Hewlett Packard
Enterprise

HPE SiteScope

Software Version: 11.32

SiteScope Public API Reference Guide

Document Release Date: March 2016
Software Release Date: March 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Intel®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

iPod is a trademark of Apple Computer, Inc.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hp.com/group/softwaresupport/search-result?keyword=.

This site requires an HPE Passport account. If you do not have one, click the **Create an account** button on the HPE Passport Sign in page.

Support

Visit the HPE Software Support web site at: **https://softwaresupport.hp.com**

This web site provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software Support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE support contacts
- Review information about available services

- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and sign in. Many also require a support contract. To register for an HPE Passport ID, go to **https://softwaresupport.hp.com** and click **Register**.

To find more information about access levels, go to:
https://softwaresupport.hp.com/web/softwaresupport/access-levels

HPE Software Integrations, Solutions and Best Practices

Visit the Solution and Integration portal at https://hpenterprise.sharepoint.com/teams/aztec/Portal/index.html to explore how the products in the HPE Software catalog work together, exchange information, and solve business needs.

Visit the Cross Portfolio Best Practices Library at **https://hpln.hpe.com/group/best-practices-hpsw** to access a wide variety of best practice documents and materials.

Contents

How This API Document Is Organized ... 8

Part 1: SOAP-based APIs ... 9

Chapter 1: Configuration APIs ... 10

addAcknowledgment ... 16

addLicense ... 17

addTagValue ... 18

addTagValuesToMonitor ... 19

createNewTag ... 20

createTemplateContainer ... 21

deleteGroupEx ... 22

deleteGroupByExternalId ... 23

deleteMonitorEx ... 24

deleteRemote ... 25

deleteTag ... 26

deleteTemplate ... 27

deleteTemplateContainer ... 28

deploySingleTemplateEx ... 29

deploySingleTemplateWithConnectToServer ... 30

deploySingleTemplateWithConnectToServerAndTestRemotes ... 31

deploySingleTemplateWithResult ... 32

disableAlertEx ... 33

disableAssociatedAlerts ... 34

disableGroupFullPathEx ... 35

disableGroupWithDescription ... 36

disableMonitorEx ... 37

disableMonitorWithDescription ... 38

editTagDescription ... 39

editTagValueDescription ... 40

editTagValueName ... 41

enableAlertEx ... 42

enableAssociatedAlerts ... 43

enableGroupEx ... 44

enableGroupWithDescription ... 45

enableMonitorEx ... 46

enableMonitorWithDescription ... 47

exportTemplate ... 48

getAcknowledgments 49
getAlertReport 50
getAlertSnapshots 51
getAllTemplates 52
getConfigurationSnapshotEx 53
getConfigurationViaTemplateEx 54
getConfigurationViaSourceTemplateEx 55
getFullConfigurationSnapshot 56
getGroupsConfigurationSnapshot 57
getHostsMap 58
getMonitorSnapshots 59
getQuickReport 60
getReadOnlyMode 61
getSiteScopeMonitoringStatus 62
getSiteScopeMonitoringStatusWithIdentifier 63
getSchedulePreferencesSnapshot 64
importSSHKey 65
importTemplate 66
importTemplateWithOverride 67
publishTemplateChanges 68
removeTagValue 69
removeTagValuesFromMonitor 70
runExistingMonitorEx 71
runExistingMonitorExWithIdentifier 72
runExistingMonitorsInGroup 73
runMonitorFromTemplate 74
runToolOnMonitorEx 75
search 76
setReadOnlyMode 77
updateMonitorViaTemplateEx 78
updateTemplate 79
updateViaSourceTemplateEx 80
updateViaTemplateEx 81
updateViaTemplateWithRootGroupEx 82
Chapter 2: Data Acquisition APIs 83
getData 84
getDataWithTopology 86
getMonitorTypesWithMetricNames 88
Example: SOAP Query for Data Acquisition API 89
Chapter 3: Use-Case Scenario - Configuring SiteScope APIs Calls 91
Part 2: REST APIs 96

