Site Recovery Manager API Developer's Guide

Site Recovery Manager 8.2

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About Site Recovery Manager API Developer's Guide

1

The Site Recovery Manager API Developer's Guide provides information about programming applications with the Web services interfaces to VMware Site Recovery Manager.

This manual provides information about interfaces in the Site Recovery Manager 5.0, 5.8, 6.0, 6.1, 6.5, 8.1, and 8.2 releases, for developers who are interested in automating site recovery tasks.

Intended Audience

This book is intended for developers who want to set up their environment to program applications with the Site Recovery Manager API. These developers are typically programmers using the Java or C# language and libraries to perform replication, recovery, and reprotection of virtual machines in VMware vSphere.

Site Recovery Manager developers should have some familiarity with the Web Services Description Language (WSDL) and the Simple Object Access Protocol (SOAP) for transmitting XML across the network. However, the important interfaces are completely visible in Java or C# code.

VMware Developer Publications

To view the current version of this book and other VMware API and SDK public documentation, go to http://www.vmware.com/support/developer.

Visit https://www.vmware.com/support/pubs/srm_pubs.html for information about this version of Site Recovery Manager.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

The Site Recovery Manager API 2

This manual describes a Web services programming interface to protection groups and recovery plans in VMware Site Recovery Manager.

This chapter includes the following topics:

Introduction to the API

The Site Recovery Manager API provides an interface similar to vSphere API, an object-oriented Web service that provides access to vSphere and virtual machine management on vCenter Server and ESXi hosts.

You can program the vSphere API in Java, C#, or any language that supports Web services definition language (WSDL).

The Site Recovery Manager API offers a way for third-party systems to create protection groups and initiate test, recovery, and reprotect operations and collect the results.

API Releases

The Site Recovery Manager 5.0 release extended the API with new methods to list and modify protection groups, and revised methods to list, modify, and run recovery plans.

The Site Recovery Manager 5.8 release introduced 30 methods and 4 new managed objects, adding several requested features to the API :

- Ability to add folder, network, and resource pool mappings
- Support for planned migrations

- Navigation capabilities for protection group folders and recovery plan folders
- Ability to create protection groups, and to modify selected fields in a virtual machine's recovery settings

The Site Recovery Manager 6.0 release introduced new methods to support authentication by tokens:

- New concept of the Site Recovery Manager solution user for vCenter Single Sign-On authentication
- Functions to get the Site Recovery Manager solution user name for the local and remote sites
- Functions to log in to local or remote sites using security assertion markup language (SAML) token
- Ability to use lookup service URL and vCenter Server instance ID
- Although `DisasterRecoveryApi` is deprecated, it gains forward compatibility with `LoginByToken`

In release 6.1, VMware Site Recovery Manager introduced Storage Profile Protection Groups (SPPG). However, the Site Recovery Manager API does not support SPPG related objects.

Site Recovery Manager API 6.5 introduced methods that expands the operations on the inventory mappings, recovery plans, and protection groups.

The Site Recovery Manager 8.1 release introduces new methods that provides ability to:

- Add replicated datastores (that are newly provisioned) to an existing protection group.
- Remove datastores from the protection group.
- Configure the IP address and corresponding DNS, WINS of the virtual machine, after the migration is complete.

The Site Recovery Manager 8.2 release introduces new methods that provides ability to:

- Read information specific to an ArrayManager instance and list all the array pairs in the array manager.
- Get information about all the replicated RDMs in a replicated array pair.
- Get a list all the available array managers.

Terminology

This document uses the following terms.

SOAP

Client applications invoke operations by sending SOAP formatted messages. When passing data objects between client and server, programs build or parse XML messages representing data structures described by the WSDL. Standardized by W3C as Simple Object Access Protocol (SOAP) 1.1.

Web service operations

Client interfaces that perform server-side management and monitoring tasks. Standardized as Web Services Interoperability Organization (WS-I) Basic Profile 1.0.

WSDL

The Web services API is defined in a WSDL file, which is used by client-side Web services to create proxy code (stubs) that client applications use to interact with the server. Standardized as Web Services Description Language (WSDL) 1.1.

XML

A text representation scheme similar to HTML but with more stringent, regularized syntax. Standardized by W3C as Extensible Markup Language (XML) 1.0.

The Site Recovery Manager API is similar to, and derived from the vSphere API. For information about the vSphere API, see the *vSphere Web Services SDK Programming Guide* and the *vSphere API Reference* at the VMware Web site.

List of API Operations

By using Site Recovery Manager API you can perform a set of operations such as creating Site Recovery Manager objects and retrieving information about the objects.

The following tables provide a list of Site Recovery Manager API methods organized by approximate order of use.

Table 2-1. Solution User Information

Method	Description of Operation
`GetSolutionUserInfo`	Obtain the Site Recovery Manager solution user name for the local site.
`GetPairedSiteSolutionUserInfo`	Obtain the Site Recovery Manager solution user name for the remote site.

Table 2-2. SAML Token Authentication

Method	Description of Operation
`SrmLoginByTokenLocale`	Begin a session with Site Recovery Manager Server.
`SrmLoginSitesByToken`	Log in to both the local and remote vCenter Server.
`SrmLoginRemoteSiteByToken`	Log in to remote site when escalated privileges are required and the current session has already been authenticated using `SrmLoginSitesByToken`.

Table 2-3. Credential-Based Authentication

Method	Description of Operation
`SrmLoginLocale`	Begin a session with Site Recovery Manager Server.
`SrmLoginSites`	Log in to both the local and remote vCenter Server.
`SrmLogoutLocale`	Log out sites and terminate the current session.
`SrmLoginRemoteSite`	Log in to remote site when escalated privileges are required on the remote site and the current session has already been authenticated using `SrmLoginSites`.

Table 2-4. Service Instance

Method	Description of Operation
`GetSiteName`	Get the name of the current local site. (deprecated in 6.5)
`GetPairedSite`	Retrieve information about the remote site that is paired with this local site.
`RetrieveContent`	Retrieve the properties of a service instance. Additionally the `AboutInfo` data object provides information about this service.
`GetLocalSiteInfo`	Get information about the local site

Table 2-5. Inventory Mapping

Method	Description of Operation
`AddFolderMapping`	Add a folder mapping between the primary and secondary vCenter Server.
`AddNetworkMapping`	Add a network mapping between the primary and secondary vCenter Server.
`AddResourcePoolMapping`	Add resource pool mapping between primary and secondary vCenter Server.
`AddTestNetworkMapping`	Add a test network mapping.
`RemoveFolderMapping`	Remove a folder mapping.
`RemoveNetworkMapping`	Remove a network mapping.
`RemoveResourcePoolMapping`	Remove a resource pool mapping.
`RemoveTestNetworkMapping`	Remove a test network mapping.

Table 2-6. Storage

Method	Description of Operation
DiscoverDevices	Loops through all array managers, local and remote, discovering devices.
QueryArrayManagers	Returns a list of all the available array managers.

Table 2-7. Protection

Method	Description of Operation
`ListProtectionGroups`	Get a list of the protection groups that are currently configured.
`ListInventoryMappings`	(New in 8.2) Get a list of the configured inventory mappings on the protection site.
`ListReplicatedDatastores`	Get a list of the replicated datastores. (deprecated in 6.0)
`GetProtectionGroupRootFolder`	Get the root folder for all protection groups, so as to navigate folder hierarchy. The methods below give users the ability to locate replicated resources and construct protection groups, key features of the 5.8 API.
`ListUnassignedReplicatedDatastores`	Get list of replicated datastores that can be used to create new protection groups.

Table 2-7. Protection (continued)

Method	Description of Operation
`ProtectionListProtectedDatastores`	Get list of replicated datastores that are protected by Site Recovery Manager.
`ListUnassignedReplicatedVms`	Get list of replicated VMs not currently assigned to a Site Recovery Manager protection group.
`ProtectionListProtectedVms`	Get list of virtual machines that are protected by Site Recovery Manager.
`CreateAbrProtectionGroup`	Create an array-based replication (ABR) protection group. The method returns a `CreateProtectionGroupTask`.
`CreateHbrProtectionGroup`	Create a host-based replication (HBR) protection group. The method returns a `CreateProtectionGroupTask`.
`RemoveProtectionGroup`	Delete a protection group.

Table 2-8. Protection Group Folder

Method	Description of Operation
`ListChildProtectionGroupFolders`	Return the child `ProtectionGroupFolders` located in this folder.
`ListChildProtectionGroups`	Return the `SrmProtectionGroup` objects located in this folder.
`GetProtectionGroup`	Return a reference to the protection group.
`GetParentFolder`	Get parent folder for a `ProtectionGroupFolder` or `RecoveryPlanFolder`.

Table 2-9. Create Protection Group Task

Method	Description of Operation
`IsCreateProtectionGroupComplete`	Check completeness of the task to create a new protection group.
`GetCreateProtectionGroupResult`	Once task is complete, check the result to ensure that it succeeded.
`GetNewProtectionGroup`	After checking task result, obtain the newly created `SrmProtectionGroup`.

Table 2-10. Protection Group

Method	Description of Operation
`GetInfo`	Retrieve basic information about this protection group.
`ProtectionGroupGetParentFolder`	Retrieve the folder that contains this protection group.
`GetPeer`	Retrieve the peer protection group.
`ListProtectedVms`	List VMs protected in this group with information about their protection state.
`ListProtectedDatastores`	Retrieve a list of the Datastores protected by this protection group.
`ListAssociatedVms`	Retrieve a list of VMs associated with this group. Only for vSphere Replication.

Table 2-10. Protection Group (continued)

Method	Description of Operation
GetProtectionState	Get the current state of the protection group.
ProtectionGroupListRecoveryPlans	Retrieve a list of all Recovery Plans this protection group is a member of.
ProtectionGroupQueryVmProtection	Determine whether the specified VMs can be or currently are protected, which must be mapped to the recovery site as per ListInventoryMappings.
ProtectVms	Protect the specified VMs. The folder, resource pool, and network of each virtual machine must be mapped to the recovery site. Returns a ProtectionTask.
UnprotectVms	Unprotect the specified VMs.
AssociateVms	Associate the specified VMs with a group. This is a prerequisite for protection. Only for vSphere Replication.
UnassociateVms	Unassociate the specified VMs with this group. Only for vSphere Replication.
ProtectionGroupGetOperationalLocation	Get the effective location of the protection group.
CheckConfigured	Check the protection group for VMs that are not configured, have configuration issues, and protected VMs that must be configured.
addDatastores	(New in 8.1) Adds datastores to the protection group. Additionally, the virtual machines on these datastores can be protected by the protection group. This can be done by calling protectVms method from this interface.
removeDatastores	(New in 8.1) Removes datastores from the protection group. Virtual machines on the removed datastores are no longer protected by the protection group.

Table 2-11. Protection Task

Method	Description of Operation
GetProtectionStatus	Get the results of ProtectVms or UnprotectVms
GetTasks	Get Task information from the vCenter Server for each virtual machine that was requested to be protected or unprotected.
IsComplete	Check if this Task has finished.
GetResult	Get the results of this Task.

Table 2-12. Recovery

Method	Description of Operation
ListPlans	Retrieve all the Recovery Plans for Site Recovery Manager Server.
GetHistory	Retrieve the history for a given Recovery Plan.
GetRecoveryPlanRootFolder	Retrieve the root folder for all Recovery Plans.

Table 2-12. Recovery (continued)

Method	Description of Operation
`CreateRecoveryPlan`	Create a recovery plan.
`DeleteRecoveryPlan`	Delete a recovery plan.

Table 2-13. Recovery Plan Folder

Method	Description of Operation
`ListChildRecoveryPlanFolders`	Return the child `RecoveryPlanFolders` located in the folder.
`ListChildRecoveryPlans`	Return an array of `SrmRecoveryPlan` objects located in the folder.
`GetRecoveryPlan`	`Get MoRef` to recovery plan with the specified name in the `RecoveryPlanFolder`.
`GetName, GetParentFolder`	See Table 2-8. Protection Group Folder

Table 2-14. Recovery Plan

Method	Description of Operation
`RecoveryPlanGetInfo`	Retrieve basic information about the specified Recovery Plan.
`RecoveryPlanGetPeer`	Get the peer plan for this Recovery Plan. The returned object refers to a plan at the paired site, not the local site.
`Start`	Start the Recovery Plan in a selected mode: test, cleanupTest, recovery, or reprotect. Requires Run privilege for tests, and the Failover privilege for the others.
`Cancel`	Cancel the specified Recovery Plan.
`ListPrompts`	List the current prompts that are waiting for input. When a prompt step is reached, the plan goes into the waiting state until `AnswerPrompt` is received. Prompts are given in the same order in which VMs are scheduled to start up.
`AnswerPrompt`	Answer the current prompt displayed by a Recovery Plan. Requires the Run privilege for test, or the Failover privilege for the other modes.
`RecoveryPlanGetParentFolder`	Retrieve the root folder for all Recovery Plans.
`GetRecoverySettings`	Retrieve the per-VM recovery settings for VMs in the Recovery Plan.
`SetRecoverySettings`	Modify the per-VM recovery settings for VMs in the Recovery Plan. (New in 8.1) Configure the IP address and corresponding DNS, WINS of the virtual machine, after the migration is complete, using the VmIpCustomization API.
`AddProtectionGroup`	Add a protection group to this Recovery Plan.
`RemoveProtectionGroupFromRecoveryPlan`	Remove a protection group from a recovery plan.
`AddTestNetworkMappingToRecoveryPlan`	Add a test network mapping to a recovery plan.
`RemoveTestNetworkMappingFromRecoveryPlan`	Remove a test network mapping from a recovery plan.

Table 2-14. Recovery Plan (continued)

Method	Description of Operation
`RecoveryPlanGetLocation`	Check whether the recovery plan is hosted locally or on the paired site.
`RecoveryPlanHasRunningTask`	Check whether there is a task that is associated with the recovery plan.

Table 2-15. Recovery History

Method	Description of Operation
`GetRecoveryResult`	Retrieve the recovery result for a given run of a Recovery Plan.
`GetResultCount`	Retrieve total number of stored results, including Recovery and peer plans.
`GetResultLength`	Get length of XML result document for the requested recovery result.
`RetrieveStatus`	Retrieve XML document for a historical run of the specified Recovery Plan.

Table 2-16. Array Manager

Method	Description of Operation
`ReadInfo`	(New in 8.2) Returns information specific to the ArrayManager instance.
`QueryReplicatedArrayPairs`	(New in 8.2) Returns list of all the replicated array pairs in the ArrayManager.

Table 2-17. Replicated Array Pair

Method	Description of Operation
`QueryReplicatedRdms`	(New in 8.2) Returns info for all replicated RDMs in the ReplicatedArrayPair.

Table 2-18. Deprecated DisasterRecoveryApi

Method	Description of Operation
`Login`, `Logout`, `LoginByToken`	Log in to and out of Site Recovery Manager Server.
`GetApiVersion`	Obtain the API version.
`ListRecoveryPlans`	Get a list of Recovery Plans at the SRM site.
`RecoveryPlanSettings`	Get the settings of a specific Recovery Plan at the SRM site.
`RecoveryPlanStart`	Start a specific Recovery Plan in recovery or test mode.
`RecoveryPlanPause`	Pause a running Recovery Plan.
`RecoveryPlanResume`	Restart a paused Recovery Plan.
`RecoveryPlanAnswerPrompt`	Answer a prompt.

Table 2-18. Deprecated DisasterRecoveryApi (continued)

Method	Description of Operation
`RecoveryPlanCancel`	Cancel a Recovery Plan.
`GetFinalStatus`	Get the final status of a Recovery Plan.

New vCenter Single Sign-On APIs

A set of login-by-token functions was added to the ServiceInstance managed object. For an example of use, see the Functions for Logging Into Sites Logging into Sites with SAML Tokens.

Deprecated APIs

The version 1.0 DisasterRecoveryApi was discontinued in Site Recovery Manager 5.8 and marked deprecated in Site Recovery Manager 6.0, although a new login-by-token function was implemented for backward compatibility.

Note The InvalidLogin fault and others use a different namespace in DisasterRecoveryApi (drextapi.fault.InvalidLogin) than in ServiceInstance (vim.fault.InvalidLogin).

In the RemoteSite managed object, the vcHost and vcPort fields are deprecated. They are replaced by the lkpUrl (Lookup Service URL) and vcInstanceUUID (vCenter Server unique ID).

The `GetSiteName` method is deprecated in Site Recovery Manager 6.5. You must use `LocalSiteInfo.siteInfo` to get the local site name.

Managed Object Hierarchy

The Managed object hierarchy table shows the Site Recovery Manager managed object hierarchy with the methods of each managed object in alphabetical order.

