from the "
            "Synchronous Mode). \n"
            "Return to ResourcePool: Dissolve a snapshot and "
            "free up its space back to the storage pool. \n"
            "Reverse Roles: Source element becomes the target "
            "element and vise versa. \n"
            "Split: Same as Fracture, however steps are taken "
            "to ensure the target elements are consistent. For "
            "example, stop I/O to source elements, wait for "
            "in-transit copy operations between source and "
```

```
            "target elements to stop, then instantly split "
            "source/target groups/elements. Suspend: Stop the "
            "background copy previously started. \n"
            "Unprepare: Causes the synchronization to be "
            "reinitialized." ),
         ValueMap { "2", "3", "4", "5", "6", "7", "8", "9",
            "10", "11", "12", "13", "14", "15", "16", "17",
            "18", "19", "20", "21", "22", "23", "..",
            "0x8000..0xFFFF" },
         Values { "Abort", "Activate Consistency", "Activate",
            "AddSyncPair", "Deactivate Consistency",
            "Deactivate", "Detach", "Dissolve", "Failover",
            "Failback", "Fracture", "RemoveSyncPair",
            "Resync Replica", "Restore from Replica", "Resume",
            "Reset To Sync", "Reset To Async",
            "Return To ResourcePool", "Reverse Roles", "Split",
            "Suspend", "Unprepare", "DMTF Reserved",
            "Vendor Specific" }]
   uint16 Operation,
      [Required, IN, Description (
          "The reference to the target ReplicationGroup." )]
   TPD_ReplicationGroup REF TargetGroup,
      [IN ( false ), OUT, Description (
          "Array of references to the jobs (may be NULL if the task completed)."
           )]
   CIM_ConcreteJob REF Jobs[]);

   [Description (
       "Create consistent group physical copies or snapshots of "
"a list of virtual volumes." ),
    ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
       "4097..32767", "0x8000.." },
    Values { "Completed with No Error", "Not Supported",
       "Unknown", "Timeout", "Failed", "Invalid Parameter",
       "In Use", "DMTF Reserved",
       "Method Parameters Checked - Job Started",
       "Method Reserved", "Vendor Specific" }]
uint32 TPD_CreateConsistentReplicaList(
      [Required, IN, Description (
          "SyncType describes the type of copy that will be made."
           ),
       ValueMap { "..", "6", "7", "8", "..", "0x8000.." },
       Values { "DMTF Reserved", "Mirror", "Snapshot",
          "Clone", "DMTF Reserved", "Vendor Specific" },
       ModelCorrespondence { "CIM_Synchronized.SyncType" }]
   uint16 SyncType,
      [IN, Description (
          "Mode describes whether the target elements will be "
          "updated synchronously or asynchronously. If NULL, "
          "implementaton decides the mode." ),
       ValueMap { "2", "3", "..", "0x8000.." },
       Values { "Synchronous", "Asynchronous",
          "DMTF Reserved", "Vendor Specific" },
       ModelCorrespondence { "CIM_Synchronized.Mode" }]
   uint16 Mode,
      [IN, Description (
          "Array of references to source volumes to replicate. "
   "Number of elements in this array must be the same "
   "as that of TargetNames and TargetElements.")]
   TPD_StorageVolume REF SourceElements[],
      [IN, Description (
          "Array of names, matched with SourceElements array, "
   "of the resulting target snapshot volumes. This "
   "parameter is valid only for Snapshot/Synchronous "
   "replicas (virtual copy). This and TargetElements cannot both "
   "be specified.")]
```