Chapter 4: Configuration APIs 97
addAcknowledgment 98
addLicense 99
addTagValue 100
addTagValuesToMonitor 101
createNewTag 102
createTemplateContainer 103
deleteGroupByExternalId 104
deleteGroupEx 105
deleteMonitorEx 106
deleteRemote 107
deleteTag 108
deleteTemplate 109
deleteTemplateContainer 110
deploySingleTemplateEx 111
deploySingleTemplateWithConnectToServer 112
deploySingleTemplateWithConnectToServerAndTestRemotes 113
deploySingleTemplateWithResult 114
disableAlertEx 115
disableAssociatedAlerts 116
disableGroupFullPathEx 117
disableGroupWithDescription 118
disableMonitorEx 119
disableMonitorWithDescription 120
editTagDescription 121
editTagValueDescription 122
editTagValueName 123
enableAlertEx 124
enableAssociatedAlerts 125
enableGroupEx 126
enableGroupWithDescription 127
enableMonitorEx 128
enableMonitorWithDescription 129
exportTemplate 130
getAcknowledgments 131
getAlertReport 132
getAlertSnapshots 133
getAllTemplates 134
getConfigurationSnapshotEx 135
getConfigurationViaSourceTemplateEx 136
getConfigurationViaTemplateEx 137

getFullConfigurationSnapshot ... 138
getGroupsConfigurationSnapshot ... 139
getGroupSnapshots ... 140
getHostsMap ... 141
getMonitorSnapshots ... 142
getQuickReport ... 143
getReadOnlyMode ... 144
getSiteScopeMonitoringStatus ... 145
getSiteScopeMonitoringStatusWithIdentifier ... 146
getSchedulePreferencesSnapshot ... 147
importSSHKey ... 148
importTemplate ... 149
importTemplateWithOverride ... 150
publishTemplateChanges ... 151
removeTagValue ... 152
removeTagValuesFromMonitor ... 153
runExistingMonitorEx ... 154
runExistingMonitorExWithIdentifier ... 155
runExistingMonitorsInGroup ... 156
runToolOnMonitorEx ... 157
search ... 158
setReadOnlyMode ... 159
updateMonitorViaTemplateEx ... 160
updateTemplate ... 161
updateViaSourceTemplateEx ... 162
updateViaTemplateEx ... 163
updateViaTemplateWithRootGroupEx ... 164
Chapter 5: Data Acquisition APIs ... 165
getData ... 166
getDataWithTopology ... 167
getMonitorTypesWithMetricNames ... 168
Send Documentation Feedback ... 169

How This API Document Is Organized

Note: This guide was updated for the latest SiteScope build patch.

This Application Programming Interface (API) document contains detailed information about SiteScope Public APIs. SiteScope public APIs enable you to perform various tasks automatically without using the SiteScope user interface. SiteScope provides SOAP-based APIs and REST APIs to perform these tasks.

This document is divided into main parts:

- SOAP-based APIs
- REST APIs

Each part lists methods in alphabetical order. Each method is described in a topic with description, method parameters, and returned data. A use-case scenario describes how the SiteScope administrator can automate the process of configuring and deploying a monitor.

Part 1: SOAP-based APIs

The SOAP-based APIs for SiteScope can be invoked by any known Web Services framework such as Axis or WSIF, or by any SOAP client application. This provides a powerful set of tools for managing and automating large environments and implementing complex business logics.

You can find additional information on SiteScope SOAP-based APIs, including exceptions, snapshots, and error codes in the HPE SiteScope API Reference javadoc which is located in <SiteScope installation directory>\examples\integrations\api\doc\javadoc.zip file. To open the guide, extract the contents of the zip file and double-click the index.html file.

Chapter 1: Configuration APIs

The following configuration actions are supported using the SiteScope Configuration API:

Method	Description
addAcknowledgment	Adds an acknowledgment comment to an entity (monitor or group), and enables or disables the entity's associated alerts. For details, see "addAcknowledgment" on page 16.
addLicense	Adds a license to SiteScope. For details, see "addLicense" on page 17.
addTagValue	Adds a tag value by the name `tagValueName` and description `tagValueDescription` to an existing tag with the name `tagName`. For details, see "addTagValue" on page 18.
addTagValuesToMonitor	Adds tag values to a monitor. For details, see "addTagValuesToMonitor" on page 19
createNewTag	Creates a new tag with the name tagName. "createNewTag" on page 20
createTemplateContainer	Creates a template container (an exception is thrown if a template container with the requested name already exists). For details, see "createTemplateContainer" on page 21.
deleteGroup	*Deprecated.* Use *"deleteGroupEx" on page 22* instead.
deleteGroupEx	Deletes a group from SiteScope. For details, see "deleteGroupEx" on page 22.
deleteGroupByExternalId	Deletes a group by its external ID. For details, see "deleteGroupByExternalId" on page 23.
deleteMonitor	*Deprecated.* Use *"deleteMonitorEx" on page 24* instead.
deleteMonitorEx	Deletes a monitor. For details, see "deleteMonitorEx" on page 24.
deleteRemote	Deletes a SiteScope remote server. For details, see *"deleteRemote" on page 25*.
deleteTag	Deletes a tag by the name `tagName`. For details, see *"deleteTag" on page 26*.
deleteTemplate	Deletes a template. For details, see "deleteTemplate" on page 27.