Table 2-19. Managed object hierarchy

Managed Object	Remarks	Local Methods
ArrayManager	Query information about array managers	`ReadInfo` `QueryReplicatedArrayPairs`

CreateRecoveryPlanTask	Contains the status of the operation	`GetCreateRecovryPlanFailure`
		`GetNewRecoveryPlan`
		`IsCreateRecoveryPlanComplete`

CreateProtectionGroupTask	Handle an ABR or HBR protection group	`GetCreateProtectionGroupResult`
		`GetNewProtectionGroup`
		`IsCreateProtectionGroupComplete`

DeleteRecoveryPlanTask	Contains the status of the operation	`GetDeleteRecovryPlanFailure` `IsDeleteRecoveryPlanComplete`
DiscoverDevicesTask	Contains the status of the operation	`GetDiscoverDevicesTaskFailures` `IsDiscoverDevicesTaskComplete`

Table 2-19. Managed object hierarchy (continued)

Managed Object	Remarks	Local Methods
Folder	Site Recovery Manager folder class	`GetName` `GetParentFolder`

InventoryMapping	Map items from the local to the remote site	`AddFolderMapping`
		`AddNetworkMapping`
		`AddResourcePoolMapping`
		`AddTestNetworkMapping`
		`RemoveFolderMapping`
		`RemoveNetworkMapping`
		`RemoveResourcePoolMapping`
		`RemoveTestNetworkMapping`

SrmProtection	Create an ABR or HBR protection group, list inventory mappings, query datastores and VMs, and list protection groups	`CreateAbrProtectionGroup`
		`CreateHbrProtectionGroup`
		`GetProtectionGroupRootFolder`
		`ListInventoryMappings`
		`ListProtectionGroups`
		`ListReplicatedDatastores` (deprecated in 6.0)
		`ListUnassignedReplicatedDatastores`
		`ListUnassignedReplicatedVms`
		`ProtectionListProtectedDatastores`
		`ProtectionListProtectedVms`
		`RemoveProtectionGroup`

SrmProtectionGroup	Add virtual machines to a protection group, get peer, query protected datastores, add datastore, and remove datastore	`AssociateVms`
		`GetPeer`
		`GetProtectionState`
		`ListAssociatedVms`
		`ListProtectedDatastores`
		`ListProtectedVms`
		`ProtectionGroupGetParentFolder`
		`ProtectionGroupListRecoveryPlans`
		`ProtectionGroupQueryVmProtection`
		`ProtectVms`
		`UnassociateVms`
		`UnprotectVms`
		`ProtectionGroupGetOperationalLocation`
		`CheckConfigured`
		`GetInfoGetPeer`
		`addDatastores`
		`removeDatastores`

ProtectionGroupFolder	Site Recovery Manager folder for protection groups	`GetProtectionGroup`
		`ListChildProtectionGroupFolders`
		`ListChildProtectionGroups`

Table 2-19. Managed object hierarchy (continued)

Managed Object	Remarks	Local Methods

ProtectionTask	Handle VM protection	`GetProtectionStatus`
		`GetResult`
		`GetTasks`
		`IsComplete`

RecoveryPlanFolder	Site Recovery Manager folder for recovery plans	`GetRecoveryPlan`
		`ListChildRecoveryPlanFolders`
		`ListChildRecoveryPlans`

RemoveProtectionGroupTask	Handle protection group removal	`GetRemoveProtectionGroupResult` `IsRemoveProtectionGroupComplete`
ReplicatedArrayPair	Query info about RDM devices	`queryReplicatedRdms`

SrmRecovery	Query recovery plans	`CreateRecoveryPlan`
		`DeleteRecoveryPlan`
		`GetHistory`
		`GetRecoveryPlanRootFolder`
		`ListPlans`

SrmRecoveryPlan	Run a recovery plan	`AddProtectionGroup`
		`AddTestNetworkMappingToRecoveryPlan`
		`AnswerPrompt`
		`Cancel`
		`GetRecoverySettings`
		`ListPrompts`
		`RecoveryPlanGetInfo`
		`RecoveryPlanGetPeer`
		`RecoveryPlanGetParentFolder`
		`RecoveryPlanGetLocation`
		`RemoveProtectionGroupFromRecoveryPlan`
		`RecoveryPlanHasRunningTask`
		`RemoveTestNetworkMappingFromRecoveryPlan`
		`SetRecoverySettings`
		`Start`

SrmStorage	Access the storage	`DiscoverDevices` `QueryArrayManagers`

SrmRecoveryHistory	Recovery plan status	`GetRecoveryResult`
		`GetResultCount`
		`GetResultLength`
		`RetrieveStatus`

Table 2-19. Managed object hierarchy (continued)

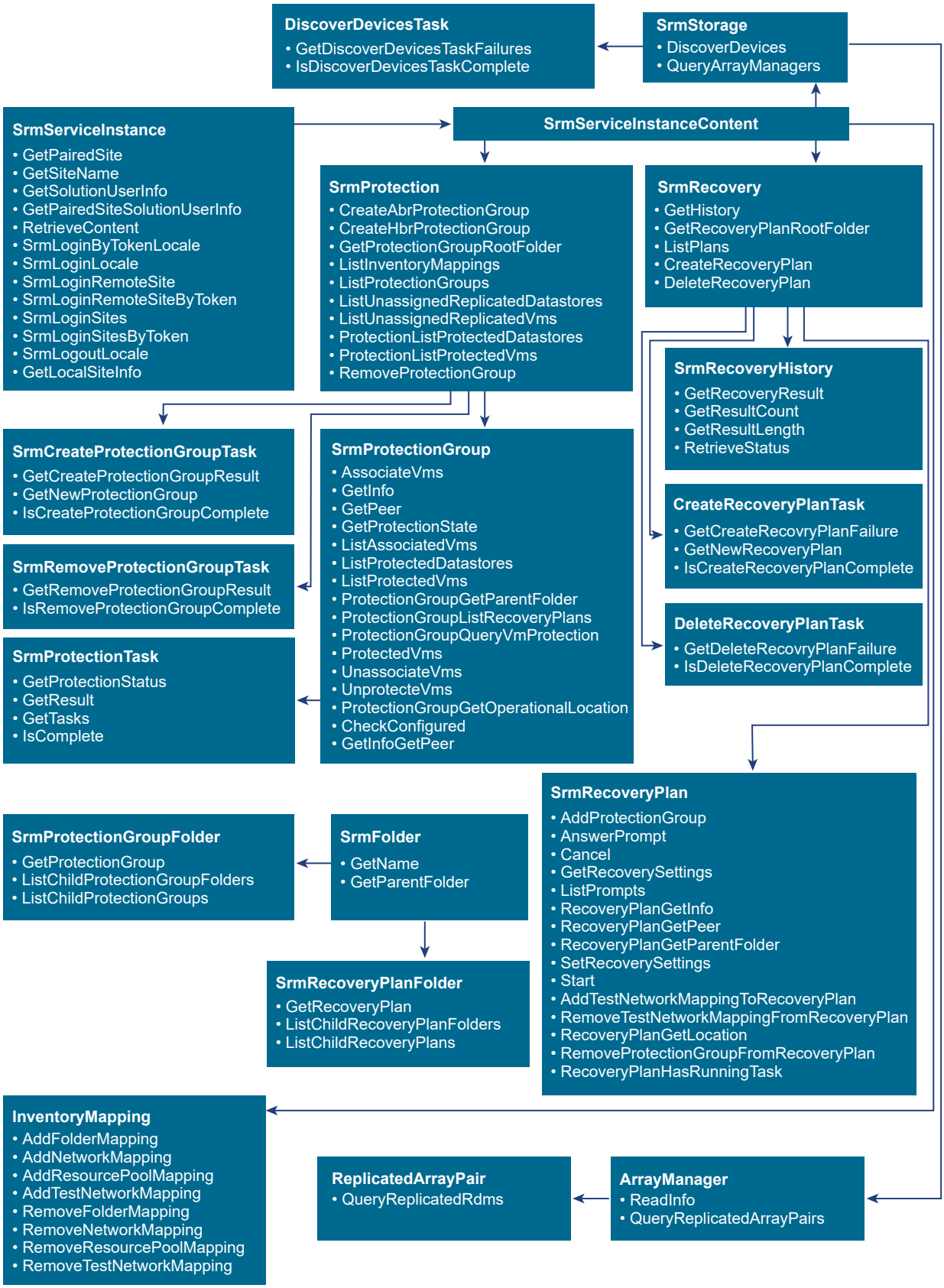
Managed Object	Remarks	Local Methods

ServiceInstance	Open or close session, get information about local and remote sites	`GetPairedSite`
		`GetSiteName` (deprecated in 6.5)
		`RetrieveContent`
		`SrmLoginLocale`
		`SrmLoginRemoteSite`
		`SrmLoginSites`
		`SrmLogoutLocale`
		`GetLocalSiteInfo`
		New in 6.0:
		`GetSolutionUserInfo`
		`GetPairedSiteSolutionUserInfo`
		`SrmLoginByTokenLocale`
		`SrmLoginRemoteSiteByToken`
		`SrmLoginSitesByToken`

DisasterRecoveryApi	Old version 1.0 API, deprecated but still provided for backward compatibility	`GetApiVersion`
		`GetFinalStatus`
		`ListRecoveryPlans`
		`RecoveryPlanAnswerPrompt`
		`RecoveryPlanCancel`
		`RecoveryPlanPause`
		`RecoveryPlanResume`
		`RecoveryPlanSettings`
		`RecoveryPlanStart`
		`SrmLogin`
		`SrmLoginByToken`
		`SrmLogout`

The SRM Object Classes graphic shows the Site Recovery Manager managed object class hierarchy with the methods of each managed object.

Figure 2-1. SRM Object Classes



WSDL Programming Environments

You can program Web services and read WSDL files using the C# language with Visual Studio .NET, or using the Java language with the Axis framework or the JAX-WS framework. You can program Web services using many other languages and frameworks, but they are beyond the scope of this manual.

Java JAX-WS Framework

The SDK provides sample code that uses the Java Development Kit (JDK) 1.6 with the JAX-WS framework bundled with the JDK 1.6. The build scripts generate Java stubs from the Site Recovery Manager specific WSDL.

C# and Visual Studio

The Site Recovery Manager SDK provides sample C# .NET code prepared for use with Visual Studio 2008, which you can convert for use with Visual Studio 2010 and perhaps later versions as well.

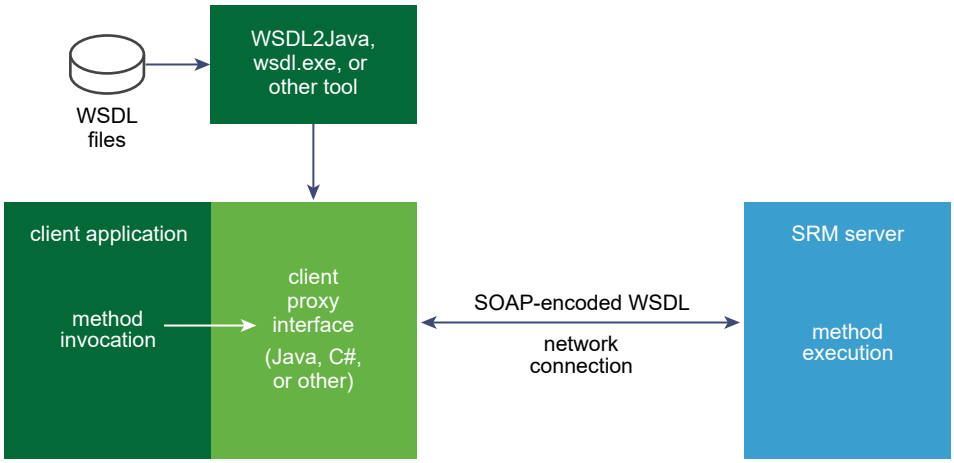
Java Axis Framework

The Site Recovery Manager SDK provides legacy sample code that requires Java SE 1.5 or later and Apache Axis 1.4. Samples are set up for stub generation on Windows or on Linux.

Managed Objects as WSDL

The WSDL Programming Components graphic shows the WSDL programming components used by various language frameworks.

Figure 2-2. WSDL Programming Components



Site Recovery Manager managed objects and methods are derived from classes and methods in the vSphere API, also known as the virtual machine object description language (VMODL). In the SDK, the Site Recovery Manager interfaces are mixed in with vSphere interfaces. For specific information about vSphere interfaces, see the *vSphere API Reference* manual, which is in the *vSphere Web Services SDK* under the *VMware vSphere Management SDK*.

Accessing VMware Site Recovery Manager

The Site Recovery Manager API provides language-neutral interfaces to the Site Recovery Manager Server management framework. Interfaces are provided for managing protection groups and recovery plans. Both array-based replication and vSphere Replication are supported.

Location of the API

The Site Recovery Manager 8.2 API is located at the following endpoints:

- Site Recovery Manager APIs for Windows

  `https://<FQDN_Server_or_IP_Address>:9086/vcdr/extapi/sdk`

- Site Recovery Manager APIs for Photon Virtual Appliance (VA)

  `https://<FQDN_Server_or_IP_Address>:443/drserver/vcdr/extapi/sdk`

All services use this single network port, and all communications are TLS encrypted. SSLv3 is disabled for security reasons. The API is implemented as an industry-standard Web service running on Site Recovery Manager Server.

Obtaining WSDL for APIs

The API complies with the Web Services Interoperability Organization (WS-I) Basic Profile 1.0, which includes XML Schema 1.0, SOAP version 1.1, and WSDL version 1.1. For details about WS-I Basic Profile 1.0, see the http://www.ws-i.org Web site.

You can obtain the WSDL for Site Recovery Manager API by requesting the file srm-Service.wsdl from the server root path.

- On a Windows server, the root path can be https://*<FQDN_Server_or_IP_Address>*:9086/srm-Service.wsdl
- On a Virtual Appliance, the root path can be https://*<FQDN_Server_or_IP_Address>*:443/drserver/srm-Service.wsdl

Associated vCenter Servers

As of SRM 6.0, Platform Services Controller and vCenter Server are associated with the Site Recovery Manager Server at both the local (protected) and the remote (recovery) sites.

Platform Services Controller can be embedded in vCenter Server, or it can be hosted on a separate machine. Platform Services Controller performs three services: Lookup, vCenter Single Sign-On, and Licensing.

The vCenter Server performs tagging and authorization for Site Recovery Manager. A system administrator installs the Site Recovery Manager plug-in at both local and remote sites to control the site's Site Recovery Manager Server through vCenter Server.

Managed object SrmServiceInstance provides functions for local and remote site discovery. You obtain the local site information with `getLocalSiteInfo`, and obtain the local solution user with `GetSolutionUserInfo`.

The local Platform Services Controller LookupService does not know anything about services on the remote site. You obtain the remote site name with `GetPairedSite` and obtain the remote solution user with `GetPairedSiteSolutionUserInfo`. The RemoteSiteobject contains the URL of the remote LookupService, and the UUID of the remote vCenter Server.

Logging into Sites with SAML Tokens

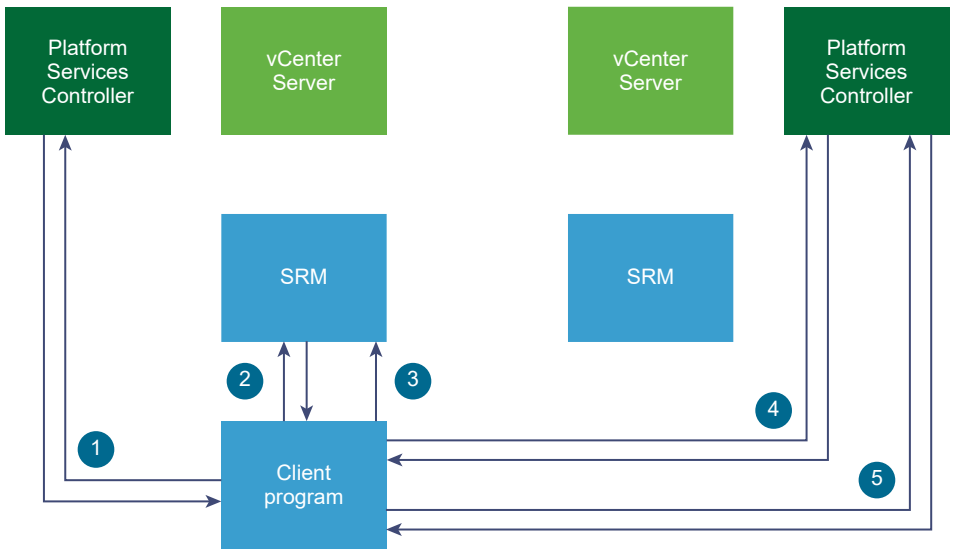
Site Recovery Manager release 6.0 improves security by obtaining a security assertion markup language (SAML) token from the vCenter Single Sign-On service for both the local and remote sites.