```
      String TargetNames[],
         [IN, OUT, Description (
             "Array of references to target volumes, matched with "
      "with SourceElements. As an input, this parameter is valid only "
      "for Snapshot/Async or Clone/Async replicas (physical "
      "copy), and this and TargetNames cannot both be specified. "
      "As an output, this contains the references to created StorageVolumes "
      "valid only for Snapshot/Synchronous replicas (virtual copy).")]
      TPD_StorageVolume REF TargetElements[],
         [IN ( false ), OUT, Description (
             "Reference to the list of jobs (may be NULL if job is completed)."
              )]
      TPD_ConcreteJob REF Jobs[],
         [IN, Description (
             "The definition for the SettingData to be "
             "maintained by the target storage object (the "
             "replica). If a target element is supplied, this "
             "parameter shall be NULL." )]
      TPD_StorageSetting REF TargetSettingGoal,
         [IN, Description (
             "Method must wait until this CopyState is reached "
             "before returning. Only a subset of valid "
             "CopyStates apply. For example, Initialized: "
             "Associations have been established, but there is "
             "no data flow. Inactive: Initialization is "
             "complete, but the data flow remains idle until it "
             "is activated. Synchronized: Replicas are an exact "
             "copy of the source. UnSynchronized: Copy operation "
             "is in progress." ),
          ModelCorrespondence { "CIM_Synchronized.CopyState" }]
      uint16 WaitForCopyState,
         [IN ( false ), OUT, Description (
             "References to the created StorageSynchronized associations between "
             "the source and the target elements. If a job is "
             "created, this parameter may be NULL until the "
             "association is actually formed." )]
      TPD_StorageSynchronized REF Synchronizations[]);
};

// ===================================================================
// 3PAR Replication Service Capabilities
// ===================================================================
[Description (
       "A subclass of Capabilities that defines the Capabilities of a "
       "ReplicationService. An instance of "
       "ReplicationServiceCapabilities is associated with a "
       "ReplicationService using ElementCapabilities." )]
class TPD_ReplicationServiceCapabilities : CIM_ReplicationServiceCapabilities
{
};

// ===================================================================
// 3PAR CapabilitiesOfReplicationService
// ===================================================================
[Association,
    Description (
        "TPD_CapabilitiesOfReplicationService is an association "
        "between ReplicationService and ReplicationServiceCapabilities "
        "This association is used to query the features and functions "
        "supported by ReplicationService.") ]
class TPD_CapabilitiesOfReplicationService: CIM_ElementCapabilities
{
    [Override ( "ManagedElement" ), Description (
          "The ReplicationService.")]
    TPD_ReplicationService REF ManagedElement;
```

```
    [Override ( "Capabilities" ), Description (
          "The configuration capabilities that the "
   "ReplicationService can support. ")]
    TPD_ReplicationServiceCapabilities REF Capabilities;
};

// ===================================================================
// 3PAR Replication Group
// ===================================================================
[Description (
       "This class represents a collection of storage objects, such as "
       "a group of StorageVolumes. It is used to represent volume set." )]
class TPD_ReplicationGroup : CIM_ReplicationGroup
{
      [Description (
          "System assigned ID for the volume set.")]
   uint32 ID;
};

// ===================================================================
// 3PAR System and ReplicationService association.
// ===================================================================
[Association,
    Description (
       "TPD_HostedReplicationService is an association "
       "between TPD_StorageSystem and TPD_HostedReplicationService. "
       "The cardinality of this association is 1-to-1. A System may "
       "host only one TPD_HostedReplicationService. ")]
class TPD_HostedReplicationService: CIM_HostedService
{
      [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
     "The InServ StorageSystem.")]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ), Weak, Description (
          "The ReplicationService hosted on the 3PAR InServ.")]
   TPD_ReplicationService REF Dependent;
};

// ===================================================================
// MemberOfReplicationGroup
// ===================================================================
   [Association, Aggregation, Description (
       "MemberOfReplicationGroup is an aggregation used "
       "to establish membership of StorageVolume in a ReplicationGroup, "
       "i.e., volume set.")]
class TPD_MemberOfReplicationGroup : CIM_MemberOfCollection {

      [Key, Aggregate, Description (
          "The MemberOfReplicationGroup that aggregates StorageVolume .")]
   TPD_ReplicationGroup REF Collection;

      [Key, Description (
          "The aggregated member of the "
     "MemberOfReplicationGroup.")]
   TPD_StorageVolume REF Member;
};

// ===================================================================
// 3PAR ReplicationAffectsElement
// ===================================================================
[Association,
    Description (
        "TPD_ReplicationAffectsElement is an association "
        "between ReplicationService and ReplicationGroup "
        "This provider is used to query the association "
```

```
        "between affecting and affected sets.") ]
class TPD_ReplicationAffectsElement : CIM_ServiceAffectsElement
{
    [Override ( "AffectingElement" ),
     Key, Description ("The ReplicationService.")]
    TPD_ReplicationService REF AffectingElement;