Method	Description
deleteTemplateContainer	Deletes a template container. For details, see "deleteTemplateContainer" on page 28.
deploySingleTemplate	*Deprecated.* Use *"deploySingleTemplateEx" on page 29* instead.
deploySingleTemplateEx	Deploys a single template. For details, see "deploySingleTemplateEx" on page 29.
deploySingleTemplateWith ConnectToServer	Deploys a single template, with option to verify monitor measurements against the remote server during deployment. For details, see "deploySingleTemplateWithConnectToServer" on page 30.
deploySingleTemplateWith ConnectToServerAndTest Remotes	Deploys a single template, with option to test deployed remote server and verify monitor measurements against the remote server during deployment. For details, see "deploySingleTemplateWithConnectToServerAndTestRemotes" on page 31.
deploySingleTemplate WithResult	Deploys a single template and provides details of the template deployment results. For details, see "deploySingleTemplateWithResult" on page 32.
disableAlert	*Deprecated.* Use *"disableAlertEx" on page 33* instead.
disableAlertEx	Disables the specified alert. For details, see "disableAlertEx" on page 33.
disableAssociatedAlerts	Disables the alerts associated with the given entity (Group or Monitor). For details, see "disableAssociatedAlerts" on page 34.
disableGroupFullPath	*Deprecated.* Use *"disableGroupFullPathEx" on page 35* instead.
disableGroupFullPathEx	Disables all monitors under the specified group and its subgroups. For details, see "disableGroupFullPathEx" on page 35.
disableGroupWithDescription	Disables a group with given time period and description. For details, see "disableGroupWithDescription" on page 36.
disableMonitor	*Deprecated.* Use *"disableMonitorEx" on page 37* instead.
disableMonitorEx	Disables a monitor. For details, see "disableMonitorEx" on page 37.

Method	Description
disableMonitorWithDescription	Disables a monitor with given time period and description. For details, see "disableMonitorWithDescription" on page 38.
editTagDescription	Changes the description value to `tagDescription` for a tag with the name `tagName`. For details, see "editTagDescription" on page 39.
editTagValueDescription	Changes the tag description value to `tagValueDescription` for a tag with the name `tagName` for the value with the name `tagValue`. For details, see "editTagValueDescription" on page 40.
editTagValueName	Changes the tag value name from `oldTagValueName` to `newTagValueName` for a tag with the name `tagName`. For details, see "editTagValueName" on page 41.
enableAlert	*Deprecated.* Use *"enableAlertEx" on page 42* instead.
enableAlert	Enables the specified alert. For details, see "enableAlertEx" on page 42.
enableAssociatedAlerts	Enables the alerts associated with the given entity (Group or Monitor). For details, see "enableAssociatedAlerts" on page 43.
enableGroup	*Deprecated.* Use *"enableGroupEx" on page 44* instead.
enableGroupEx	Enables a group whether it was disabled indefinitely or for a specified time period. For details, see "enableGroupEx" on page 44.
enableGroupWithDescription	Enables a group regardless of whether the group was disabled indefinitely, or for a specified time period. For details, see "enableGroupWithDescription" on page 45.
enableMonitor	*Deprecated.* Use *"enableMonitorEx" on page 46* instead.
enableMonitorEx	Enables a monitor whether it was disabled indefinitely or for a specified time period. For details, see "enableMonitorEx" on page 46.
enableMonitorWithDescription	Enables a monitor with given description regardless of whether the monitor was disabled indefinitely, or for a specified time period. For details, see "enableMonitorWithDescription" on page 47.
exportTemplate	Exports the template. For details, see "exportTemplate" on page 48.