Table 2-20. Functions for Logging Into Sites

Function	Description of Operation
`GetSolutionUserInfo`	Obtain the UUID of Site Recovery Manager Server and the Site Recovery Manager solution user name.
`SrmLoginByTokenLocale`	After obtaining a token from vCenter Single Sign-On, begin session with the local Site Recovery Manager Server
`GetPairedSiteSolutionUserInfo`	Obtain the remote the UUID of Site Recovery Manager Server and the solution user name
`SrmLoginRemoteSiteByToken`	After obtaining remote token, begin session with the paired Site Recovery Manager Server
`SrmLoginSitesByToken`	Log in to both local and remote Site Recovery Manager Server, passing both SAML tokens

The following figure shows the sequence of calling for LoginSitesByToken

Figure 2-3. Calling Sequence for LoginSitesByToken



Order of operations

1 Obtain local token from the vCenter Single Sign-On service located on the local Platform Services Controller.

2 Get remote site information from Site Recovery Manager, and extract the URL of remote LookupService.

3 Use remote LookupService to find the remote vCenter Single Sign-On service.

4 Obtain remote access SAML token from vCenter Single Sign-On service located on the remote Platform Services Controller.

5 Make the SrmLoginSitesByToken call locally to Site Recovery Manager.

SDK Installation and Setup

3

The SDK Installation and Setup chapter describes how to unpack and use the software development kit (SDK).

This chapter includes the following topics:

Contents of the SDK Package

The Site Recovery Manager SDK is delivered as a ZIP archive (`VMware-srm-sdk-<version>-<build>.zip` file.

You can obtain the SDK package by navigating to http://www.vmware.com/support/developer/srm-api and clicking the **Download SDK** link. You will need to provide an email address or customer number, with valid password, for authentication on the Site Recovery Manager download site.

The package contains:

- The WSDL and XML schema files that define the API for Site Recovery Manager Server management.
- Sample C# code for .NET and Java code for JAX-WS or Axis showing how to display a recovery plan.
- Batch files and shell scripts to automate the process of generating client-side stubs, and for rebuilding the sample applications. For C# developers, Visual Studio project and solution files are included.
- Documentation, including the VMware Infrastructure SDK Reference Guide (the previous name for the vSphere API Reference) with language-neutral descriptive information about object type definitions, properties, and method signatures for the Site Recovery Manager API.

SDK Directory Structure

After you unzip the Site Recovery Manager SDK, the following directories and sub-directories appear. Many of the sub-directories contain helpful readme files.

Table 3-1. SDK directory structure

Directory or File	Description
/doc	Contains SDK README files and reference documentation for the SDK.
/doc/ReferenceGuide	API Reference for the Site Recovery Manager API.
/doc/ReferenceGuide/index.html	To view the API Reference, open `index.html` with a Web browser.
/doc/SDK_Terms_and_Conditions.*	End user license agreement for the Site Recovery Manager SDK.
/samples	Top-level directory for language-specific versions of sample client applications.
/samples/DotNet/	Directory containing command scripts to generate the .NET proxy classes and Web service stubs. The `GeneratingStubs.txt` file gives helpful notes about how to generate stubs with your own namespace for Visual Studio 2008.
/samples/DotNet/cs	Directory containing Visual Studio 2008 solution (.sln) file and sub-directories with C# `AppUtil` support code and `RecoveryPlan.cs` sample application with project (.csproj) file.
/samples/JAXWS/	Directory containing Java source code for the JAX-WS framework. Sample program RecoveryPlanList.java is in the `com/vmware/samples/recovery` subdirectory. Shell scripts and batch files are provided to build and run the sample program.
/samples/Axis/	Directory containing Java source code for the Axis framework. Sample program RecoveryPlanList.java is in the `com/vmware/samples/recovery` subdirectory. Shell scripts and batch files are provided to build and run the sample program.
/wsdl/srm/srm.wsdl	The Web services description language (WSDL) file containing definitions for the Site Recovery Manager API.
/wsdl/srm/srm-Service.wsdl	The WSDL file defining the Web services endpoint at which the API is available. This file references srm.wsdl with an import statement, so you should use the appropriate generation tool with srm-Service.wsdl (not srm.wsdl directly).
/wsdl/srm/*.xsd	XML schema definition files (six).

Download and Setup

Setting up your environment to develop client applications with the SDK involves a number of steps, but if you are already developing vSphere applications, some of the steps are unnecessary.

Procedure

1. Choose a programming language (C# or Java) for Web services client application development. You can use Linux or Windows for Java development. C# development is done on Windows.
2. Identify the target VMware Site Recovery Manager server (or servers) to use for development. A “target server” is a Site Recovery Manager server that your client application will manage.
3. Install, or verify presence of, the development environment appropriate for your programming language.
   - For C#, you need one of the Microsoft development environments, such as Visual Studio 2008 or Microsoft Visual C#. VMware recommends using Microsoft Visual Studio 2008 or later, which includes the required .NET Framework. For more information, visit the MSDN Web site.
   - You can use Java Standard Edition (SE) 6.0 or 7.0. VMware recommends Java Development Kit (JDK) 1.7.0_45 or later. For more information, visit the Oracle Java Web site. Open JDK works also.
4. Obtain the appropriate Web services client tools (XML parser, WSDL-to-proxy-code generation tools, and runtime) for your programming language.
   - For C#, you need Microsoft .NET Framework 2.0 or 1.1. If you already use Microsoft development tools, it is likely you already have this. You can obtain the .NET Framework 2.0 from MSDN. You also need the .NET 2.0 Software Development Kit, which includes the WSDL-to-stub generation tool (`wsdl.exe`) and the command-line C# compiler (`csc.exe`), both of which get called from the `gensrmstubs.cmd` script. You can get the .NET 2.0 Software Development Kit from Microsoft: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=19988.
   - For Java with JAX-WS, you can use the JAX-WS framework that comes with the JDK.
   - For Java with Axis, you need the Apache Axis 1.4 client-side Web service libraries. For documentation and downloads, visit the Axis Apache Web site.
5. The SDK includes sample code to list and (with additions) prepare to test a recovery plan.

What to do next

- For C# .NET, see About the C# .NET Samples
- For JAX-WS, see About Java JAX-WS Samples
- For Java Axis, see About Java Axis Samples

About the C# .NET Samples

Currently the SDK includes recovery plan sample code that you can build on .NET. You can build C# .NET samples using Microsoft Visual C# 2008 Express or Microsoft Visual Studio 2008.

If you have a later version of Visual Studio, it might ask to convert the solution and project files before proceeding. An earlier version of this SDK supported Visual Studio 2005. Visual Studio 2003 was never supported because of performance issues (.NET took a long time to instantiate the `VimService` class).

Build Sample Code with Visual Studio 2008

You can build C# .NET samples using Microsoft Visual C# 2008 Express or Microsoft Visual Studio 2008.

Procedure

1. Open a Visual Studio 2008 command prompt from the Windows Start Menu as follows: **Start > Programs > Visual Studio 2008 > Visual Studio Tools > Command Prompt**.
2. Start the Windows command prompt.

   On 64-bit Windows systems, run `C:\Windows\SysWOW64\cmd.exe` so the sample programs execute under Windows 32-bit on Windows 64-bit (WOW64).
3. Navigate to the `SDK\samples\DotNet` sub-directory.
4. At the command prompt, type **`Build2008.cmd`** to execute the build commands.

Build Sample Code with Visual C# 2008 Express

You can build C# .NET samples using Microsoft Visual C# 2008 Express or Microsoft Visual Studio 2008. If you have a later version of Visual Studio, it might ask to convert the solution and project files before proceeding.

Procedure

1. Select the default (Full) Microsoft Visual C# 2008 Express installation.
2. If you installed Visual C# 2008 Express in the default location, skip this step. Otherwise:
   a. Create the System environment variable `VSINSTALLDIR`.
   b. Set the `VSINSTALLDIR` environment variable to the location of the Microsoft Visual Studio tools, in the `Common7` sub-directory of the Microsoft Visual C# 2008 Express installation. Default locations are shown below. Use quotation marks around directory names that contain spaces, as these do.

      “`C:\Program Files\Microsoft Visual Studio 9.0\Common7`"

      “`C:\apps\Microsoft Visual Studio 9.0\Common7`”

      If Visual C# Express is installed in its default folder `C:\Program Files\Microsoft Visual Studio 9.0`, you do not need to create or set the `VSINSTALLDIR` environment variable.
3. Open a Visual Studio 2008 command prompt from the Windows Start Menu as follows: **Start > Programs > Visual Studio 2008 > Visual Studio Tools > Command Prompt**
4. Navigate to the `SDK\samples\DotNet` sub-directory.

5 At the command prompt, type **Build2008.cmd** to execute the build commands.

The build process generates the RecoveryPlan sample program, which lists all recovery plans and optionally gets state for the specified recovery plan. A sample build can be executed from the \bin or \debug directory of a project. You can also run samples from within Visual Studio, at the .NET command prompt. To display help text for any application, you can run the application without any parameters.

Run Sample Code from Visual Studio

You can build C# .NET samples using Microsoft Visual C# 2008 Express or Microsoft Visual Studio 2008.

Procedure

1 Start Visual Studio.

2 Open the Sample2008.sln solution file.

3 Change the Project Properties to specify the command line arguments:

a From the **Project** menu, select **Properties** to display the **Property Pages** dialog.

b In the Project_Name Property Pages dialog, select **Configuration Properties—Debugging** on the left.

c In the right-hand pane (under Start Options), select **Command Line Arguments**.

d Click **OK** to save your changes.

4 Run the sample code at the command prompt.

Run C# Sample Code

You can build C# .NET samples using Microsoft Visual C# 2008 Express or Microsoft Visual Studio 2008.

Procedure

1 After you generate the sample program, you can run it as follows: `RecoveryPlan –-url <webserviceurl> –-username <user> –-password <passwd> –-planname <plan>`

The `RecoveryPlan` program lists all recovery plans and optionally gets the state for the plan specified after the `–-planname` option.

2 You can remove build files by running the `clean.bat` batch script.

About Java JAX-WS Samples

This section describes how to build and run the sample program that uses JAX-WS bindings for the Site Recovery Manager API.

The program was developed to work with the JAX-WS framework that is bundled with the JDK 1.6 and later Java source code is located in the `samples/JAXWS/com/vmware/samples/recovery` directory, as extracted from the ZIP archive.

Build JAX-WS Sample Code

The sample program that uses JAX-WS bindings for the Site Recovery Manager API was developed to work with the JAX-WS framework that is bundled with the JDK 1.6 and later.

Procedure

1. Set the `JAVAHOME` environment variable to the base directory of your installed JDK.

   On Linux this could be `/usr/lib/jvm/java-7-openjdk-i386` for example.

   On Windows this could be `C:\Program Files\Java\jdk1.7.0_65` for example.

2. Change directory to `sdk/samples/JAXWS` and run the `build.sh` script (or on Windows, the `build.bat` file) to generate the Site Recovery Manager API Java stubs from the `srm-Service.wsdl` definitions, generate the Java stubs, and compile the sample Java code into a class file.

   Note the WSDL file dependency: JAX-WS requires a WSDL file for stub generation and compilation. To manage this dependency, the build script performs the following operations:

   - It calls the wsimport JDK tool to generate Java stubs from the `srm-Service.wsdl` SRM WSDL file.
   - It specifies the wsimport -wsdlLocation command line option to identify the WSDL file location.
   - It copies the WSDL file and related schema files into the `srm.jar` file.

   To compile Java code that imports the generated stubs and uses the `srm.jar` built by the `build.sh` script, the WSDL file must be in the same location that was specified by the `-wsdlLocation` command line option. To establish this location, the build script modifies the `SrmService` class to reference the WSDL location inside the JAR file. Then you just need to add the `srm.jar` file to your class path.

Run JAX-WS Sample Code

After building, you can run the sample program that uses JAX-WS bindings for the Site Recovery Manager API. The program was developed to work with the JAX-WS framework that is bundled with the JDK 1.6 and later.

Procedure

1. Change directory to `sdk\samples\JAXWS`, where the JAR files are located, if you are not already there, and set `CLASSPATH`. Sometimes `%CLASSPATH%` has already been set system wide. Example settings for Linux and Windows are `export CLASSPATH=/usr/lib/jvm/java-7-openjdk-i386/lib` and `set CLASSPATH=%JAVAHOME%\lib`.

2. Define `VMKEYSTORE` as the path to the Java key store. This is needed to securely access Site Recovery Manager Server.

   ```
   export VMKEYSTORE=/usr/share/mime/application/x-java-keystore.xml
   ```

   ```
   set VMKEYSTORE=C:\cygwin\usr\share\mime\application\x-java-keystore.xml
   ```

   For more information about `VMKEYSTORE`, see SSL Certificates.

3 Call the run script or batch file. This sample program prints its usage summary, as if you specified --help on the command line, `run.sh com.vmware.samples.recovery.RecoveryPlanList` or `run.bat com.vmware.samples.recovery.RecoveryPlanList`.

4 As you can see from the usage message, the RecoveryPlanList sample code requires a user name and password for log in to the Site Recovery Manager administrator account. You need to pass in additional options: `––username srmadmin ––password secret ––planname myRecoveryPlan`

Clean Up JAX-WS Sample Code

You can clean up the JAX-WS Sample Code by using either a script or a batch file.

Procedure

1 Change directory to `sdk\samples\JAXWS`, if you are not already there.

2 Run the `clean.sh` script or the `clean.bat` batch file.

About Java Axis Samples

This section describes how to build and run the sample program that uses Axis Web services for the Site Recovery Manager API.

Axis can be downloaded from the Apache Web site, or as the libaxis-java package in some Linux distributions. Axis works with JDK 1.6 and later. Source code build files are located in the `samples/Axis/java` directory, as extracted from the ZIP archive.

Build Java Axis Sample Code

You can build the sample program that uses Axis Web services for the Site Recovery Manager API. Axis can be downloaded from the Apache Web site, or as the `libaxis-java` package in some Linux distributions. Axis works with JDK 1.6 and later. Source code build files are located in the `samples/Axis/java` directory, as extracted from the ZIP archive.

Procedure

1 Make sure the Java development kit and Apache Axis are installed and functioning.

2 Start the Linux terminal (shell) or Windows command prompt.

3 Set the environment variables as shown in the Java and Axis environment variables table.

Table 3-2. Java and Axis environment variables

Environment Variable	Description and Usage Notes	Example Setting
`AXISHOME`	Complete path to the top-level Axis installation directory. Must be set before using the build scripts.	`C:\Apache\axis1.4 /usr/share/java`
`JAVAHOME`	Path to the binary directory for the Java JDK.	`C:\Java\jdk1.7.0_65 /usr/lib/jvm/java-7-openjdk-i386`

4 Change directory to `sdk/samples/Axis/java` and run the `build.sh` script (or on Windows, the `build.bat` file) to compile the sample Java code into a class file.

5 If the build script produces error messages about missing classes (could not find or load a class), edit the script and change the `LOCALCLASSPATH` line so path names refer to the proper jar file versions. Some Java archives contain symbolic links where a generic file points to a specific version of the jar file.

   The script takes time to build the `RecoveryPlanList` sample program, which lists all recovery plans, or optionally gets state for a specified recovery plan.

   Note The sample program was written for Axis version 1. It may require modifications for version 2.

Run Java Axis Sample Code

Source code build files are located in the `samples/Axis/java` directory, as extracted from the ZIP archive.

Procedure

1 Change directory to `sdk/samples/JAXWS`, where the JAR files are located, if you are not already there, and set `CLASSPATH`. Example settings for Linux and Windows are `export CLASSPATH=/usr/lib/jvm/java-7-openjdk-i386/lib` and `set CLASSPATH=%JAVAHOME%\lib`

   Sometimes `CLASSPATH` has already been set system wide.

2 Define `VMKEYSTORE` as the path to the Java key store. This is needed to securely access a Site Recovery Manager Server, `export VMKEYSTORE=/usr/share/mime/application/x-java-keystore.xml` and `set VMKEYSTORE=C:\cygwin\usr\share\mime\application\x-java-keystore.xml`

3 After you build the sample program, you can call it with the run script as follows, `run.sh com.vmware.samples.recovery.RecoveryPlanList --url <srm-URL> --username <user> --password <passwd>`

   If you include the `--ignorecert` option, the sample code runs the following to get around an untrusted server certificate: `System.setProperty("org.apache.axis.components.net.SecureSocketFactory", "org.apache.axis.components.net.SunFakeTrustSocketFactory");`

Clean Up Java Axis Sample Code

You can clean up the JAVA Axis sample code by either running a script or a batch file.

Procedure

1 Change directory to `sdk/samples/Axis/java`, if you are not already there.

2 Run the `clean.sh` script or the `clean.bat` batch file.

Logical Order API Usage

4

This chapter contains the following major sections:

API descriptions in this chapter follow logical usage order of List of API Operations. In examples below, `MoRef` indicates a String that references a managed object.