    [Override ( "AffectedElement" ),
     Key, Description (
          "The ReplicationGroup that the ReplicationService "
          "may affect.")]
    TPD_ReplicationGroup REF AffectedElement;
};

// Remote Replication related classes

// ===================================================================
// 3PAR RemoteStorageSynchronized
// ===================================================================
[Association,
   Description (
       "Indicates that two Storage objects were remote replicated at the "
       "specified point in time. If the UndiscoveredElement indicates "
       "either SystemElement or SyncedElement, then the corresponding "
       "property will contain the reference to TPD_ReplicationEntity, "
       "otherwise it will contain the reference to a TPD_StorageVolume.")]
class TPD_RemoteStorageSynchronized : CIM_StorageSynchronized {
      [Description (
          "True if role of the SystemElement and SyncedElement has been switched "
        "due to a fail over.")]
   boolean RoleReversed;

      [Description (
          "Specifies that groups that are in asynchronous periodic mode should "
        "be periodically synchronized in accordance with the specified value "
        "of the replication. 0 means synchronization will not happen unless "
        "it is done manually." )]
   datetime SyncInterval;

      [Description (
         "Specifies the policy of the Remote Copy volume groups for dealing with I/O
 "
        "failure and error handling of the replication. Valid values are:"
        "    no_fail_wrt_on_err "
        "        Specifies that if Remote Copy is started for the volume group "
        "        and a write to the secondary system fails, then the Remote Copy "
        "        operation is stopped and an I/O error is not returned to the "
        "        host (default). This allows the application writing the data to "
        "        continue, but makes the secondary volumes out of date with the "
        "        primary volumes. "
        "    auto_recover                                                       "
        "         Specifies that if the Remote Copy is stopped as a result of the "
        "         Remote Copy links going down, the group is restarted          "
        "         automatically after the links come back up. If this policy is   "
        "         enabled for a group while the group is stopped after link       "
        "         failures, it will be only be started when the links come up for "
        "         the failed target.  If the links are already up at the time the "
        "         policy is set then the group will not be restarted at that      "
        "         time.                    "
        "     no_auto_recover "
        "         Specifies that if the Remote Copy is stopped as a result of the  "
        "         Remote Copy links going down, the group must be restarted  "
        "         manually after the links come back up (default). "
        "     over_per_alert "
        "         If a synchronization of a periodic Remote Copy group takes  "
        "         longer to complete than its synchronization period then an  "
```

```
        "         alert will be generated. This is the default behavior. "
        "     no_over_per_alert "
        "         If a synchronization of a periodic Remote Copy group takes  "
        "         longer to complete than its synchronization period then an  "
        "         alert will not be generated."),
       BitMap { "1","2","3","4","5"},
       BitValues  { "no_fail_wrt_on_err", "auto_recover", "no_auto_recover",
          "over_per_alert", "no_over_per_alert" }]
   uint32 SyncPolicy;
};

// ===================================================================
// 3PAR Remote Replication Group
// ===================================================================
[Description (
       "This class represents a collection of StorageVolumes "
       "used for remote replication.")]
class TPD_RemoteReplicationGroup : CIM_ReplicationGroup
{
   [Description (
       "Name of the administrative domain that this group belongs "
       "to.\n")]
    String Domain;
};

// ===================================================================
//  TPD_OrderedMemberOfRemoteReplicationGroup
// ===================================================================
[Association,
    Description (
       "TPD_OrderedMemberOfRemoteReplicationGroup is an aggregation used to "
       "establish an ordered membership of StorageVolume in a "
       "RemoteReplicationGroup." )]
class TPD_OrderedMemberOfRemoteReplicationGroup : CIM_OrderedMemberOfCollection {
      [Override ("Collection"),
       Key, Aggregate, Description (
          "The RemoteReplicationGroup that aggregates StorageVolumes." )]
   TPD_RemoteReplicationGroup REF Collection;

      [Override ("Member"),
      Key, Description ( "The aggregated StorageVolumes of the RemoteReplicationGroup."
 )]
   TPD_StorageVolume REF Member;
};