Method	Description
getAcknowledgment	Returns the acknowledgment data log of the given Entity. For details, see "getAcknowledgments" on page 49.
getAlertReport	Returns the Alert Report URL for the monitor or group. For details, see "getAlertReport" on page 50.
getAlertSnapshots	Returns the corresponding snapshots for the alerts. For details, see "getAlertSnapshots" on page 51.
getAllTemplates	Gets all the template. For details, see "getAllTemplates" on page 52.
getConfigurationSnapshotEx	*Deprecated.* Use *"getConfigurationSnapshotEx" on page 53* instead.
getConfigurationSnapshotEx	Returns a map of the currently deployed entities in SiteScope together with basic properties for each entity. For details, see "getConfigurationSnapshotEx" on page 53.
getConfigurationViaTemplate	*Deprecated.* Use *"getConfigurationViaTemplateEx" on page 54* instead.
getConfigurationViaTemplateEx	Returns a map of template variables to current values. For details, see "getConfigurationViaTemplateEx" on page 54.
getConfigurationViaSource TemplateEx	Returns a map of template variables to current values. For details, see "getConfigurationViaSourceTemplateEx" on page 55.
getFullConfigurationSnapshot	Returns a map of the currently deployed entities in SiteScope together with all the entity's properties. For details, see "getFullConfigurationSnapshot" on page 56.
getGroupsConfiguration Snapshot	Returns the corresponding snapshots for the group. For details, see "getGroupsConfigurationSnapshot" on page 57.
getGroupSnapshots	*Deprecated.* Use *"getGroupsConfigurationSnapshot" on page 57* instead.
getHostsMap	Returns a map of the hosts monitored by SiteScope. For details, see "getHostsMap" on page 58.
getMonitorSnapshots	Returns the corresponding snapshots for the given monitors. For details, see "getMonitorSnapshots" on page 59.
getQuickReport	Returns the Quick Report URL for the monitor or group. For details, see "getQuickReport" on page 60.
getReadOnlyMode	Returns true if SiteScope APIs are in read-only mode; otherwise it returns false. For details, see "getReadOnlyMode" on page 61.

Method	Description
getSiteScopeMonitoringStatus	Returns the SiteScope monitoring status string. For details, see "getSiteScopeMonitoringStatus" on page 62.
getSiteScopeMonitoringStatus WithIdentifier	Returns the SiteScope monitoring status string. For details, see "getSiteScopeMonitoringStatusWithIdentifier" on page 63.
importSSHKey	Imports the given SSH key file to SiteScope. For details, see "importSSHKey" on page 65.
importTemplate	Imports a template to SiteScope. For details, see "importTemplate" on page 66.
importTemplateWithOverride	Imports an external template. For details, see "importTemplateWithOverride" on page 67.
publishTemplateChanges	Publishes template changes to all deployed groups associated with the selected template. For details, see "publishTemplateChanges" on page 68.
removeTagValue	Removes tag value by the name `tagValueName` for a tag with the name `tagName`. For details, see "removeTagValue" on page 69.
removeTagValuesFromMonitor	Removes tag values from a monitor. For details, see "removeTagValuesFromMonitor" on page 70.
runExistingMonitor	*Deprecated.* Use *"runExistingMonitorEx" on page 71* instead.
runExistingMonitorEx	Runs the monitor. For details, see "runExistingMonitorEx" on page 71.
runExistingMonitorExWith Identifier	Runs the monitor. For details, see "runExistingMonitorExWithIdentifier" on page 72.
runExistingMonitorsInGroup	Runs existing monitors in group. For details, see "runExistingMonitorsInGroup" on page 73.
runMonitorFromTemplate	Creates a temporary monitor instance from the template (it replaces variables), and then runs the monitor. For details, see "runMonitorFromTemplate" on page 74.
runToolOnMonitorEx	Runs the monitor configuration tool for specific monitors to help configure the monitor settings. For details, see "runToolOnMonitorEx" on page 75
search	Gets the relevant elements (monitor or groups) according to the given search criteria. For details, see "search" on page 76.
setReadOnlyMode	Sets SiteScope API to read-only mode. The only configuration changes allowed in this mode are *getConfiguration* and *runExistingMonitors*. For details, see "setReadOnlyMode" on page 77.

Method	Description
updateMonitorViaTemplateEx	Updates a single monitor deployed by a template with new variables. For details, see "updateMonitorViaTemplateEx" on page 78.
updateViaTemplate	*Deprecated.* Use *"updateViaTemplateEx" on page 81* instead.
updateViaSourceTemplateEx	Updates a group of entities that were created with a template deployment operation. For details, see "updateViaSourceTemplateEx" on page 80.
updateViaTemplateEx	Updates a group of entities that were created with a template deployment operation. For details, see "updateViaTemplateEx" on page 81.
updateViaTemplateWith RootGroupEx	Updates the template deployment to use the new variables. The full path to the deployed group should point to a root group. For details, see "updateViaTemplateWithRootGroupEx" on page 82.

addAcknowledgment

The **addAcknowledgment** method adds an acknowledgment comment to an entity (monitor or group), and enables or disables the entity's associated alerts.