This chapter includes the following topics:

Recovery Plans and Reprotect

You must use the UI, not the API, to initiate forced failover. It requires complicated set-up and validation steps.

Solution User Information

At install time Site Recovery Manager creates a solution user at the local and remote sites. This improves security by avoiding use of adminstrator@vsphere.local or the Windows administrator. Programs can obtain both solution users before Site Recovery Manager login because the privilege required for these functions is `System.Anonymous`.

GetSolutionUserInfo

Obtain the Site Recovery Manager solution user name and site UUID for the local site. You must call the `getSolutionUserInfo` method before logging in (with `SrmLoginByTokenLocale` for example) if you wish to delegate a token to the Site Recovery Manager solution user.

Synopsis

```
SolutionUserInfo getSolutionUserInfo()
```

`SolutionUserInfo.siteUuid` is a string with the immutable unique identifier of the Site Recovery Manager server.

`SolutionUserInfo.username` is the name of the Site Recovery Manager solution user.

Faults

- RuntimeFault

GetPairedSiteSolutionUserInfo

Obtain the Site Recovery Manager solution user name and site UUID for the paired remote site.

Synopsis

```
SolutionUserInfo getPairedSiteSolutionUserInfo()
```

`SolutionUserInfo.siteUuid` is a string with the immutable unique identifier of the remote Site Recovery Manager Server.

`SolutionUserInfo.username` is the name of the remote Site Recovery Manager solution user.

Faults

- RuntimeFault
- RemoteSiteNotEnabled, if the remote site is not enabled.

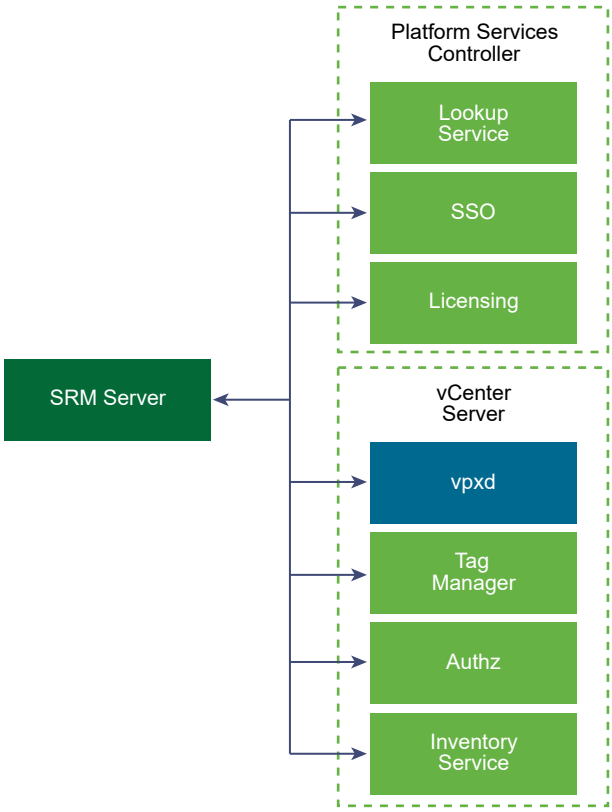
SAML Token Authentication

In the 6.0 release, Site Recovery Manager interacts with both vCenter Server (as before) and Platform Services Controller (PSC).

The PSC contains a Lookup Service to locate other services, a Licensing Service to replace the VIM license manager, and Single Sign On (SSO) service for authentication and token acquisition.

The vCenter Server management node contains a new Tag Manager to create categories and tags for Storage DRS or SPBM, and a new Authz service to replace the VIM authorization manager.

Figure 4-1. Management (M) and PSC (N) nodes



When programs connect to the local or remote SRM server, they must obtain a SAML token from the SSO service on the local or remote PSC. (The older login functions implicitly obtain a SAML token.)

SrmLoginByTokenLocale

This function begins a session with Site Recovery Manager Server.

Synopsis

`void`

`loginByToken`(String `samlToken`, @optional String `locale`)

`samlToken` is an XML encoded security assertion markup language (SAML) token for authenticating login to the SRM server. The token should either be a bearer token or a holder of key token delegated to the Site Recovery Manager solution user.

`locale` is the optional locale for this session.

Faults

- NoPermission if the user does not have permissions.
- AlreadyLoggedInFault if there is already an established session.
- InvalidLogin if the given token is not valid.
- InvalidLocale if the requested locale is invalid or unavailable.
- ConnectionLimitReached if the configured connection limit has been reached.
- RuntimeFault
- InvalidTokenLifetime if the SSO token is either expired or not yet valid.

SrmLoginSitesByToken

Log in to both the local and remote vCenter Servers using the provided tokens. This function is needed when escalated privileges are required to perform operations on the remote site, such as protecting virtual machines.

Synopsis

```
void loginSitesByToken(String samlToken,
        String remoteSamlToken,    @optional String locale)
```

`samlToken` is an XML encoded SAML token for authenticating login to the Site Recovery Manager server. The token should either be a bearer token or a holder of key token delegated to the Site Recovery Manager solution user.

`remoteSamlToken` is an XML encoded SAML token for authenticating login to the Site Recovery Manager server. The token should either be a bearer token or a holder of key token delegated to the remote Site Recovery Manager solution user.

`locale` is the optional locale for this session.

Faults

- AlreadyLoggedInFault if there is already an established session.
- InvalidLogin if the given token is not valid.
- InvalidLocale if the requested locale is invalid or unavailable.
- RemoteSiteNotEnabled if the remote site is not enabled.
- ConnectionLimitReached if the configured connection limit has been reached.
- RuntimeFault

SrmLoginRemoteSiteByToken

Log in to the remote Site Recovery Manager server using the provided credentials. This function may be called when escalated privileges are required on the remote site and the current session has already been authenticated by login.

Synopsis

```
void loginRemoteSiteByToken(String remoteSamlToken,
        @optional String locale)
```

`remoteSamlToken` is an XML encoded SAML token for authenticating login to the Site Recovery Manager server. The token should either be a bearer token or a holder of key token delegated to the remote Site Recovery Manager solution user.

`locale` is the optional locale for this session.

Faults

- NotAuthenticated if there is no session
- AlreadyLoggedInFault if there is already an established session.
- InvalidLogin if the given token is not valid.
- InvalidLocale if the requested locale is invalid or unavailable.
- RemoteSiteNotEnabled if the remote site is not enabled.
- ConnectionLimitReached if the configured connection limit has been reached.
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

Credential Based Authentication

In the 6.0 release, login functions in earlier Site Recovery Manager releases have been implemented on top of the SAML token authentication functions. The password-based login functions are now offered for convenience.

Programs can connect to the local Site Recovery Manager Server using the `SrmLoginLocale` function, or to Site Recovery Manager Server instances at both the (local) protected site and the (remote) recovery site using the `SrmLoginSites` API.

SrmLoginLocale

This method logs in to the Site Recovery Manager server. The `Connect` public method requires the URL of an Site Recovery Manager server and authentication credentials. The `SrmLoginLocale` method takes the `_srcRef` managed object reference from `SrmServiceInstance`, and fails if the user name and password combination is invalid, or if the user is already logged in. In these examples, a locale string could be provided instead of the null parameter.

Synopsis

`void SrmLoginLocale`(String `username`, String `password`, @optional String `locale`)

`username` – user name authorized for access to the local vCenter Server.

`password` – password for that user on the local vCenter Server

`locale` – name of the locale for this session

Faults

- InvalidLocale
- InvalidLogin
- NoPermission
- AlreadyLoggedInFault
- ConnectionLimitReached
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: C# code for Site Recovery Manager login

```
protected SrmService _service;
protected SrmServiceInstanceContent _sic;
protected ManagedObjectReference _svcRef;
...
public void Connect(string url, string username, string password)
{
_service = new SrmService();
_service.Url = url;
_service.Timeout = 600000;
_service.CookieContainer = new System.Net.CookieContainer();
_sic = _service.RetrieveContent(_svcRef);
_service.SrmLoginLocale(_svcRef, username, password, null);
...
}
```

The Java code is similar to the C# code but uses a service locator.

Example: Java code for Site Recovery Manager login

```
private static SrmPortType srmPort;
private static SrmServiceInstanceContent serviceContent;
private static boolean isConnected = false;
...
srmPort = srmService.getSrmPort();
Map<String, Object> ctxt =
((BindingProvider) srmPort).getRequestContext();
ctxt.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, url);
ctxt.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);
serviceContent = srmPort.retrieveContent(SVC_INST_REF);
srmPort.srmLoginLocale(SVC_INST_REF, userName, password, null);
isConnected = true;
```

Subsequent methods in the Site Recovery Manager are called as a subclass of `_service`, for example `_service.ListPlans()` in C# or `srmport.listPlans()` in Java.

SrmLoginSites

The `SrmLoginSites` API is very similar to `SrmLoginLocale`, except it takes an additional user name and password combination for the remote (usually recovery) site. The `SrmServiceInstance _this` is obtained from the local (usually protected) site.

Synopsis

`void SrmLoginSites`(String `username`, String `password`, String `remoteUser`, String `remotepass`,@optional String `locale`)

`username` – user name authorized for access to the local vCenter Server

`password` – password for that user on the local vCenter Server

`remoteUser` – user name authorized for access to the remote vCenter Server

`remotePass` – password for that user on the remote vCenter Server

`locale` – name of the locale for this session

Faults

- InvalidLocale
- InvalidLogin
- NoPermission
- RemoteSiteNotEnabled
- AlreadyLoggedInFault
- ConnectionLimitReached
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: C# and Java for double SRM login

```
/// C#
_service.SrmLoginSites(_svcRef, username, password, remoteuser, remotepass, locale);
// Java
_service.srmLoginSites(_svcRef, username, password, remoteuser, remotepass, locale);
```

SrmLogoutLocale

This method logs out of the Site Recovery Manager server and terminates the current session. It takes the same managed object reference as for `SrmLoginLocale`, and should be called with other methods to clean up a connection.

Synopsis

`void SrmLogoutLocale( )`

Faults

- NotAuthenticated
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: C# code to log out

```
public void Disconnect()
{
if (_service != null)
{
_service.SrmLogoutLocale(_svcRef);
_service.Dispose();
_service = null;
_sic = null;
}
}
```

The Java code is simpler than the C# code.

Example: Java code to log out

```
private static void disconnect() throws Exception {
if (isConnected) {
srmPort.srmLogoutLocale(SVC_INST_REF);
}
isConnected = false;
}
```

SrmLoginRemoteSite

Log in to the remote Site Recovery Manager server using the provided credentials. This function may be invoked when escalated privileges are required on the remote site and the current session has already been authenticated using login.

Synopsis

`void`

`SrmLoginRemoteSite`(String `remoteUser`, String `remotePassword`, @optional String `locale`)

`remoteUser` is a username to be used to login to the remote VirtualCenter server.

`remotePassword` is a password to be used to login to the remote VirtualCenter server.

`locale` is the locale for this session.

Faults

- InvalidLocale
- InvalidLogin

- AlreadyLoggedInFault
- ConnectionLimitReached
- RemoteSiteNotEnabled
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Service Instance

Site Recovery Manager methods take a managed object reference `_this`, which references the `SessionManager` used for making method calls. Programs obtain `_this` by retrieving content of the `ServiceInstance`, which is accomplished by creating a new managed object reference of type `SrmServiceInstance`.

Example: C# code to create `SrmServiceInstance`

```
public SvcConnection(string svcRefVal)
{
...
_svcRef = new ManagedObjectReference();
_svcRef.type = "SrmServiceInstance";
_svcRef.Value = svcRefVal;
}
```

The Java code is similar to the C# code.

Example: Java code to create `SrmServiceInstance`

```
private static final String SVC_INST_NAME = "SrmServiceInstance";
private static ManagedObjectReference SVC_INST_REF = new ManagedObjectReference();
...
SVC_INST_REF.setType(SVC_INST_NAME);
SVC_INST_REF.setValue(SVC_INST_NAME);
srmService = new SrmService();
```

GetSiteName

Gets the name of the current site. The method is deprecated.

Synopsis

`String getSiteName( )`

String representing the local Site Recovery Manager site name.

Faults

- RuntimeFault

GetPairedSite

This function gets the remote site paired with this site. A remote site may be acting as the secondary site for this site, or may be acting as the primary site with this site as its secondary. Most of the initial Site Recovery Manager calls work for everyone (`System.Anonymous` privilege) but `GetPairedSite` requires the `System.View` role, so it must be called after login to the local site. Also `GetPairedSiteSolutionUserInfo` is useless without the remote paired site, so the two must be called together.

Synopsis

```
RemoteSite getPairedSite( )
```

The `RemoteSite` class contains the following fields:

Field	Description
`name`	a String with the name of the site.
`uuid`	a String with the UUID of the site.
`vcHost`	a String with the DNS name or IP address of the remote vCenter Server.
`vcInstanceUuid`	a String with instance UUID of the vCenter Server associated with the remote Site Recovery Manager.
`vcPort`	an integer with the port used for VMOMI access to the remote vCenter Server.
`vcUrl`	a String with the URL of the remote vCenter Server
connected	a boolean that is true when the sites are connected and false when the sites are disconnected.
lkpUrl	a String with the URL of the remote LookupService server.

Faults

- RuntimeFault
- RemoteSiteNotEnabled if the remote site is not enabled.

Example: Example

```
String RemoteSite getPairedSite(_this)
```

RetrieveContent

Retrieves properties of the service instance.

Synopsis

```
ServiceInstanceContent retrieveContent()
```

The `ServiceInstanceContent` class contains the following fields.

Field	Description
about	shows information about this service.
apiVersion	represents the version of this API.
inventoryMapping	an external API to InventoryMappings.
protection	an external API to Protection.
recovery	an external API to Recovery.
storage	an external API to Storage.

Faults

- RuntimeFault

GetLocalSiteInfo

The `getLocalSiteInfo` method gets information about the local site.

Synopsis

```
LocalSiteInfo getLocalSiteInfo()
```

The method returns information about the local site such as site name, UUID, and URLs of the local LookupService server and vCenter Server.

Faults

RuntimeFault

Inventory Mappings

This section covers the Site Recovery Manager API methods for inventory (resource) mapping.

Resource mappings are the pairings of resources between the protected and recovery sites. In other words, mapping the networks, resource pools, datacenters and so forth on the protected site to their counterparts on the recovery site. This is done so that virtual machines will recover in the correct places on the recovery site. Previously this was done only in the UI, but APIs have been added to automate these mappings.

AddFolderMapping

Adds a folder mapping between a folder on the primary vCenter Server and another folder on the secondary vCenter Server.

Synopsis

`void addFolderMapping(vim.Folder primaryFolder, vim.Folder secondaryFolder)`

`primaryFolder` is a read-only `MoRef` to the folder on the protection site (must be local).

`secondaryFolder` is a read-only `MoRef` to the folder on the recovery site (must be remote).

Faults

- ConnectionDownFault
- InvalidPrimaryNetwork
- InvalidSecondaryFolder
- UnknownPrimaryNetwork
- UnknownSecondaryNetwork
- RuntimeFault
- NotAuthenticated

See Chapter 6 Faults in Site Recovery Manager for more details.

RemoveFolderMapping

The `removeFolderMapping` method removes a folder mapping. The method does not check whether the folders in the mapping exist on the protected and recovery sites. You can use the method to remove broken mappings.

Synopsis

```
void removeFolderMapping(Folder primaryFolder)
```

The `primaryFolder` parameter specifies the folder on the primary site whose mapping must be deleted.

Faults

- ConnectionDownFault
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

AddNetworkMapping

Adds a network mapping between a network on the primary vCenter Server and another network on the secondary vCenter Server.

Synopsis

`void addNetworkMapping(vim.Network primaryNetwork, vim.Network secondaryNetwork)`

`primaryNetwork` is a read-only `MoRef` to the network on the protection site (must be local).

`secondaryNetwork` is a read-only `MoRef` to the network on the recovery site (must be remote).

Faults

- ConnectionDownFault
- InvalidPrimaryNetwork
- InvalidSecondaryFolder
- UnknownPrimaryNetwork
- UnknownSecondaryNetwork
- RuntimeFault
- NotAuthenticated

See Chapter 6 Faults in Site Recovery Manager for more details.

RemoveNetworkMapping

The `removeNetworkMapping` method removes a network mapping. The method does not check whether the networks in the mapping exist on the protected and recovery sites. You can use the method to remove broken mappings.

Synopsis

```
void removeNetworkMapping(Network primaryNetwork)
```

The `primaryNetwork` parameter specifies the primary site network whose mapping must be deleted.

Faults

- ConnectionDownFault
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

AddResourcePoolMapping

Adds a resource pool mapping between a resource pool on the primary vCenter Server and another on the secondary vCenter Server.