// ===================================================================
// 3PAR RemoteGroupSynchronized
// ===================================================================
[Association,
    Description (
       "Indicates that two remote replication groups are associated." )]
class TPD_RemoteGroupSynchronized : CIM_GroupSynchronized {
      [Override ( "SystemElement" ),
       Description (
          "SystemElement represents the group that is the source of "
          "the replication." )]
   TPD_RemoteReplicationGroup REF SystemElement;

      [Override ( "SyncedElement" ),
       Description (
          "SyncedElement represents the target that is the target "
          "of the replication." )]
   TPD_RemoteReplicationGroup REF SyncedElement;

      [Description (
          "True if role of the SystemElement and SyncedElement has been switched "
```

```
        "due to a fail over.")]
   boolean RoleReversed;

      [Description (
          "Specifies that groups that are in asynchronous periodic mode should "
        "be periodically synchronized in accordance with the specified value "
        "of the replication. 0 means synchronization will not happen unless "
        "it is done manually." )]
   datetime SyncInterval;

      [Description (
         "Specifies the policy of the Remote Copy volume groups for dealing with I/O
 "
        "failure and error handling of the replication. Valid values are:"
        "    no_fail_wrt_on_err "
        "        Specifies that if Remote Copy is started for the volume group "
        "        and a write to the secondary system fails, then the Remote Copy "
        "        operation is stopped and an I/O error is not returned to the "
        "        host (default). This allows the application writing the data to "
        "        continue, but makes the secondary volumes out of date with the "
        "        primary volumes. "
        "    auto_recover                                                        "
        "         Specifies that if the Remote Copy is stopped as a result of the "
        "         Remote Copy links going down, the group is restarted           "
        "         automatically after the links come back up. If this policy is   "
        "         enabled for a group while the group is stopped after link       "
        "         failures, it will be only be started when the links come up for "
        "         the failed target.  If the links are already up at the time the "
        "         policy is set then the group will not be restarted at that      "
        "         time.                    "
        "     no_auto_recover "
        "         Specifies that if the Remote Copy is stopped as a result of the  "
        "         Remote Copy links going down, the group must be restarted  "
        "         manually after the links come back up (default). "
        "     over_per_alert "
        "         If a synchronization of a periodic Remote Copy group takes  "
        "         longer to complete than its synchronization period then an  "
        "         alert will be generated. This is the default behavior. "
        "     no_over_per_alert "
        "         If a synchronization of a periodic Remote Copy group takes  "
        "         longer to complete than its synchronization period then an  "
        "         alert will not be generated."),
       BitMap { "1","2","3","4","5"},
       BitValues  { "no_fail_wrt_on_err", "auto_recover", "no_auto_recover",
          "over_per_alert", "no_over_per_alert" }]
   uint32 SyncPolicy;
};

// ===================================================================
// 3PAR ReplicationServiceAffectsRemoteReplicationGroup
// ===================================================================
[Association,
    Description (
        "TPD_ReplicationServiceAffectsRemoteReplicationGroup is an association "
        "between ReplicationService and RemoteReplicationGroup "
        "This provider is used to query the association "
        "between affecting and affected sets.") ]
class TPD_ReplicationServiceAffectsRemoteReplicationGroup : CIM_ServiceAffectsElement
{
    [Override ( "AffectingElement" ),
     Key, Description ("The ReplicationService.")]
    TPD_ReplicationService REF AffectingElement;

    [Override ( "AffectedElement" ),
     Key, Description (
          "The RemoteReplicationGroup that the ReplicationService "
```

```
          "may affect.")]
    TPD_RemoteReplicationGroup REF AffectedElement;
};

// ===================================================================
//  TPD_HostedRemoteReplicationGroup
// ===================================================================
[Association,
    Description (
       "HostedCollection defines a RemoteReplicationGroup in the "
       "context of StorageSystem. It represents a Collection that "
       "has meaning only in the context of a System, a Collection "
       "whose elements are restricted by the definition of the System, "
       "or both of these types of Collections." )]
class TPD_HostedRemoteReplicationGroup : CIM_HostedCollection {

      [Override ( "Antecedent" ),
       Min ( 1 ),
       Max ( 1 ),
       Description ( "The scoping system." )]
   TPD_StorageSystem REF Antecedent;

      [Override ( "Dependent" ),
       Description (
          "The collection defined in the context of a system." )]
   TPD_RemoteReplicationGroup REF Dependent;
};