Usage	`public void addAcknowledgment(String[] fullPathToEntity,`
	`String acknowledgeComment,`
	`String associatedAlertsDisableStartTime,`
	`String associatedAlertsDisableEndTime,`
	`String associatedAlertsDisableDescription,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `acknowledgeComment` - The acknowledgment comment to add. `associatedAlertsDisableStartTime` - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `associatedAlertsDisableDescription` - Associated alerts disable's description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

addLicense

The **addLicense** method adds a license to SiteScope.

Usage	`public void addLicense(byte[] licenseFile,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`licenseFile` - Binary representation of the license file. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted
Throws	`ExternalServiceAPIException`

addTagValue

The **addTagValue** method adds a tag value by the name `tagValueName` and description `tagValueDescription` to an existing tag with the name `tagName`. An exception is thrown if the tag does not exist. If the tag does exist and also a tag value by the name `tagValueName` exists, a uniqueness valuation exception is thrown.

Usage	`public void addTagValue(String tagName, String tagValueName, String tagValueDescription, String username, String password) throws ExternalServiceAPIException`
Parameters	`tagName` - The tag's name. `tagValueName` - The tag's value name. `tagDescription` - The tag's description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

addTagValuesToMonitor

The **addTagValuesToMonitor** method adds tag values to a monitor.

Usage	`public void addTagValuesToMonitor(String[] fullPathToMonitor,`
	`String tagName,`
	`String[] tagValueNames,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - Full path from SiteScope root to monitor as sequence of groups and monitor in array format. `tagName` - Name of tag that holds the values. `tagValueNames` - Names of values to be checked in monitor. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

createNewTag

The **createNewTag** method creates a new tag with the name tagName. An exception is throw if a tag by this name already exists.

Usage	`public void createNewTag(String tagName, String tagDesc, String[] valueName, String[] valueDesc, String username, String password) throws ExternalServiceAPIException`
Parameters	`tagName` - tag's name. `tagDesc` - tag's description. `valueName` - tag's value name. valueDesc - tag's value description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

createTemplateContainer

The **createTemplateContainer** method creates a template container (it throws an exception if a template container with the requested name already exists).

Usage	`public void createTemplateContainer(String fatherEntityFullPath, String templateContainerName, String username, String password) throws ExternalServiceAPIException`
Parameters	`fatherEntityFullPath` - A String specifying the full path to the template container or SiteScope root to create the template container under. This parameter should be an empty string when a template container is created under the SiteScope root. The path should start with the name of the first template container name under SiteScope's root and be separated by forward slashes. For example: "tc1/tc2" `templateContainerName` - Name of requested template container. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

deleteGroupEx

The **deleteGroupEx** method deletes a group from SiteScope.

Usage	`public void deleteGroupEx(String[] fullPathToGroup,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group to delete. The path starts with the name of the first child under SiteScope's root and ends with the name of the group to delete. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

deleteGroupByExternalId

The **deleteGroupByExternalId** method deletes a group by its external ID.

Usage	`public void deleteGroupEx(String[] fullPathToGroup, String username, String password) throws ExternalServiceAPIException`
Parameters	`groupExternalId` - External ID of the group. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException` - If there are errors during group deletion.

deleteMonitorEx

The **deleteMonitorEx** method deletes a monitor.

Usage	`public void deleteMonitorEx(String[] fullPathToMonitor,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to delete. The path starts with the name of the first child under SiteScope's root and ends with the name of the monitor to delete. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	Whether SiteScope APIs are in read-only mode or not.
Throws	`ExternalServiceAPIException`

deleteRemote

The **deleteRemote** method deletes a SiteScope remote server.

Usage	`public void deleteRemote(String platform,`
	` String remoteName,`
	`String username,`
	` String password)`
	`throws ExternalServiceAPIException`

Parameters	`platform` - "Windows" for Windows remote servers or "UNIX" for Unix remote servers. `remoteName` - Remote display name. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

deleteTag

The **deleteTag** method deletes a tag by the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) an entity depends on it.

Usage	`public void deleteTag(String tagName,`
	` String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`tagName` - The tag's name. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

deleteTemplate

The **deleteTemplate** method deletes a template.

Usage	`public void deleteTemplate(String templateFullPath,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`templateFullPath` - A String specifying the full path to the template to delete. The path should start with the name of the first template container name under the SiteScope root and be separated by forward slashes (/). For example: "tc1/tc2/tcToDelete" `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

deleteTemplateContainer

The **deleteTemplateContainer** method deletes a template container.