Synopsis

`void addResourcePoolMapping(vim.ResourcePool primaryResourcePool,vim.ResourcePool secondaryResourcePool)`

`primaryPool` – The resource pool on the protection site (must be local).

`secondaryPool` – The resource pool on the recovery site (must be remote).

Faults

- UnknownPrimaryResourcePool

- UnknownSecondaryResourcePool
- ConnectionDownFault
- RuntimeFault
- NotAuthenticated

See Chapter 6 Faults in Site Recovery Manager for more details.

RemoveResourcePoolMapping

The `removeResourcePoolMapping` method removes a resource pool mapping. The method does not check whether the pools in the mapping exists on the protected and recovery sites.

Synopsis

```
void removeResourcePoolMapping(ResourcePool primaryPool)
```

The `primaryPool` parameter specifies the resource pool on the primary site whose mapping must be deleted.

Faults

- ConnectionDownFault
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

AddTestNetworkMapping

The `addTestNetworkMapping` method adds a test network mapping between a network on the secondary site and a network on the secondary site that is used for testing.

Synopsis

```
void addTestNetworkMapping(@readonly Network secondaryNetwork,
        @readonly Network destinationTestNetwork)
```

secondaryNetwork

Specifies a network on the remote recovery site.

destinationTestNetwork

Specifies a test network on the remote recovery site.

Faults

- InvalidSecondaryNetwork
- UnknownSecondaryNetwork

- ConnectionDownFault
- RuntimeFault
- RemoteSiteNotAuthenticated

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

RemoveTestNetworkMapping

The `removeTestNetworkMapping` method removes the test network mapping. The method does not check whether the networks in the mapping exist on the secondary site. You can use the method to remove broken mappings.

Synopsis

```
void removeTestNetworkMapping(vim.Network secondaryNetwork)
```

The `secondaryNetwork` parameter specifies the network on the secondary site whose mapping must be removed.

Faults

- ConnectionDownFault
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

Protection

This section covers the Site Recovery Manager API methods for protection groups and virtual machine replication.

In SRM 5.8, new APIs appeared to create protection groups, assign them to recovery plans, and protect virtual machines using array-based or host-based replication. The new APIs provide three types of functionality for vSphere disaster recovery operations:

1 Infrastructure
   - workflows to create protection groups
   - workflows to create inventory mappings between matching objects
   - workflows to add protection groups to recovery plans
2 Virtual machine (VM) protection
   - workflows to protect VMs using a pre-configured array-based protection group
   - workflows to protect VMs using a pre-configured host-replicated protection group

3 Virtual machine (VM) recovery settings
   - recovery priority
   - per-VM callouts
   - final power state

ListProtectionGroups

This method lists the configured protection groups.

Synopsis

`SrmProtectionGroup[] listProtectionGroups( )`

`SrmProtectionGroup[]` is an array of managed object references to all the `SrmProtectionGroup` managed objects that are currently configured.

Faults

- RuntimeFault

Example: Method to list protection groups

```
SrmProtectionGroup[] = _service.ListProtectionGroups(_svcRef);
```

ListInventoryMappings

This method returns the configured inventory mappings. You establish placeholder datastores as described in the VMware Site Recovery Manager Documentation. You establish inventory mappings using the `AddFolderMapping`, `AddNetworkMapping`, and `AddResourcePoolMapping` methods documented in this manual, or using procedures described in the VMware Site Recovery Manager Documentation.

Synopsis

`Protection.InventoryMappingInfo listInventoryMappings( )`

`inventoryMappingInfo` is a list of inventory mappings from the protected site to the recovery site:

- `folders` is a list of mapped VirtualMachine Folders
- `networks` is a list of mapped virtual machine Networks and dvPortgroups
- `pools` is a list of mapped Resource Pools
- `testNetworks` is a list of mapped networks to test networks.

Faults

- RuntimeFault

Example: Method to list inventory mappings

```
inventoryMappingInfo = _service.ListInventoryMappings(_svcRef);
```

ListReplicatedDatastores

This method queries and lists replicated but unprotected datastores. A datastore is replicated if it contains any virtual machines in a protection group. The method is deprecated.

Synopsis

```
vim.Datastore[] listReplicatedDatastores()
```

`Datastore[]` is a list of replicated datastores on this site that can be used to create new protection groups.

Faults

- RuntimeFault

Example: Method to list replicated datastores

```
Datastore[] = _service.ListReplicatedDatastores(_svcRef);
```

GetProtectionGroupRootFolder

Returns a reference to the top-level container for protection groups.

Synopsis

`ProtectionGroupFolder getProtectionGroupRootFolder( )`

`ProtectionGroupFolder` – the top-level folder for protection groups.

Faults

- RuntimeFault

ListUnassignedReplicatedDatastores

Gets a list of replicated datastores that can be used to create new protection groups.

Synopsis

```
vim.Datastore[] listUnassignedReplicatedDatastores( )
```

`Datastore[]` is a list of all datastores on this site that are replicated but not currently protected by Site Recovery Manager.

Faults

- RuntimeFault

ProtectionListProtectedDatastores

Get a list of the replicated datastores that are protected by Site Recovery Manager.

Synopsis

```
vim.Datastore[] ProtectionListProtectedDatastores( )
```

`Datastore[]` isa list of all datastores on this site that are replicated and protected by Site Recovery Manager.

Faults

- RuntimeFault

ListUnassignedReplicatedVms

Gets a list of replicated VMs that are currently not assigned to a Site Recovery Manager protection group.

Synopsis

```
vim.VirtualMachine[] listUnassignedReplicatedVms(String replicationType)
```

`replicationType` is an enumeration defined in `SrmProtectionGroup`. Valid values are `san` for ABR replicated virtual machines, and `vr` for HBR replicated virtual machines.

`VirtualMachine[]` is a list of replicated virtual machines that are suitable for protection by Site Recovery Manager.

Faults

- RuntimeFault
- InvalidArgument

See Chapter 6 Faults in Site Recovery Manager for more details.

ProtectionListProtectedVms

Get a list of virtual machines that are protected by the Site Recovery Manager.

Synopsis

```
vim.VirtualMachine[] ProtectionListProtectedVms( )
```

`VirtualMachine[]` is a list of protected virtual machines.

Faults

- RuntimeFault

CreateAbrProtectionGroup

Create a new storage array based ProtectionGroup using the provided datastores This method does not automatically protect VMs on the storage array. Programs must call `ProtectVms()` separately for VMs on the storage array to be protected.

Synopsis

`CreateProtectionGroupTask createAbrProtectionGroup(Folder location, String name,@optional String description, vim.Datastore[] datastores)`

Parameter	Description
`location`	folder in which to create the protection group
`name`	the name of the protection group
`description`	an optional description of the protection group
`datastores`	array of datastores to add to the new protection group

Returns `CreateProtectionGroupTask` to monitor the asynchronous operation of this method.

Faults

- InvalidType
- InvalidArgument
- ReplicationProviderFault
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: Example for CreateAbrProtectionGroup

```
MoRef ProtectionGroup
CreateAbrProtectionGroup(_this, MoRef protectionFolder, String nameAbrProtectionGroup,
@optional String descriptionAbrProtectionGroup, MoRef datastoreRef)
/* _this is a managed object reference to an SrmProtection object. */
```

The following exceptions are presented by the `CreateProtectionGroupTask` instance that is returned by the `CreateAbrProtectionGroup` and `CreateHbrProtectionGroup` methods:

- `ConnectionDownFault` if the other site involved in the operation could not be contacted.
- `DuplicateName` if a group with this name already exists.
- `StringArgumentTooLong` if the size of either name or description in the settings parameter is too long.

CreateHbrProtectionGroup

Create a host based replication (vSphere replication) protection group using the provided VMs. This method does not automatically protect VMs on the storage array. Programs must call `ProtectVms()` separately for VMs on the storage array to be protected.

Synopsis

`CreateProtectionGroupTask createHbrProtectionGroup(Folder location, String name,@optional String description, vim.VirtualMachine[] vms)`

Parameter	Description
location	folder in which to create the protection group
name	the name of the protection group
description	an optional description of the protection group
vms	virtual machines to associate with the new protection group

Returns CreateProtectionGroupTask to monitor the asynchronous operation of this method.

Faults

- InvalidType
- ReplicationProviderFault
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: Example for CreateHbrProtectionGroup

```
MoRef ProtectionGroup
CreateHbrProtectionGroup(_this, MoRef protectionFolder, String nameHbrProtectionGroup,
@optional String descriptionHbrProtectionGroup, MoRef datastoreRef)
_this is a managed object reference to an SrmProtection object.
```

Protection Group Folder

This section presents methods to navigate folder hierarchy and retrieve specific protection groups.

ListChildProtectionGroupFolders

Returns the child Protection Group Folders located within this folder.

Synopsis

`ProtectionGroupFolder[] listChildProtectionGroupFolders( )`

`ProtectionGroupFolder[]` is the array of Protection Group Folders within this folder. Protection Group Folders for which the current session does not have the `System.View` privilege are removed from the result set.

Faults

- RuntimeFault

ListChildProtectionGroups

Returns the child Protection Groups located within this folder.

Synopsis

```
ProtectionGroup[] listChildProtectionGroups( )
```

`ProtectionGroup[]` is the array of Protection Groups within this folder. Protection Groups for which the current session does not have the `System.View` privilege are removed from the result set.

Faults

- RuntimeFault

GetProtectionGroup

Retrieves the protection group with the specified name, if any.

Synopsis

```
ProtectionGroup getProtectionGroup(String name)
```

`name` is the name of the protection group.

Returns `ProtectionGroup` is the specific Protection Group with the given name.

Faults

- ProtectionGroupNotFound
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

GetName

Retrieves the name of this folder object, given existence of a `ProtectionGroupFolder` or `RecoveryPlanFolder`.

Synopsis

```
String getName( )
```

Returns the name of this folder object.

Faults

- RuntimeFault

Example: Example for GetName

```
String entityName
GetName(MoRef _folderRef)
```

GetParentFolder

Gets a reference to the parent folder. `Folder` extends and is inherited by Site Recovery Manager from the vSphere API, where it is a managed object acting as a container to store and organize inventory objects, such as protection groups and recovery plans.

Synopsis

```
Folder getParentFolder( )
```

Returns the parent `Folder` as a managed object. This value is null for the root object.

Faults

- RuntimeFault

Create Protection Group Task

This chapter presents methods to track progress and completion of create protection group calls.

IsCreateProtectionGroupComplete

A task returned by `GetNewProtectionGroup` fills in the protection group result with the final status. This function checks completeness of the operation. To get the result, see `GetCreateProtectionGroupResult`.

Synopsis

```
Boolean IsCreateProtectionGroupComplete( )
```

True if this task has been completed.

Faults

- RuntimeFault

GetCreateProtectionGroupResult

A task returned by `GetNewProtectionGroup` fills in the protection group result with the final status of the operation. This function gets the `TaskInfo` object containing the detailed results. To check completeness, see`IsCreateProtectionGroupComplete`.

Synopsis

```
TaskInfo GetCreateProtectionGroupResult( )
```

Returns the details results of this task.

Faults

- RuntimeFault

GetNewProtectionGroup

After calling `CreateAbrProtectionGroup` or `CreateHbrProtectionGroup` to create a protection group, this call makes a new one and fills in the protection group with the final result of the operation. To get the task results, see `GetCreateProtectionGroupResult`. To check status, see `IsCreateProtectionGroupComplete`.

Synopsis

```
ProtectionGroup GetNewProtectionGroup( )
```

Returns the newly created `ProtectionGroup`.

Faults

- RuntimeFault
- InvalidState

See Chapter 6 Faults in Site Recovery Manager for more details.

Protection Group

For array based replication, Site Recovery Manager organizes datastore groups to collect files associated with protected virtual machines. You configure array based replication by associating datastore groups with protection groups. All virtual machines in a datastore group replicate files together, and all virtual machines recover together.

You configure host based replication (vSphere replication) for one virtual machine by associating it with a protection group, or you can configure multiple virtual machines by associating their folder or datacenter with a protection group. You use methods AssociateVms and UnassociateVms with host based replication, but not with array based replication.

GetInfo

This method retrieves basic information about the specified protection group. To get an `SrmProtectionGroup` managed object reference, see `ListProtectionGroups`.

Synopsis

```
ProtectionGroup.Info getInfo( )
```

`ProtectionGroup.Info` is information about the protection group:

- `description` is protection group description.
- `name` is protection group name
- `type` is either `san` for array based replication, or `vr` for vSphere replication.

Faults

- RuntimeFault

Example: Example for GetInfo

```
ptGrpInfo = _service.GetInfo(_svcPtGrp);
```

ProtectionGroupGetParentFolder

Given a protection group, gets the parent folder.

Synopsis

`ProtectionGroupFolder ProtectionGroupGetParentFolder( )`

`ProtectionGroupFolder` – extends `Folder`; can hold `ProtectionGroup` and `ProtectionGroupFolder`.

Faults

- RuntimeFault

GetPeer

Given an `SrmProtectionGroup` on the local site, this method retrieves the `SrmProtectionGroup` at the peer site.

Synopsis

`ProtectionGroup.Peer getPeer( )`

`ProtectionGroup.Peer` – the peer protection group from the remote site.

Faults

- RuntimeFault

Example: Example for GetPeer

```
peerPtGrp = _service.GetPeer(_svcPtGrp);
```

ListProtectedVms

Retrieves the list of virtual machines currently protected in the specified protection group, with information about their placeholder VM and protection state.

Synopsis

`ProtectionGroup.ProtectedVm[] listProtectedVms( )`

`ProtectedVm[]` is an array of ProtectedVm data objects with the following fields:

Field	Description
`faults`	any faults associated with this protected virtual machine
`needsConfiguration`	the protected virtual machine needs to be configured or repaired
`peerProtectedVm`	the protected virtual machine identifier on the remote site

Field	Description
peerState	the protection state on the remote site
protectedVm	the protected virtual machine identifier on the local site
state	the protection state of this particular virtual machine
vm	the locally protected virtual machine (this reference is valid after reprotect or revert operations)

Faults

- RuntimeFault

Example: Example for ListProtectedVms

```
protectedVm[] = _service.ListProtectedVms(_svcPtGrp);
```

ListProtectedDatastores

This method retrieves the list of datastores that are protected by the specified protection group. A datastore can be a VMFS volume, a NAS directory, or a local file system path.

Synopsis

`vim.Datastore[] listProtectedDatastores( )`

`Returns Datastore[]` is an array of all Datastore objects protected by this protection group.

Faults

- vmodl.fault.NotSupported if this protection group is not a SAN group.
- RuntimeFault

Example: Example for ListProtectedDatastores

```
datastore[] = _service.ListProtectedDatastores(_svcPtGrp);
```

ListAssociatedVms

This method lists the virtual machines currently associated with a specified vSphere Replication protection group. For the method to get a list of protection groups, see `ListProtectionGroups`.

Synopsis

`vim.VirtualMachine[] listAssociatedVms( )`

`VirtualMachine[]` is an array listing the associated virtual machines.

Faults

- vmodl.fault.NotSupported, if this protection group is not a VR group.

- RuntimeFault

Example: Example for ListAssociatedVms

```
VirtualMachine[] = _service.ListAssociatedVms(_svcPtGrp);
```

GetProtectionState

Gets current state of the specified protection group. Not to be confused with `GetProtectionStatus` which returns a virtual machine's (un)protect status, not the state of an entire protection group.

Synopsis

`ProtectionGroup.ProtectionState getProtectionState( )`

`ProtectionState` is an enumeration for the protection group state:

Fields	Description
`failedOver`	the protection group has been failed over to the remote site
`partiallyRecovered`	the protection group is partially recovered
`ready`	the protection group is in a ready state
`recovered`	the protection group has been recovered
`recovering`	the protection group is in the process of being recovered
`shadowing`	this protection group is shadowing the remote site group that is in ready state
`testing`	the protection group is currently being tested

Faults

- RuntimeFault

Example: Example for GetProtectionState

```
protState = _service.GetProtectionState(_svcPtGrp);
```

ProtectionGroupListRecoveryPlans

This method retrieves a list of all the recovery plans that this protection group is a member of.

Synopsis

`RecoveryPlan[] ProtectionGroupListRecoveryPlans( )`

`RecoveryPlan[]` is an array of all the Recovery Plans that this protection group belongs to.

Fault

- RuntimeFault

Example: Example for ProtectionGroupListRecoveryPlans

```
plans[] = _service.ProtectionGroupListRecoveryPlans(_svcPtGrp);
```

ProtectionGroupQueryVmProtection

Determine whether the specified virtual machines are currently protected, or can be protected. To protect a Virtual Machine, its folder, resource pool, and network must be mapped from the protected site to the recovery site. To get a list of currently configured mappings, see `ListInventoryMappings`. You can also query replicated datastores with `ListReplicatedDatastores`.