// ===================================================================
//  TPD_SAPAvailableForRemoteReplicaVolume
// ===================================================================
[Association,
    Description (
       "Association between a StorageVolume in a remote copy group "
       "and the port used for this remote copy group.")]
class TPD_SAPAvailableForRemoteReplicaVolume : CIM_SAPAvailableForElement {

      [Key, Description (
          "The Service Access Point that is available. "
          "For RCFC, the endpoint is a TPD_SCSIProtocolFCEndpoint. "
          "For RCIP, the endpoint is a TPD_IPProtocolEndpoint." )]
   CIM_ProtocolEndpoint REF AvailableSAP;

      [Key, Description (
          "The StorageVolume for which the SAP is available." )]
   TPD_StorageVolume REF ManagedElement;
};

// ===================================================================
//  TPD_SAPAvailableForRemoteReplicationGroup
// ===================================================================
[Association,
    Description (
       "Association between a remote copy group "
       "and the port used for this remote copy group.")]
class TPD_SAPAvailableForRemoteReplicationGroup : CIM_SAPAvailableForElement {

      [Key, Description (
          "The Service Access Point that is available. "
          "For RCFC, the endpoint is a TPD_SCSIProtocolFCEndpoint. "
          "For RCIP, the endpoint is a TPD_IPProtocolEndpoint." )]
   CIM_ProtocolEndpoint REF AvailableSAP;

      [Key, Description (
          "The RemoteReplicationGroup for which the SAP is available." )]
   TPD_RemoteReplicationGroup REF ManagedElement;
```

```
};

// ===================================================================
//  TPD_ReplicationEntity
// ===================================================================
   [Description (
       "This class represents a replication entity, such as an entity "
       "known by its World Wide Name (WWN). "
       "If the remote StorageVolume in a remote replication relationship "
       "cannot be discovered, then this will be represented in the "
       "RemoteStorageSynchronized association as either SystemElement or "
       "SyncedElement property. This will not exist as an instance "
       "on its own." )]
class TPD_ReplicationEntity : CIM_ReplicationEntity {
};
```

3PAR_TPDStats.mof

```
//%////////////////////////////////////////////////////////////////////////
//
//
//  Copyright 2007 3PAR, Inc. All Rights Reserved.
//  This software is the property of 3PAR, Inc. Distribution
//  or deployment of the source or derived binaries of this
//  software are not permitted.
//
//
//%////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
//  File    : 3PAR_TPDStats.mof
//
//  Purpose : This MOF contains 3PAR Statistical (Block Server Performance
//            Subprofile) classes that will be loaded into root/tpd namespace.
//
//  Date created: 4/18/2006
//
////////////////////////////////////////////////////////////////////////////

// ===================================================================
// Statistics Collection
// ===================================================================
[Description (
   "Statistics collection")]
class TPD_StatisticsCollection : CIM_StatisticsCollection
{
};

// ===================================================================
//  Hosted Statistics Collection
// ===================================================================
[Association, Description (
   "HostedCollection defines a SystemSpecificCollection in the "
   "context of a scoping System. It represents a Collection that "
   "has meaning only in the context of a System, a Collection "
   "whose elements are restricted by the definition of the System, "
   "or both of these types of Collections.")]
class TPD_HostedStatisticsCollection : CIM_HostedCollection
{
   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
       "The scoping system.")]
   TPD_StorageSystem REF Antecedent;
```

```
   [Override ( "Dependent" ), Description (
       "The collection defined in the context of a system.")]
   TPD_StatisticsCollection REF Dependent;
};

// ===================================================================
// Block Statistics Statistics Collection
// ===================================================================
[Description (
   "Block Statistics Statistics collection")]
class TPD_BlockStatisticsManifestCollection :
                                  CIM_BlockStatisticsManifestCollection
{
};

// ===================================================================
//  Associated Block Statistics Manifest Collection
// ===================================================================
[Association, Description (
   "Instances of this class associate a "
   "BlockStatisticsManifestCollection to the StatisticsCollection "
   "to which is is applied. The ManifestCollection contains the "
   "Manifests that are used to filter requests for the retrieval "
   "of statistics.")]
class TPD_AssociatedBlockStatisticsManifestCollection :
                     CIM_AssociatedBlockStatisticsManifestCollection
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
       "The collection of statistics filtered by the "
       "BlockStatisticsManifestCollection.")]
   TPD_StatisticsCollection REF Statistics;