Usage	`public void deleteTemplateContainer(String templateContainerFullPath, String username, String password) throws ExternalServiceAPIException`
Parameters	`templateContainerFullPath` - A String specifying the full path to the template container to delete. The path should start with the name of the first template container name under the SiteScope root and be separated by forward slashes (/). For example: "tc1/tc2/tcToDelete" `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

deploySingleTemplateEx

The **deploySingleTemplateEx** method deploys a single template. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

Usage	`public void` **`deploySingleTemplateEx`**`(String[] fullPathToTemplateName, HashMap actualVariablesValuesHashMap, String[] pathToTargetGroup, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplateName` - A String array specifying the full path to the template name to deploy. The path starts with the name of the first child under SiteScope's root and ends with the name of the template. `actualVariablesValuesHashMap` - A String->String Hash Map of all variables in the template and their values. `pathToTargetGroup` - A String array specifying the full path to the group to deploy the template name under. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. To deploy the template in the SiteScope root, pass a non-null String array of length 0. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

deploySingleTemplateWithConnectToServer

The **deploySingleTemplateWithConnectToServer** method deploys a single template, with option to verify monitor measurements against the remote server during deployment. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

Usage	`public void` **`deploySingleTemplateWithConnectToServer`**`(String[] fullPathToTemplateName, HashMap actualVariablesValuesHashMap, String[] pathToTargetGroup, boolean connectToServer, String userName, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplateName` - A String array specifying the full path to the template name to deploy. The path starts with the name of the first child under SiteScope's root and ends with the name of the template. `actualVariablesValuesHashMap` - A String->String Hash Map of all variables in the template and their values. `pathToTargetGroup` - A String array specifying the full path to the group to deploy the template name under. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. To deploy the template in the SiteScope root, pass a non-null String array of length 0. `connectToServer` - If true, the monitor measurements are verified against the remote server during deployment. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

deploySingleTemplateWithConnectToServerAndTestRemotes

The **deploySingleTemplateWithConnectToServerAndTestRemotes** method deploys a single template, with option to test deployed remote server and verify monitor measurements against the remote server during deployment. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

Usage	`public void` **`deploySingleTemplateWithConnectToServerAndTestRemotes`**`(String[] fullPathToTemplateName, HashMap actualVariablesValuesHashMap, String[] pathToTargetGroup, boolean connectToServer, boolean testRemotes, String userName, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplateName` - A String array specifying the full path to the template name to deploy. The path starts with the name of the first child under SiteScope's root and ends with the name of the template. `actualVariablesValuesHashMap` - A String->String Hash Map of all variables in the template and their values. `pathToTargetGroup` - A String array specifying the full path to the group to deploy the template name under. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. To deploy the template in the SiteScope root, pass a non-null String array of length 0. `connectToServer` - If true, the monitor measurements are verified against the remote server during deployment. `testRemotes` - If true, runs test on deployed remote server. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

deploySingleTemplateWithResult

The **deploySingleTemplateWithResult** method deploys a single template and provides details of the template deployment results.

Usage	`public HashMap` **`deploySingleTemplateWithResult`**`(String[] fullPathToTemplateName, HashMap actualVariablesValuesHashMap, String[] pathToTargetGroup, boolean connectToServer, boolean testRemotes, String username, String password, String identifier) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplateName` - A String array specifying the full path to the template name to deploy. The path starts with the name of the first child under SiteScope's root and ends with the name of the template. `actualVariablesValuesHashMap` - A String->String Hash Map of all variables in the template and their values. `pathToTargetGroup` - A String array specifying the full path to the group to deploy the template name under. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. To deploy the template in the SiteScope root, pass a non-null String array of length 0. `connectToServer` - If true, the monitor measurements are verified against the remote server during deployment. `testRemotes` - If true, runs test on deployed remote server. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	HashMap contains actual details of deployment.
Throws	`ExternalServiceAPIException` - If errors occurred during deployment.

disableAlertEx

The **disableAlertEx** method disables the specified alert.

Usage	`public void` **`disableAlertEx`**`(String[] fullPathToAlert, String username, String password)` `throws ExternalServiceAPIException`
Parameters	`fullPathToAlert` - A String array specifying the full path to the alert to disable. The path should starts with the name of the first child under SiteScope's root and ends with with the name of the alert to be disabled. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

disableAssociatedAlerts

The **disableAssociatedAlerts** method disables the alerts associated with the given entity (Group or Monitor).