Synopsis

`ProtectionGroup.VmProtectionInfo[]`

`ProtectionGroupQueryVmProtection(vim.VirtualMachine[] vms)`

`vms[]` is an array of managed object references to VirtualMachine objects.

`VmProtectionInfo[]`is an array of VmProtectionInfo data objects with the following fields:

Fields	Description
`faults`	any faults encountered while processing queryVmProtection for this virtual machine
`peerProtectedVm`	the protected virtual machine identifier on the remote site
`protectedVm`	the protected virtual machine identifier on the local site
`protectionGroup`	the group this virtual machine is a member of, if it is protected
`protectionGroupName`	the name of this virtual machine’s protection group, if it is protected
`recoveryPlanNames`	the name(s) of any recovery plans the virtual machine will be recovered in
`recoveryPlans`	any recovery plans the virtual machine will be recovered in
`status`	the current protection status of the virtual machine
`vm`	the virtual machine for which protection status is being returned

Faults

- RuntimeFault

Example: Example for ProtectionGroupQueryVmProtection

```
protectInfo[] = _service.ProtectionGroupQueryVmProtection(_svcPtGrp, vms[]);
```

ProtectVms

This method adds virtual machines to a protection group. With array based replication, the protection group is determined by datastore location of the virtual machines. With host based replication (vSphere replication), you can use the `AssociateVms` method to place virtual machines into a protection group. To

protect a Virtual Machine, its folder, resource pool, and network must be mapped from the protected site to the recovery site. To get a list of currently configured mappings, see `ListInventoryMappings`.

Synopsis

`ProtectionTask protectVms(ProtectionGroup.VmProtectionSpec[] vms)`

`vms[]` is an array of managed object references to the Virtual Machine objects for protection.

`ProtectionTask` is the task object to monitor for status of the requested virtual machines.

Faults

- RuntimeFault

Example: Example for ProtectVms

```
SrmProtectionTaskRef = _service.ProtectVms(_svcRef, vm[] );
```

UnprotectVms

This method removes virtual machines from their protection group. With array based replication, the protection group is determined by datastore location of the virtual machines. With vSphere Replication, you must also `UnassociateVms` from the protection group.

Synopsis

`ProtectionTask unprotectVms(vim.VirtualMachine[] vms)`

`vms[]` is an array Virtual Machine objects not to protect.

`ProtectionTask` is the task object to monitor for status of the requested virtual machines.

Faults

- InvalidState, if a specified VM was not being protected.
- RuntimeFault

Example: Example for UnprotectVms

```
SrmProtectionTaskRef = _service.UnprotectVms(_svcRef, vm[] );
```

AssociateVms

This method associates one or more virtual machines with a vSphere Replication (VR) protection group. Before you can protect a virtual machine, it must first be associated with a protection group.

Synopsis

`void associateVms(vim.VirtualMachine[] vms)`

`vms[]` is an array of Virtual Machine objects to associate with.

Faults

- vmodl.fault.NotSupported, if this protection group is not a VR group.
- RuntimeFault
- InvalidState, if a specified VM was already associated with another group.
- vim.fault.ConcurrentAccess, if another operation has modified the object and the change version no longer matches.

Example: Example for AssociateVms

```
void _service.AssociateVms(_svcPtGrp, VirtualMachine[] );
```

UnassociateVms

This method removes the association of one or more virtual machines from a specified vSphere Replication (VR) protection group. Once a virtual machine is unassociated, it can no longer be protected.

Synopsis

`void unassociateVms(vim.VirtualMachine[] vms)`

`vms[]` is an array of Virtual Machine objects to disassociate from.

Faults

- vmodl.fault.NotSupported, if this protection group is not a VR group.
- RuntimeFault
- InvalidState, if a specified VM was already associated with another group.
- vim.fault.ConcurrentAccess, if another operation has modified the object and the change version no longer matches.

Example: Example for UnassociateVms

```
void _service.UnssociateVms(_svcPtGrp, VirtualMachine[] );
```

RemoveProtectionGroup

The `removeProtectionGroup` method unprotects the VMs in the protection group and deletes the group.

Synopsis

```
RemoveProtectionGroupTask removeProtectionGroup(ProtectionGroup group)
```

RemoveProtectionGroupTask is a Task object that contains information about the status of the operation. Site Recovery Manager Server retains the object for 30 minutes after the task finishes.

The `group` parameter specifies the protection group that must be deleted.

Faults

- ProtectionGroupNotEmpty
- ConnectionDownFault
- ReplicationProviderFault
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

CheckConfigured

The `checkConfigured` method checks the protection group for VMs that are not configured, configuration issues, and protected VMs that must be configured.

Synopsis

```
boolean checkConfigured()
```

The method returns true if the protection group is configured and you can use the group.

Faults

- InvalidState is thrown if the group is not on the protected site and cannot get information about the remote object.
- vmodl.fault.NotSupported is thrown if the group is not a VMware vSphere or a VMware Virtual SAN group.
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

ProtectionGroupGetOperationalLocation

The `ProtectionGroupGetOperationalLocation` method returns the effective location of the protection group for the purposes of determining when various operations should be run.

Synopsis

```
String ProtectionGroupGetOperationalLocation()
```

Faults

RuntimeFault

addDatastores

Adds datastores to the protection group. Additionally, the virtual machines on these datastores can be protected by the protection group. This can be done by calling protectVms method from this interface.

Synopsis

```
addDatastores(@optional Datastore[] datastores)
```

`datastores` is the list of datastores that will be added to the protection group.

Faults

- InvalidState
- InvalidArgument
- NotSupported
- SystemError
- vim.fault.ConcurrentAccess

See Chapter 6 Faults in Site Recovery Manager for more details.

removeDatastores

Removes datastores from the protection group. Virtual machines on the removed datastores are no longer protected by the protection group.

Synopsis

```
removeDatastores(@optional Datastore[] datastores)
```

`datastores` is the list of datastores that will be added to the protection group.

Faults

- InvalidState
- InvalidArgument
- NotSupported
- SystemError
- vim.fault.ConcurrentAccess

See Chapter 6 Faults in Site Recovery Manager for more details.

Protection Task

A task returned by `ProtectVms` or `UnprotectVms` acquires the final status of an operation upon completion. While the task is running, partial results may be returned. Once the task has been completed, the object will be removed from the server after 30 minutes.

GetProtectionStatus

This method gets the virtual machine protection status after completion of `ProtectVms` or `UnprotectVms`.

Synopsis

`ProtectionGroup.VmProtectionInfo[] getProtectionStatus()`

`VmProtectionInfo[]` – the completed protection status of VMs that were requested to be protected or unprotected.

The VmProtectionInfo.ProtectionStatus has the following fields:

Fields	Description
`canBeProtected`	the VM is able to be protected, but is not currently
`canNotBeProtected`	the VM is not able to be protected
`isProtected`	the VM is already protected
`needsConfiguration`	the VM must be configured or repaired before it may be protected. Please check the faults property for information about any additional prerequisites.

Faults

- RuntimeFault

Example: Example for GetProtectionStatus

```
protectionInfo[] = _service.GetProtectionStatus(_svcRef);
```

GetTasks

This method retrieves task information from the vCenter Server after a `ProtectVms` or `UnprotectVms` request, which both take some time to complete.

Synopsis

`ProtectionTask.VmTask[] getTasks()`

`VmTask[]` is an array of monitorable task information keyed by Virtual Machine, containing:

- `task` – managed object reference to a task on the Site Recovery Manager server.
- `vm` – managed object reference to a VirtualMachine.

Faults

- RuntimeFault

Example: Example for GetTasks

```
SrmProtectionTaskVmTask[] = _service.GetTasks(_svcRef);
```

IsComplete

This method checks whether the protection task has completed.

Synopsis

```
boolean isComplete( )
```

Returns true if the task has completed, false if not.

Faults

- RuntimeFault

Example: Example for IsComplete

```
isDone = _service.IsComplete(_svcRef);
```

GetResult

This method gets detailed results of the completed protection task.

Synopsis

```
vim.TaskInfo[] getResult( )
```

`TaskInfo[]` is a data object with 22 task properties set during task execution. For details see `TaskInfo` in the `API Reference Guide`.

Faults

- RuntimeFault

Example: Example for GetResult

```
TaskInfoObj[] = _service.GetResult(_svcRef);
```

Recovery

This section covers the Site Recovery Manager API methods for recovery plans and the next reprotection. You can set the recovery point objective (RPO), a desired time period for rerunning replication to avoid data loss, using a slider in the vSphere Client. This is also where you can configure quiescing of guest OS disk. You can also set the recovery priority of virtual machines as part of running a recovery plan.

ListPlans

This method retrieves all the recovery plans for this Site Recovery Manager server. Once you have a list of recovery plans, you can retrieve information about each plan.

Synopsis

`RecoveryPlan[] listPlans()`

`RecoveryPlan[]` is a list of Recovery Plans, including plan information, peer recovery plan, recovery mode, recovery plan location, recovery prompt, and recovery state.

Faults

- RuntimeFault

GetHistory

This method retrieves the history of a given recovery plan.

Synopsis

`RecoveryPlanHistory getHistory(RecoveryPlan plan)`

`plan` is the Recovery Plan of interest.

`RecoveryPlanHistory` is the history of the given Recovery Plan.

Faults

- RecoveryPlanNotFound
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: Example for GetHistory

```
history = _service.GetHistory(_srm.recovery, plan);
```

GetRecoveryPlanRootFolder

Gets a reference to the top level container (the root folder) for recovery plans.

Synopsis

`RecoveryPlanFolder getRecoveryPlanRootFolder()`

`RecoveryPlanFolder` – a Site Recovery Manager folder that holds Recovery Plans and Recovery Plan Folders.

Faults

- RuntimeFault

CreateRecoveryPlan

The `createRecoveryPlan` method creates a recovery plan. You call the method by passing the name and folder of the plan, and the protection group that must be included in the plan.

Synopsis

```
CreateRecoveryPlanTask createRecoveryPlan(
        String name,
        drextapi.Folder folder,
        ProtectionGroup[] groups,
        @optional String description,
        @optional TestNetworkMapping[] mapping)
```

Parameter	Description
`name`	Specifies the name of the plan. Must be unique in the parent folder.
`folder`	Specifies the folder where the plan is created.
`groups`	Specifies the protection groups that are added to the recovery plan.
`description`	Optional parameter. Specifies the recovery plan description.
`mapping`	Optional parameter. Specifies the test network mappings.

CreateRecoveryPlanTask is a Task object that contains information about the status of the operation. Site Recovery Manager Server retains the object for 30 minutes after the task finishes.

`IsCreateRecoveryPlanComplete` returns true if the task for creating a recovery plan is complete.

`GetCreateRecoveryPlanFailure` returns the failure during the operation for creating a recovery plan. If you pass a name of a plan that exists, the failure is DuplicateNames.

If the operation is successful, the `GetNewRecoveryPlan` returns the created recovery plan.

Faults

- DirectionError
- InvalidArgument
- ProtectionGroupNotFound
- RemoteSiteNotEnabled
- StringArgumentTooLong
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

DeleteRecoveryPlan

The `deleteRecoveryPlan` method deletes the recovery plan that is passed as parameter.

Synopsis

```
DeleteRecoveryPlanTask deleteRecoveryPlan(RecoveryPlan plan)
```

DeleteRecoveryPlanTask is a Task object that contains information about the status of the operation. Site Recovery Manager Server retains the object for 30 minutes after the task finishes.

The `plan` parameter specifies the recovery plan that must be deleted

The `IsDeleteRecoveryPlanComplete` returns true if the task is complete.

The `GetDeleteRecovryPlanFailure` returns the failure that occurs during the delete operation. If the plan or its peer is not in a valid state, the task fails with InvalidState.

Faults

RuntimeFault

Recovery Plan Folder

This section presents methods to traverse the folder hierarchy for Recovery Plans.

ListChildRecoveryPlanFolders

Returns the child Recovery Plan Folders located in this folder.

Synopsis

`RecoveryPlanFolder[] listChildRecoveryPlanFolders( )`

`RecoveryPlanFolder[]` is the array of sub-folders within this folder. If the current session does not have the `System.View` privilege for a `RecoveryPlanFolder`, it is removed from the result set.

Faults

- RuntimeFault

Example: Example for ListChildRecoveryPlanFolders

```
MoRef RecoveryPlanFolder[]
ListChildRecoveryPlanFolders(MoRef RecoveryPlanFolder)
```

ListChildRecoveryPlans

Returns an array of `RecoveryPlan` objects located within this folder.

Synopsis

`RecoveryPlan[] listChildRecoveryPlans( )`

`RecoveryPlan[]` is the array of Recovery Plans within this folder. If the current session does not have the `System.View` privilege for a `RecoveryPlan`, it is removed from the result set.

Faults

- RuntimeFault

Example: Example for ListChildRecoveryPlans

```
MoRef SrmRecoveryPlan[]
ListChildRecoveryPlans(MoRef RecoveryPlanFolder)
```

GetRecoveryPlan

Retrieves a specific recovery plan.

Synopsis

`RecoveryPlan getRecoveryPlan(String name)`

`name` is the name of a Recovery Plan.

`RecoveryPlan` is a Recovery Plan, which includes plan information, peer recovery plan, recovery mode, recovery plan location, recovery prompt, and recovery state.

Faults

- RecoveryPlanNotFound
- RuntimeFault

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: Example for GetRecoveryPlan

```
MoRef RecoveryPlan
GetRecoveryPlans(MoRef RecoveryPlanFolder, String planName);
```

GetName

Retrieves the name of this folder object, given existence of a `ProtectionGroupFolder` or `RecoveryPlanFolder`.

Synopsis

`String getName( )`

Returns the name of this folder object.

Faults

- RuntimeFault

Example: Example for GetName

```
String entityName
GetName(MoRef _folderRef)
```

GetParentFolder

Gets a reference to the parent folder. `Folder` extends and is inherited by Site Recovery Manager from the vSphere API, where it is a managed object acting as a container to store and organize inventory objects, such as protection groups and recovery plans.

Synopsis

```
Folder getParentFolder( )
```

Returns the parent `Folder` as a managed object. This value is null for the root object.

Faults

- RuntimeFault

Recovery Plan

This section covers the interfaces to recovery plans.

RecoveryPlanGetInfo

This method retrieves status information about a given recovery plan, including the name of the recovery plan and its current state.

Synopsis

`RecoveryPlan.Info` `RecoveryPlanGetInfo( )`

`RecoveryPlan.Info` is a data object that describes details of this recovery plan, including:

- `name` is the name of this recovery plan.
- `description` is a description of this recovery plan.
- `protectionGroups[]` is an array of protection groups that will be recovered as part of this plan.
- `state` – the state of this recovery plan, enumerated as:
  - `cancelling` – recovery plan is in the process of cancelling
  - `error` – recovery plan has errors
  - `failedOver` – recovery plan has failed over
  - `needsCleanup` – need to cleanup a test run
  - `needsFailover` – need to re-run recovery (failover)
  - `needsReprotect` – need to re-run reprotect
  - `needsRollback` – need to re-run rollback
  - `prompting` – recovery plan is running, but requires user-interaction before it may continue
  - `protecting` – recovery plan is protecting to remote site, run peer recovery plan on remote site

- `ready` – recovery plan is not in a running state and may be run
- `running` – recovery plan is currently running

Faults

- RuntimeFault

Example: Examples for RecoveryPlanGetInfo

```
status = _service.RecoveryPlanGetInfo(_srm.plan);
```

The sample C# and Java code below combines `ListPlans` with `RecoveryPlanGetInfo` to retrieve a specified plan.

C# sample code for recovery plan

```
ManagedObjectReference[] plans = _service.ListPlans(_sic.recovery);
if (plans != null && plans.Length > 0)
{
for (int i = 0; i < plans.Length; ++i)
{ SrmRecoveryPlanInfo info = _service.RecoveryPlanGetInfo(plans[i]);
Console.WriteLine("RecoveryPlan : " + info.name);
if (info.name.Equals(planName))
{
Console.Write(" RecoveryPlan state : ");
Console.WriteLine(info.state);
}
}
}
```

Java sample code for recovery plan

```
private static void listPlans() throws Exception { List<ManagedObjectReference> plans =
srmPort.listPlans(serviceContent.getRecovery());
if (plans != null && plans.size() > 0)
{
for (int i = 0; i < plans.size(); ++i)
{ SrmRecoveryPlanInfo info = srmPort.recoveryPlanGetInfo(plans.get(i));
System.out.println("RecoveryPlan : " + info.getName()); if (info.getName().equals(planName))
    {
    System.out.print(" RecoveryPlan state : "); System.out.println(info.getState());
 }
}
}
}
```

RecoveryPlanGetPeer

This method retrieves a recovery plan peer, which is the plan at the paired site rather than at the local site.