   [Key, Description (
       "The collection of Manifests applied to the "
       "StatisticsCollection.")]
   TPD_BlockStatisticsManifestCollection REF ManifestCollection;
};

// ===================================================================
// Block Statistics Manifest
// ===================================================================
[Description (
   "Block Statistics manifest")]
class TPD_BlockStatisticsManifest : CIM_BlockStatisticsManifest
{
};

// ===================================================================
//  Member Of Block Statistics Manifest Collection
// ===================================================================
[Association, Aggregation, Description (
   "Aggregation used to establish "
   "membership of ManagedElements in a Collection.")]
class TPD_MemberOfBlockStatisticsManifestCollection : CIM_MemberOfCollection {

   [Key, Aggregate, Description (
       "The Collection that aggregates members.")]
   TPD_BlockStatisticsManifestCollection REF Collection;

   [Key, Description (
       "The aggregated member of the Collection.")]
   TPD_BlockStatisticsManifest REF Member;
};
// ===================================================================
//  Hosted Block Statistics Manifest Collection
// ===================================================================
```

```
[Association, Description (
   "HostedCollection defines a SystemSpecificCollection in the "
   "context of a scoping System. It represents a Collection that "
   "has meaning only in the context of a System, a Collection "
   "whose elements are restricted by the definition of the System, "
   "or both of these types of Collections.")]
class TPD_HostedStatisticsManifestCollection : CIM_HostedCollection
{
   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
       "The scoping system.")]
   TPD_StorageSystem REF Antecedent;

   [Override ( "Dependent" ), Description (
       "The collection defined in the context of a system.")]
   TPD_BlockStatisticsManifestCollection REF Dependent;
};

// ===================================================================
//  Member Of Statistics Collection
// ===================================================================
[Association, Aggregation, Description (
   "Aggregation used to establish "
   "membership of ManagedElements in a Collection.")]
class TPD_MemberOfStatisticsCollection : CIM_MemberOfCollection {

   [Key, Aggregate, Description (
       "The Collection that aggregates members.")]
   TPD_StatisticsCollection REF Collection;

   [Key, Description (
       "The aggregated member of the Collection.")]
   CIM_BlockStorageStatisticalData REF Member;
};

// ===================================================================
// Array Statistical Data
// ===================================================================
[Description (
   "Statistical data for the Array")]
class TPD_ArrayStatisticalData : CIM_BlockStorageStatisticalData
{
};

// ===================================================================
//  Array Element Statistical Data
// ===================================================================
[Association, Description (
   "An association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_ArrayElementStatisticalData : CIM_ElementStatisticalData
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
      "The ManagedElement for which statistical or metric data is "
      "defined.")]
   TPD_StorageSystem REF ManagedElement;

   [Key, Description (
      "The statistic information/object.")]
   TPD_ArrayStatisticalData REF Stats;
};
```

```
// ===================================================================
// Node Statistical Data
// ===================================================================
[Description (
   "Statistical data for a node controller")]
class TPD_NodeStatisticalData : CIM_BlockStorageStatisticalData
{
};

// ===================================================================
//  Node Element Statistical Data
// ===================================================================
[Association, Description (
   "An association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_NodeElementStatisticalData : CIM_ElementStatisticalData
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
      "The ManagedElement for which statistical or metric data is "
      "defined.")]
   TPD_NodeSystem REF ManagedElement;

   [Key, Description (
      "The statistic information/object.")]
   TPD_NodeStatisticalData REF Stats;
};

// ===================================================================
// Disk Drive Statistical Data
// ===================================================================
[Description (
   "Statistical data for the Disk Drive")]
class TPD_DiskStatisticalData : CIM_BlockStorageStatisticalData
{
};

// ===================================================================
//  Disk Element Statistical Data
// ===================================================================
[Association, Description (
   "An association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_DiskElementStatisticalData : CIM_ElementStatisticalData
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
      "The ManagedElement for which statistical or metric data is "
      "defined.")]
   TPD_DiskStorageExtent REF ManagedElement;