Usage	`public void disableAssociatedAlerts(String[] fullPathToEntity,`
	`String disableStartTime,`
	`String disableEndTime,`
	`String disableDescription,`
	`String username,`
	`String password,`
	` String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `disableStartTime` - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds) `disableEndTime` - The time difference in milliseconds from the [current time] and the required [end time]. For example: If the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds) `disableDescription` - Disable's description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

disableGroupFullPathEx

The **disableGroupFullPathEx** method disables all monitors under the specified group. If the group contains subgroups, their monitors are also disabled, and so on recursively to the bottom of the tree. Disabling a group that is already disabled has no effect.

Usage	`public void` **`disableGroupFullPathEx`**`(String[] fullPathToGroup,`
	`long timePeriod,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group to disable. The path starts with the name of the first child under SiteScope's root and ends with the name of the group to disable. `timePeriod` - The length of time the group is disabled, in seconds. If 0, disabled until explicitly enabled. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

disableGroupWithDescription

The **disableGroupWithDescription** method disables a group with given time period and description.

Usage	`public void` **`disableGroupWithDescription`**`(String[] fullPathToGroup,`
	`String fromTime,`
	`String toTime,`
	` String description,`
	`String username,`
	`String password,`
	`String identifier)`
	` throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group to disable. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the group to disable. `fromTime` - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `toTime` - The time difference in milliseconds from the [current time] and the required [end time]. For example: If the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds) To permanently disable the group, the time period between fromTime and toTime should be zero. For example: fromTime = 0 and toTime = 0 `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

disableMonitorEx

The **disableMonitorEx** method disables a monitor.

Usage	`public void` **`disableMonitorEx`**`(String[] fullPathToMonitor,`
	`long timePeriod,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to disable. The path starts with the name of the first child under SiteScope's root and ends with the name of the monitor to disable. `timePeriod` - The length of time the monitor is disabled for, in seconds. If 0, disables until explicitly enabled. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

disableMonitorWithDescription

The **disableMonitorWithDescription** method disables a monitor with given time period and description.

Usage	`public void disableMonitorWithDescription(String[] fullPathToMonitor, String fromTime, String toTime, String disableDescription, String username, String password, String identifier) throws ExternalServiceAPIException`
Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to disable. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the monitor to disable. `fromTime` - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `toTime` - The time difference in milliseconds from the [current time] and the required [end time]. For example: If the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). To permanently disable the monitor, the time period between fromTime and toTime should be zero. For example: fromTime = 0 and toTime = 0 `disableDescription` - Monitor's disable description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. identifier - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

editTagDescription

The **editTagDescription** method changes the description value to tagDescription for a tag with the name tagName. An exception is throw if a tag by this name does not exist.

Usage	`public void editTagDescription(String tagName, String tagDescription, String username, String password) throws ExternalServiceAPIException`
Parameters	`tagName` - tag's name. `tagDescription` - tag's description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

editTagValueDescription

The **editTagValueDescription** method changes the tag description value to `tagValueDescription` for a tag with the name `tagName` for the value with the name `tagValue`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists, but a tag value by the name `tagValueName` does not exist.

Usage	`public void editTagValueDescription(String tagName,`
	`String tagValueName,`
	`String tagValueDescription,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`tagName` - The tag's name. `tagValueName` - The tag's value name. `tagValueDescription` - The tag's value description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

editTagValueName

The **editTagValueName** method changes the tag value name from `oldTagValueName` to `newTagValueName` for a tag with the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists but a tag value by the name `oldTagValueName` does not exist.

Usage	`public void editTagValueName(String tagName,`
	`String oldTagValueName,`
	` String newTagValueName,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`tagName` - The tag's name. `oldTagValueName` - The tag's old value name. `newTagValueName` - The tag's new value name `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

enableAlertEx

The **enableAlertEx** method enables the specified alert.

Usage	`public void` **`enableAlertEx`**`(String[] fullPathToAlert, String username, String password)` `throws ExternalServiceAPIException`
Parameters	`fullPathToAlert` - A String array specifying the full path to the alert to enable. The path starts with the name of the first child under SiteScope's root and ends with the name of the alert. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

enableAssociatedAlerts

The **enableAssociatedAlerts** method enables the alerts associated with the given entity (Group or Monitor).