Synopsis

`RecoveryPlan.Peer RecoveryPlanGetPeer()`

`RecoveryPlan.Peer` is the peer recovery plan at the paired site.

- `plan` references to the `SrmRecoveryPlanmanaged` object.
- `state` is the same enumeration as for `RecoveryPlanGetInfo`.

Faults

- RuntimeFault

Example: Example for RecoveryPlanGetPeer

```
peer = _service.RecoveryPlanGetPeer(_srm.plan);
```

Start

This method starts or reconfigures the given recovery plan, or tests and cleans it up, depending on the mode specified. This operation requires one of these privileges depending on recovery mode, `VcDr.RecoveryProfile.com.vmware.vcDr.Failover` for a real recovery and `VcDr.RecoveryProfile.com.vmware.vcDr.Run` for a test recovery.

Synopsis

`void start(RecoveryPlan.RecoveryMode mode, @version5 @optional RecoveryOptions options)`

- `mode` – one of the following recovery modes:
  - test – run a test failover to the peer (recovery) site, without halting the local (protected) site.
  - cleanupTest – after testing a recovery plan, cleans up all effects of the test operation.
  - failover – move to the peer (recovery) site. When all groups are moved the recovery plan is complete.
  - reprotect – the peer site becomes the protected site, and the local site becomes the recovery site.
- `options` specifies the recovery options. The `RecoveryOptions.syncData` is a boolean parameter and indicates whether to replicate the recent changes to the recovery site.

Faults

- InvalidState, if the recovery plan is not in the ready state.
- IvalidArgument, if the recovery mode is not valid.
- RuntimeFault

Example: Example for Start

```
void _service.Start(_srm.plan, mode);
```

Cancel

This method cancels this recovery plan. It can take some time to cancel a recovery plan depending on its state. This operation requires one of these privileges depending on recovery mode, `VcDr.RecoveryProfile.com.vmware.vcDr.Failover` for a real recovery and `VcDr.RecoveryProfile.com.vmware.vcDr.Run` for a test recovery.

Synopsis

```
void cancel()
```

Faults

- InvalidState, if the recovery plan cannot be canceled.
- RuntimeFault

Example: Example for Cancel

```
void _service.Cancel(_srm.plan);
```

ListPrompts

This method lists the current prompts that are waiting on user input. Prompts appear in the order in which virtual machines are scheduled to power on. When a prompt step is reached, the recovery plan remains in a waiting state until the user answers the prompt or a program calls `AnswerPrompt`.

Synopsis

```
RecoveryPlan.RecoveryPrompt[] listPrompts()
```

`RecoveryPrompt[]` is an array of data objects containing the prompt and the key for responding to it.

Faults

- InvalidState, if the recovery plan is not running.
- RuntimeFault

Example: Example for ListPrompts

```
prompts[] = _service.ListPrompts(_srm.plan);
```

AnswerPrompt

This method answers the current prompt being displayed in a recovery plan. The operation requires one of these privileges depending on recovery mode, `VcDr.RecoveryProfile.com.vmware.vcDr.Failover` for a real recovery and `VcDr.RecoveryProfile.com.vmware.vcDr.Run` for a test recovery.

Synopsis

```
void answerPrompt(String key, boolean cancelVmRecovery, @optional String response)
```

`key` is a string with the key value from the recovery prompt.

`cancelVmRecovery` is true if you want to halt further processing on this virtual machine, false otherwise.

`response` is a response to the prompt that will be recorded.

Faults

- InvalidState, if the recovery plan is not running.
- PromptNotFound, if no prompt with that key exists.
- RuntimeFault

Example: Example for AnswerPrompt

```
void _service.AnswerPrompt(_srm.plan, key, cancel, response);
```

RecoveryPlanGetParentFolder

Gets the parent folder (or root) for a recovery plan.

Synopsis

`RecoveryPlanFolder RecoveryPlanGetParentFolder()`

`RecoveryPlanFolder` is a Site Recovery Manager folder that can hold Recovery Plans and Recovery Plan Folders.

Faults

- RuntimeFault

GetRecoverySettings

Gets the recovery settings for the specified virtual machine.

Synopsis

`RecoverySettings GetRecoverySettings(VirtualMachinevm )`

`vm` is the Virtual Machine whose Recovery Settings are to be retrieved.

`RecoverySettings` – the VM recovery settings for presentation in the user interface, including:

- `changeVersion` – change version control.
- `status` – recovery status.
- `recoveryPriority` – the recovery priority for this VM.
- `skipGuestShutdown` – configure the shutdown behavior for this workload during real failover to not attempt a guest shutdown, even if VMware Tools are enabled.
- `powerOffTimeoutSeconds` – configure the timeout for guest shutdown operations for this VM.
- `finalPowerState` – final power state for this VM after recovery.

- `localFaultToleranceState` – configure FT override setting for this VM when it will be failed back.
- `remoteFaultToleranceState` – configure an FT override setting for this VM after recovery.
- `powerOnTimeoutSeconds` – configure the timeout for VMware Tools to respond with a heartbeat.
- `powerOnDelaySeconds` – configure the fixed time delay after power-on operations for this workload.
- `prePowerOnCallouts` – before power-on Callouts (commands or prompts).
- `postPowerOnCallouts` – after power-on Callouts.
- `VmIpCustomization vmIpCustomizationData` - IP customization information.

Faults

- RuntimeFault
- RecoveryPlanNotFound
- VmNotFoundInRecoveryPlan

See Chapter 6 Faults in Site Recovery Manager for more details.

SetRecoverySettings

Updates the virtual machines' Recovery Settings. This method updates the specified virtual machine's Recovery Settings with values contained in the supplied Recovery Settings object. This class modifies the recovery settings available to the external API, `VmIpCustomization`. The `VmIpCustomization` API allows user to configure the IP address and corresponding DNS, WINS of the virtual machine, after the migration is complete. You can disable IP customization by setting `VmIpCustomization` to `nullptr` or by not setting `IpCustomizationSpecMapping` within `VmIpCustomization`.

Synopsis

`void setRecoverySettings(vim.VirtualMachine vm, RecoverySettings settings)`

`vm` is the Virtual Machine which Recovery Settings are to be updated.

`settings` is the Recovery Settings to update the VM.

`RecoverySettings` – the VM recovery settings for presentation in the user interface, including:

- `changeVersion` – change version control.
- `status` – recovery status.
- `recoveryPriority` – the recovery priority for this VM.
- `skipGuestShutdown` –configure the shutdown behavior for this workload during real failover to not attempt a guest shutdown, even if VMware Tools are enabled.
- `powerOffTimeoutSeconds` – configure the timeout for guest shutdown operations for this VM.
- `finalPowerState` – final power state for this VM after recovery.
- `localFaultToleranceState` – configure FT override setting for this VM when it will be failed back.

- `remoteFaultToleranceState` – configure an FT override setting for this VM after recovery.
- `powerOnTimeoutSeconds` – configure the timeout for VMware Tools to respond with a heartbeat.
- `powerOnDelaySeconds` – configure the fixed time delay after power-on operations for this workload.
- `prePowerOnCallouts` – before power-on Callouts (commands or prompts).
- `VmIpCustomization` – allows user to configure the IP address and corresponding DNS, WINS of the virtual machine, after the migration is complete.

`VmIpCustomization` contains the following data required for performing IP customization for a virtual machine, when recovering it to a given SRM site:

- `IpAddressInfo` - IP address definitions.
- `IpV4AddressInfo` - IPv4 address definitions.
- `IpV6AddressInfo` - Ipv6 address definitions.
- `IpV4AddressSpec` - IPv4 address specification. Contains either static or DHCP configuration.
- `@optional IpV4AddressInfo staticAddressInfo` - If this optional field is set, then this spec denotes a static IPv4 address configuration. Otherwise, an unset field denotes a DHCPv4 configuration.
- `IpV6AddressSpec` - IPv6 address specification. Contains either static or DHCP configuration.
- `@optional IpV6AddressInfo staticAddressInfo` - If this optional field is set, then this spec denotes a static IPv6 address configuration. Otherwise, an unset field denotes a DHCPv6 configuration.
- `enum NetBiosMode` - Used to configure NetBIOS on Windows systems. The values correspond directly to Microsoft constants for NetBIOS mode.
- `NicCustomizationSpec` - Contains IP customization info for a specific network adapter.
- `WindowsNicCustomizationSpec` - Contains Windows specific IP customization info for a specific network adapter for a virtual machine.

Faults

- `postPowerOnCallouts` – after power-on Callouts.
- RuntimeFault
- DependencyConflict
- InvalidArgument
- RecoveryPlanLocked
- RecoveryPlanNotFound
- VersionConflict
- VmNotFoundInRecoveryPlan

See Chapter 6 Faults in Site Recovery Manager for more details.

AddProtectionGroup

Adds a protection group to the recovery plan.

Synopsis

`void addProtectionGroup(ProtectionGroup group)`

`group` is the `ProtectionGroup` to be added.

Faults

- RuntimeFault
- VersionConflict
- RecoveryPlanLocked
- ProtectionGroupNotFound
- InvalidArgument
- NoPermission

See Chapter 6 Faults in Site Recovery Manager for more details.

GetRecoveryResult

Retrieves recovery results for a given run of this recovery plan. Use this method to get the key so subsequent methods can get recovery results history.

Synopsis

`RecoveryResult[] getRecoveryResult(int length)`

`length` is the maximum number of results to retrieve.

`RecoveryResult[]`is an array of recovery results for this recovery plan or its peer plan, including:

- `description` – summary of the plan at the time of this run
- `errorCount` – count of error-level faults that were generated by the operation
- `executionTimeInSeconds`– total execution time in seconds
- `key` – unique key for this recovery result, useful for subsequent methods
- `name` – the recovery plan's name at the time of this run
- `plan` – recovery plan that this result covers
- `resultState` – the result state, which is only the final state indicating completion or failure
- `runMode` – mode of recovery when plan was initiated (test, recovery, reprotect)
- `startTime`, `stopTime` – time when the recovery was started and when it completed or stopped
- `totalPausedTimeInSeconds` – total time the recovery plan was paused
- `warningCount` – count of warning-level faults that were generated by the operation

Faults

- RuntimeFault
- InvalidArgument

See Chapter 6 Faults in Site Recovery Manager for more details.

Example: Example for GetRecoveryResult

```
result[] = _service.GetRecoveryResult(_srm.history, length);
```

GetResultCount

Retrieves the total number of stored results. This include historical results from both the plan and its peer plan if the sites are connected.

Synopsis

```
int getResultCount()
```

Returns an integer count with the total number of history entries for this plan, and potentially its peer.

Faults

- RuntimeFault

Example: Example for GetResultCount

```
entries = _service.GetResultCount(_srm.history);
```

GetResultLength

Retrieves the length of the XML result document for the requested Recovery Result.

Synopsis

```
int getResultLength(int key)
```

`key` is the unique key for the plan history, from return value of the `GetRecoveryResult` method.

Returns an integer specifying the number of lines in the XML result file.

Faults

- RecoveryResultNotFound, if no result with that key exists.
- RuntimeFault

Example: Example for GetResultLength

```
length = _service.GetResultLength(_srm.history, key);
```

RetrieveStatus

Retrieves an XML representation of the specified historical run of the referenced recovery plan. This XML document is transmitted in chunks limited by the maximum length of a string in the transport layer. You specify what line to start at and how many lines to return.

Synopsis

`String[] retrieveStatus(int key, int offset, int maxLines)`

`key` is the unique key for the plan history, returned in RecoveryResult.key from getGetRecoveryResult.

`offset` is an integer specifying the starting line number in the XML file, beginning at 0,

`maxLines` is an integer specifying the maximum number of lines to retrieve.

Returns a string containing an XML representation of all recovery steps and their results.

Only after you have retrieved all the lines and assembled them do you have a valid XML document.

Faults

- InvalidArgument, if the offset exceeds the length of the document or if maxLines is not positive.
- RecoveryResultNotFound, if no result with that key exists.
- RuntimeFault

Example: Example for RetrieveStatus

```
*recoveryhistory = _service.RetrieveStatus(_srm.history, key, offset, lines);
```

AddTestNetworkMappingToRecoveryPlan

The `AddTestNetworkMappingToRecoveryPlan` method adds a test network mapping to a recovery plan.

Synopsis

```
void AddTestNetworkMappingToRecoveryPlan(
        vim.Network secondaryNetwork,
        vim.Network testNetwork)
```

secondaryNetwork

Specifies the network on the remote recovery site.

testNetwork

Specifies the test network on the remote recovery site.

If you call the method by passing a non existing network as a `secondaryNetwork` parameter, the method does not add the mapping to the recovery plan and does not throw a fault.

If you call the method by passing a non-existing test network as a `testNetwork` parameter, the method throws `ManagedObjectNotFound`.

Faults

- ConnectionDownFault
- NoPermission
- RecoveryPlanLocked
- RemoteSiteNotAuthenticated
- VersionConflict
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

RemoveTestNetworkMappingFromRecoveryPlan

The `RemoveTestNetworkMappingFromRecoveryPlan` method removes a test network mapping from a recovery plan.

Synopsis

```
void RemoveTestNetworkMappingFromRecoveryPlan(vim.Network secondaryNetwork)
```

The `secondaryNetwork` parameter specifies the secondary site network whose mapping must be removed.

Faults

- ConnectionDownFault
- NetworkNotFound
- NoPermission
- RecoveryPlanLocked
- RemoteSiteNotAuthenticated
- VersionConflict
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

RemoveProtectionGroupFromRecoveryPlan

The `RemoveProtectionGroupFromRecoveryPlan` method removes a protection group from a recovery plan.

Synopsis

```
void RemoveProtectionGroupFromRecoveryPlan(ProtectionGroup group)
```

The `group` parameter specifies the protection group that must be removed.

Faults

- NoPermission
- ProtectionGroupNotFound
- RecoveryPlanLocked
- VersionConflict
- RuntimeFault

For information about the faults that Site Recovery Manager throws, see Chapter 6 Faults in Site Recovery Manager.

RecoveryPlanGetLocation

The `RecoveryPlanGetLocation` method checks whether the recovery plan is hosted locally or on the paired site.

Synopsis

```
String RecoveryPlanGetLocation()
```

The method returns the localToRecoverySite, notLocalToRecoverySite, unknownLocationNoPgs, and unknownLocation Strings.

Returned Value	Description
localToRecoverySite	The recovery plan instance exists locally on the recovery site.
notLocalToRecoverySite	The recovery plan instance is a peer of the recovery site instance.
unknownLocationNoPgs	The recovery plan instance has no protection groups. A plan with no protection groups is not local to either site.

Faults

RuntimeFault

RecoveryPlanHasRunningTask

The `RecoveryPlanHasRunningTask` method checks whether there is a task that is associated with the recovery plan.

Synopsis

```
boolean RecoveryPlanHasRunningTask()
```

The method returns true if there is a task that is associated with the recovery plan.

Faults

RuntimeFault

Storage

This section shows a method that you can use to rescan storage.

DiscoverDevices

The `discoverDevices` method starts a loop through the array managers on the local and remote sites, and runs for each array pair. If the user does not have enough privileges, the array managers are skipped.

Synopsis

```
DiscoverDevicesTask discoverDevices()
```

The method returns a Task object that contains information about the status of the operation. Site Recovery Manager Server retains the object for 30 minutes after the task finishes.

`IsDiscoverDevicesTaskComplete` returns true if the `DiscoverDeviceTask` is complete.

`GetDiscoverDevicesTaskFailures` returns a list of failures that occurred while performing `DiscoverDevices` on array managers. The entire set of failures can be retrieved when the task is complete. If no failures occurred, the array is empty or null.

`DiscoverDevicesFailure.name` is a string that contains the array pair name.

`DiscoverDevicesFailure.fault` is an MethodFault object that indicates the failure.

Faults

RuntimeFault

QueryArrayManagers

The `queryArrayManagers` method returns a list of all the available array managers.

Synopsis

```
@optional ArrayManager[] queryArrayManagers()
```

`ArrayManager[]` is an array of available array managers.

ArrayManager

This section presents methods to interact with the SRM array managers.

ReadInfo

The `ReadInfo` method returns information specific to the `ArrayManager` instance.

Synopsys

```
ArrayManagerInfo readInfo()
```

`readInfo()` returns `ArrayManagerInfo` which contains information for the ArrayManager instance.

QueryReplicatedArrayPairs

The `QueryReplicatedArrayPairs` method returns a list of all the replicated array pairs in the `ArrayManager`.

Synopsis

```
@optional ReplicatedArrayPair[] queryReplicatedArrayPairs()
```

`ReplicatedArrayPair[]` is an array of the replicated array pairs in the `ArrayManager`.

ReplicatedArrayPair

This section presents methods to interact with the replicated array pairs.