   [Key, Description (
      "The statistic information/object.")]
   TPD_DiskStatisticalData REF Stats;
};
```

```
// ===================================================================
// Port Statistical Data
// ===================================================================
[Description (
   "Statistical data for the Port")]
class TPD_PortStatisticalData : CIM_BlockStorageStatisticalData
{
};

// ===================================================================
//  Port Element Statistical Data
// ===================================================================
[Association, Description (
   "An association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_PortElementStatisticalData : CIM_ElementStatisticalData
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
      "The ManagedElement for which statistical or metric data is "
      "defined.")]
   CIM_NetworkPort REF ManagedElement;

   [Key, Description (
      "The statistic information/object.")]
   TPD_PortStatisticalData REF Stats;
};

// ===================================================================
// Volume Statistical Data
// ===================================================================
[Description (
   "Statistical data for the Volume")]
class TPD_VolumeStatisticalData : CIM_BlockStorageStatisticalData
{
};

// ===================================================================
//  Storage Volume Element Statistical Data
// ===================================================================
[Association, Description (
   "An association that relates a "
   "ManagedElement to its StatisticalData. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementStatisticalData association for the referenced instance "
   "of CIM_StatisticalData. ElementStatisticalData describes the "
   "existence requirements and context for the "
   "CIM_StatisticalData, relative to a specific ManagedElement.")]
class TPD_VolumeElementStatisticalData : CIM_ElementStatisticalData
{
   [Key, Min ( 1 ), Max ( 1 ), Description (
      "The ManagedElement for which statistical or metric data is "
      "defined.")]
   TPD_StorageVolume REF ManagedElement;

   [Key, Description (
      "The statistic information/object.")]
   TPD_VolumeStatisticalData REF Stats;
};
```

```
// ===================================================================
//  Block Statistics Service
// ===================================================================
class TPD_BlockStatisticsService : CIM_BlockStatisticsService
{
};

// ===================================================================
//  Hosted Statistics Service
// ===================================================================
[Association, Description (
   "An association between a Service and the "
   "System on which the functionality is located. The cardinality "
   "of this association is one-to-many. A System can host many "
   "Services. Services are weak with respect to their hosting "
   "System. Heuristic: A Service is hosted on the System where the "
   "LogicalDevices or SoftwareFeatures that implement the Service "
   "are located. The model does not represent Services hosted "
   "across multiple systems. The model is as an ApplicationSystem "
   "that acts as an aggregation point for Services that are each "
   "located on a single host.")]
class TPD_HostedStatisticsService : CIM_HostedService
{
   [Override ( "Antecedent" ), Min ( 1 ), Max ( 1 ), Description (
       "The scoping system.")]
   TPD_StorageSystem REF Antecedent;

   [Override ( "Dependent" ), Description (
       "The collection defined in the context of a system.")]
   TPD_BlockStatisticsService REF Dependent;
};

// ===================================================================
//  Block Statistics Capabilities
// ===================================================================
class TPD_BlockStatisticsCapabilities : CIM_BlockStatisticsCapabilities
{
};

// ===================================================================
//  Statistics Element Capabilities
// ===================================================================
[Association, Description (
   "ElementCapabilities represents the association between "
   "ManagedElements and their Capabilities. Note that the "
   "cardinality of the ManagedElement reference is Min(1), Max(1). "
   "This cardinality mandates the instantiation of the "
   "ElementCapabilities association for the referenced instance of "
   "Capabilities. ElementCapabilities describes the existence "
   "requirements and context for the referenced instance of "
   "ManagedElement. Specifically, the ManagedElement MUST exist "
   "and provides the context for the Capabilities.")]
class TPD_StatisticsElementCapabilities : CIM_ElementCapabilities {

      [Key, Min ( 1 ), Max ( 1 ), Description (
          "The managed element.")]
   TPD_BlockStatisticsService REF ManagedElement;

      [Key, Description (
          "The Capabilities object associated with the element.")]
   TPD_BlockStatisticsCapabilities REF Capabilities;
};

// ===================================================================
// LUNs Statistical Data
```

```
// ===================================================================
[Description (
   "Statistical data for the LUNs")]
class TPD_LUNStatisticalData : CIM_BlockStorageStatisticalData
{
};
```