QueryReplicatedRdms

The `QueryReplicatedRdms` method returns information about all the replicated RDMs in the `ReplicatedArrayPair`.

Synopsys

```
@optional ReplicatedRdmInfo[] queryReplicatedRdms()
```

`ReplicatedRdmInfo[]` is an array and it contains the following information about all the replicated RDMs in the `ReplicatedArrayPair`:

Table 4-1.

Value	Description
key	String. Unique key identifying this object. The value is constructed from virtual machine MoID and virtual device.
device	String. Storage device identifier.
deviceGroup	@optional boolean. Indicates if this is a stretched RDM device.
stretchedStorage	@optional boolean. Indicates if this is a stretched RDM device.
sitePreference	@optional boolean. For stretched RDM devices indicates whether this device has the site preference or not.
vim.VirtualMachine vm	Virtual machine to which the RDM is attached.
deviceKey	Int. Virtual device key of the attached RDM.
lunUuid	String. UUID of the RDM LUN.

Deprecated APIs

5

There are some remaining 1.0 APIs and some replacement 5.0 APIs.

Deprecated APIs

Table 5-1. Replaced APIs

SrmApi	Replacement 5.0 API
`ListRecoveryPlans`	`SrmRecovery.ListPlans`
`RecoveryPlanAnswerPrompt`	`SrmRecoveryPlan.AnswerPrompt`
`RecoveryPlanSettings`	`SrmRecoveryPlan.RecoveryPlanGetInfo`
`RecoveryPlanStart`, `RecoveryPlanCancel`	`SrmRecoveryPlan.Start`, `SrmRecoveryPlan.Cancel`
`GetFinalStatus`	`SrmRecoveryHistory.GetRecoveryResult`
`GetApiVersion`	no replacement

Faults in Site Recovery Manager 6

There are various faults thrown by Site Recovery Manager external APIs.

Faults thrown by Site Recovery Manager functions

Table 6-1. Faults thrown by Site Recovery Manager functions

Fault	Description
AlreadyExists	The name, key, or identifier of the element already exist in the collection.
AlreadyLoggedInFault	The session is already logged in, and Login was called again
ConnectionDownFault	The VMOMI connection to the remote server is down
ConnectionLimitReached	Thrown when the configured connection limit has been reached
DependencyConflict	UpdateVmSettings operation was attempted that might cause a dependency cycle
DirectionError	The direction of the recovery plan cannot be determined.
DuplicateName	Call is unable to determine which object to use due to name conflict
InsufficientLicensesFault	Thrown by a method that cannot acquire licenses for the object to create
InternalError	An internal error occurred that cannot be described by a more specific fault
InvalidArgument	Base class for invalid argument exceptions
InvalidLogin	Cannot complete login due to an incorrect user name or password
InvalidPrimaryFolder	Thrown for an attempt to create a primary site folder that cannot contain VMs
InvalidPrimaryNetwork	Invalid primary network specified for mapping, such as uplink DVPortgroup
InvalidSecondaryFolder	Thrown for an attempt to create a secondary site folder that cannot contain VMs
InvalidSecondaryNetwork	Invalid secondary network specified for mapping (such as uplink DVPortgroup)

Table 6-1. Faults thrown by Site Recovery Manager functions (continued)

Fault	Description
InvalidState	Base class for invalid state exceptions
InvalidTokenLifetime	SSO token is either expired or not yet valid. This exception is generally thrown if there is a time skew between the local clock and the SSO server. This exception is not supported from methods with version prior to 8. However it is translated to Vim::Fault::InvalidLogin::Exception.
MultipleFault	Multiple faults occur.
NetworkNotFound	Thrown if the test network mapping does not exist in the recovery plan.
NoPermission	Operation denied because of a privilege not held on a managed object
NotAuthenticated	Operation denied because the session has not successfully logged in
NotSupported	Thrown if a group is not array based.
PromptNotFound	Thrown when a RecoveryPrompt cannot be found
ProtectionGroupNotEmpty	Thrown after attempt to remove a protection group that contains protected VMs
ProtectionGroupNotFound	Thrown when an operation on protection group cannot find the protection group
RecoveryPlanLocked	An attempt was made to change a RecoveryPlan that is locked
RecoveryPlanNotFound	Thrown when the requested recovery plan was not found
RecoveryResultNotFound	Thrown when a RecoveryResult cannot be found
RemoteSiteNotAuthenticated	Thrown if the remote site is not authenticated.
RemoteSiteNotEnabled	An attempt was made to use a remote site that is not enabled
ReplicationProviderFault	Thrown when an unspecified error was returned from the replication provider
StringArgumentTooLong	Thrown when a string argument exceeds {maxSize} characters
SystemError	Thrown in case of internal SRM error.
UnknownPrimaryFolder	Secondary site tried operation on a folder that is nonexistent on primary site
UnknownPrimaryNetwork	Secondary site tried operation on a network that is nonexistent on primary site
UnknownPrimaryResourcePool	Secondary site tried operation on resource pool that is nonexistent on primary site
UnknownSecondaryFolder	Primary site tried operation on a folder that is nonexistent on secondary site
UnknownSecondaryNetwork	Primary site tried operation on a network that is nonexistent on secondary site
UnknownSecondaryResourcePool	Primary site tried operation on resource pool that is nonexistent on secondary site

Table 6-1. Faults thrown by Site Recovery Manager functions (continued)

Fault	Description
VersionConflict	Attempt to reconfigure with a changeVersion that does not match the current value
VmNotFoundInRecoveryPlan	Attempt to retrieve settings for virtual machine that does not exist in RecoveryPlan
vim.fault.ConcurrentAccess	Thrown if another operation has modified the object and the change version no longer matches.

SSL Certificates and SNMP Traps 7

This appendix contains information for the requirements and work with SSL certificates and Simple Network Management Protocol (SNMP) Traps.

This chapter includes the following topics:

SSL Certificates

The Site Recovery Manager Web service listens by default on port 9007. It uses SSL to encrypt communications between a client application and the server. The SSL certificate of the target server must reside on the client machine. To access the Web service programmatically, use its URN from a Web services client application, for example: `https://<FQDN.hostname.or.IP.Address>:9007`.

Get vCenter Server Certificate

The Site Recovery Manager API is a secure Web service running on the Site Recovery Manager Server. To develop client applications, you must obtain the VMware vCenter Server certificate, which is used by the Site Recovery Manager Server, and import it into the certificate store of the workstation where you develop client applications.

Procedure

1. From your development workstation, open Internet Explorer (IE).
2. Navigate to the vCenter Server using HTTPS protocol – `https://<servername>`.

   A Security Alert message displays a warning regarding the certificate's certifying authority.
3. Click **View Certificate**.
4. Click **Install Certificate** to launch the Certificate Import wizard. Keep the default settings and click **Next**.
5. Click **Finish**. A security warning message displays concerning the certificate's certifying authority.
6. Click **Yes**.

   A Certificate Import wizard "success" message displays.

7 Click **OK** to dismiss the success message.

   The Certificate Properties page becomes active again.

8 Click **OK** in the Certificate dialog box to continue to the server.

   The initial Security Alert message presented in step 2 becomes active again.

9 Click **Yes** in the Security Alert message to continue with the original HTTPS request.

   The server Welcome page displays. The certificate is now installed in the IE certificate cache.

What to do next

Now that you have the certificate, your next task depends on what programming language you use to develop your client applications.

For C# developers, you can continue setting up your development environment by following the instructions at “Setting Up for Microsoft C# Development” in the Developer’s Setup Guide located at VMware’s Web site developer support page under the vSphere Web Services SDK.

For Java developers, you must export the certificates from the IE cache to a local directory. Minimize the IE browser window, and export the certificates as detailed in the following procedure.

Export Cached Certificates to a Local Directory

For Java development in a Windows environment, you must export the certificate to a local directory.

Procedure

1 Create a directory for the certificate, using the name set in the various batch files for the vSphere Web Services SDK: `C:\VMware–Certs`.

2 From the IE Tools menu, select Internet Options to open the Internet Options properties page.

3 Click the **Content** tab to activate the content advisor.

4 Click **Certificates** to open the Certificate manager.

5 Click the **Trusted Root Certificate Authorities** tab to display the list of trusted certificates.

6 Scroll through the list of certificates to find the certificate. For the vCenter Server, the certificate name is VMware.

7 Click the certificate to select it.

8 Click **Export...** to launch the Certificate Export Wizard.

9 Click **Next** to continue. The Export File Format dialog displays.

10 Keep the defaults (“DER encoded binary X.509 (.CER)”) and click **Next** to continue.

   The File To Export dialog displays, enabling you to enter a unique name for the certificate.

11 Choose a filename and enter it, along with the complete path to the directory: `C:\VMware-Certs\<servername>.cer`

   If you do not enter the complete path, the certificate is stored in your Documents and Settings folder.

12 Click **Next** to continue with the export.

   A Completing the Certificate Export Wizard page displays, summarizing the information about the certificate.

13 Click **Finish** to complete the export.

   A Certificate Export Wizard “success” message displays.

14 Click **OK** to dismiss the success message.

15 Click **Close**.

16 Click **Cancel** to close the Internet Options properties page.

What to do next

For more information about setting up your Java development environment, see “Setting Up for Java Development” in the Developer’s Setup Guide located at the VMware Web site developer support page under the vSphere Web Services SDK.

About the Virtual Machine Keystore

A Java KeyStore (JKS) is a repository of security certificates – either authorization certificates or public key certificates – used for SSL encryption and related activities. The Java Development Kit (JDK) maintains a keystore in `jre/lib/security/cacerts`, and provides the `keytool` command to manipulate it.

The `VMKEYSTORE` environment variable specifies the path to the JKS. The `run.sh` and `run.bat` scripts both refer to it. If you use the `--ignorecert` argument to run Java samples, you must still set the `VMKEYSTORE` variable, but you can set it to any location, not the actual JKS location.

Sample paths, Windows and Linux:

```
VMKEYSTORE=C:\VMware-Certs\vmware.keystore
```

```
VMKEYSTORE=/root/vmware-certs/vmware.keystore
```

SNMP Traps

Site Recovery Manager provides Simple Network Management Protocol (SNMP) traps that collect information sent by the API. All traps are compliant with the SNMPv1 type. Information provided by the traps can be used to initiate actions by client applications.Callers of the Site Recovery Manager API interface should listen for the SNMP traps. You might need to configure the vCenter Server to forward the SNMP traps to the registered SNMP Server. The MIB file is located in the following directory:`<installdir>\www\VMWARE-SRM-TRAPS-5_0.MIB`

There are two ways to generate SNMP traps from Site Recovery Manager. The first is the method presented here and in other Site Recovery Manager documentation. The second method to generate traps is by configuring SNMP actions on the events and alarms that Site Recovery Manager adds to vCenter Server. Alarms with SNMP traps configured are all raised using the generic alarm definition in `VMWARE-VC-EVENT.mib`. Consequently alarm-based traps do not have explicit definitions. To manage them, you would need to synthesize the trap, capture its contents, parse the trap, then determine how to filter it.

Look at MIB Names for SNMP Traps for more details.

MIB Names for SNMP Traps

The listed SNMP traps originate from the Site Recovery Manager, not from vCenter Server. Descriptions of SNMP traps are given according to their names in the MIB file. The names in this list can be prefaced by either `vmwareSrm` (Site Recovery Manager) or `oidDr` (object ID data recovery)

Table 7-1. SNMP Traps in the MIB

SNMP Trap	What Trap Indicates
`RecoveryPlanExecuteTestBegin`	Signaled on the recovery site when a recovery test is initiated.
`RecoveryPlanExecuteTestEnd`	Signaled on the recovery site when a recovery test has completed. If an error occurred it is available as [data.Error]
`RecoveryPlanExecuteBegin`	Signaled on the recovery site when a recovery is initiated.
`RecoveryPlanExecuteEnd`	Signaled on the recovery site when a recovery has completed. If an error occurred it is available as [data.Error]
`RecoveryVmBegin`	Signaled when the recovery virtual machine was successfully created. If an error occurs before the virtual machine's ID is known, the event is not fired.
`RecoveryVmEnd`	Signaled after the last post-power on script has completed, or after a recovery-stopping error has occurred for the virtual machine.
`RecoveryPlanPromptDisplay`	The recovery plan is displaying prompt [data.PromptKey] and is waiting for user input. PromptKey is a unique identifier.
`RecoveryPlanPromptResponse`	The recovery plan received an answer to prompt [data.PromptKey] and is no longer paused waiting for user input.
`RecoveryPlanServerCommandBegin`	Signaled on the recovery site when Site Recovery Manager starts to run a Callout command on the Site Recovery Manager server.
`RecoveryPlanServerCommandEnd`	Signaled on the recovery site when Site Recovery Manager has finished running a Callout command on the Site Recovery Manager server.
`RecoveryPlanVmCommandBegin`	Signaled on the recovery site when Site Recovery Manager has started to run a Callout command on a recovered virtual machine.
`RecoveryPlanVmCommandEnd`	Signaled on the recovery site when Site Recovery Manager has finished running a Callout command on a recovered virtual machine.
`RecoveryPlanExecuteReprotectBegin`	Signaled on the recovery site when a reprotect is initiated.

Table 7-1. SNMP Traps in the MIB (continued)

SNMP Trap	What Trap Indicates
RecoveryPlanExecuteReprotectEnd	Signaled on the recovery site when a reprotect has completed. If an error occurred it is available as [data.Error]
RecoveryPlanExecuteCleanupBegin	Signaled on the recovery site when a test cleanup is initiated.
RecoveryPlanExecuteCleanupEnd	Signaled on the recovery site when a test cleanup has completed. If an error occurred it is available as [data.Error]

Configuring SNMP Receivers in vCenter Server

For a simple procedure to configure SNMP receivers, see the section “Configure SNMP Settings in the vSphere Web Client” in the vSphere vCenter Server and Host Management manual, available in the VMware vSphere 5.5 Documentation Center. For details about configuring the SNMP receiver URL, receiver port, and community, see the section “Configure SNMP Settings for vCenter Server by Using the vSphere Web Client” in the vSphere Monitoring and Performance manual, also in the VMware vSphere 5.5 Documentation Center.

SNMP Traps and Object IDs

The MIB objects are listed below with IDs, then the SMNP traps themselves with IDs.

Table 7-2. MIB objects with IDs

MIB_OBJECT	ID
oidDrVmName	1.3.6.1.4.1.6876.51.1.1
oidDrRecoveryName	1.3.6.1.4.1.6876.51.1.2
oidDrPromptString	1.3.6.1.4.1.6876.51.1.3
oidDrRecoveryType	1.3.6.1.4.1.6876.51.1.4
oidDrRecoveryState	1.3.6.1.4.1.6876.51.1.5
oidDrSiteString	1.3.6.1.4.1.6876.51.1.6
oidDrVmUuid	1.3.6.1.4.1.6876.51.1.7
oidDrResult	1.3.6.1.4.1.6876.51.1.8
oidDrCommandName	1.3.6.1.4.1.6876.51.1.9 - new MIB object

Table 7-3. SMNP traps with IDs

MIB_TRAP	ID
RecoveryPlanExecuteTestBegin	1.3.6.1.4.1.6876.51.0.1
RecoveryPlanExecuteTestEnd	1.3.6.1.4.1.6876.51.0.2
RecoveryPlanExecuteBegin	1.3.6.1.4.1.6876.51.0.3
RecoveryPlanExecuteEnd	1.3.6.1.4.1.6876.51.0.4
RecoveryVmBegin	1.3.6.1.4.1.6876.51.0.5
RecoveryVmEnd	1.3.6.1.4.1.6876.51.0.6

Table 7-3. SMNP traps with IDs (continued)

MIB_TRAP	ID
RecoveryPlanPromptDisplay	1.3.6.1.4.1.6876.51.0.7
RecoveryPlanPromptResponse	1.3.6.1.4.1.6876.51.0.8
RecoveryPlanServerCommandBegin	1.3.6.1.4.1.6876.51.0.9 - new MIB object
RecoveryPlanServerCommandEnd	1.3.6.1.4.1.6876.51.0.10 - new MIB object
RecoveryPlanVmCommandBegin	1.3.6.1.4.1.6876.51.0.11 - new MIB object
RecoveryPlanVmCommandEnd	1.3.6.1.4.1.6876.51.0.12 - new MIB object
RecoveryPlanExecuteReprotectBegin	1.3.6.1.4.1.6876.51.0.13 - new MIB object
RecoveryPlanExecuteReprotectEnd	1.3.6.1.4.1.6876.51.0.14 - new MIB object
RecoveryPlanExecuteCleanupBegin	1.3.6.1.4.1.6876.51.0.15 - new MIB object
RecoveryPlanExecuteCleanupEnd	1.3.6.1.4.1.6876.51.0.16 - new MIB object