Usage	`public void` **`enableAssociatedAlerts`**`(String[] fullPathToEntity,`
	`String description,`
	`String username,`
	`String password,`
	`String identifier)`
	` throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	A list of acknowledgments.
Throws	`ExternalServiceAPIException`

enableGroupEx

The **enableGroupEx** method enables a group whether it was disabled indefinitely or for a specified time period. Enabling a group that is already enabled has no effect.

Usage	`public void` **`enableGroupEx`**`(String[] fullPathToGroup,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group to enable. The path starts with the name of the first child under SiteScope's root and ends with the name of the group to enable. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

enableGroupWithDescription

The **enableGroupWithDescription** method enables a group regardless of whether the group was disabled indefinitely, or for a specified time period.

Usage	`public void` **`enableGroupWithDescription`**`(String[] fullPathToGroup,`
	` String description,`
	` String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group to enable. The path starts with the name of the first child under SiteScope's root and ends with the name of the group to enable. `description` - Group's enable description `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. identifier - Identifier to be written to audit log
Throws	`ExternalServiceAPIException`

enableMonitorEx

The **enableMonitorEx** method enables a monitor whether it was disabled indefinitely or for a specified time period. Enabling a monitor that is already enabled has no effect.

Usage	`public void` **`enableMonitorEx`**`(String[] fullPathToMonitor,`
	`String username,`
	`String password)`
	` throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to enable. The path starts with the name of the first child under SiteScope's root and ends with the name of the monitor to enable. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

enableMonitorWithDescription

The **enableMonitorWithDescription** method enables a monitor with given description regardless of whether the monitor was disabled indefinitely, or for a specified time period.

Usage	`public void` **`enableMonitorWithDescription`**`(String[] fullPathToMonitor,`
	`String description,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to enable. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the monitor to enable. `description` - Monitor's enable description `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log
Throws	`ExternalServiceAPIException`

exportTemplate

The **exportTemplate** method exports the template.

Usage	`public byte[] exportTemplate(String templatePatch,`
	`String username,`
	`String password,`
	`String identifier)`
	` throws ExternalServiceAPIException`

Parameters	`templatePatch` - Path to template. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	Byte array contains template.
Throws	`ExternalServiceAPIException` - If some error occurred during the API call.

getAcknowledgments

The **getAcknowledgments** method returns the acknowledgment data log of the given Entity.

Usage	`public HashMap[] getAcknowledgments(String[] fullPathToEntity,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	A list of acknowledgments.
Throws	`ExternalServiceAPIException`

getAlertReport

The **getAlertReport** method returns the Alert Report URL for the monitor or group.

Usage	`public String` **`getAlertReport`**`(String[] fullPathToEntity,`
	`HashMap reportProperties,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `reportProperties` - Report properties. must contain the following keys: • start_time - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 14:50:00, the value that should be sent is [14:00:00] - [15:00:00] = -60*60*1000 (-3600000 milliseconds). • end_time - The time difference in milliseconds from the [current time] and the required [end time]. For example: If the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [14:30:00] - [15:00:00] = -30*60*1000 (-1800000 milliseconds). `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	Report URL without base part.
Throws	`ExternalServiceAPIException`

getAlertSnapshots

The **getAlertSnapshots** method returns the corresponding snapshots for the alerts.

Usage	`public HashMap` **`getAlertSnapshots`**`(String[] fullPathsToAlerts,`
	`HashMap propertiesFilter,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathsToAlerts` - An array of the alerts to which to return snapshots. The path to each alert will be delimited using "_sis_path_delimiter_". For example: group_sis_path_delimiter_ monitor_sis_path_delimiter_alert `propertiesFilter` - Properties to filter. Eash key stored in map will be filtered and not included in returned snapshot. Allowed filter values: `name, full_path, is_disabled`. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	A map of the snapshots for the given alert paths
Throws	`ExternalServiceAPIException`

getAllTemplates

The **getAllTemplates** method gets all the templates.

Usage	`public HashMap` **`getAllTemplates`**`(String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	Hashmap containing snapshot of all templates.
Throws	`ExternalServiceAPIException` - If some error occurred during the API call.

getConfigurationSnapshotEx

The **getConfigurationSnapshotEx** method returns a map of the currently deployed entities in SiteScope together with basic properties for each entity. You can use the **SnapshotConfigurationVisitor** method to convert the map representation back to a tree-like representation of the result.

Usage	`public HashMap` **`getConfigurationSnapshotEx`**`(String username, String password)` `throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of the currently deployed entities in SiteScope.
Throws	`ExternalServiceAPIException`

getConfigurationViaTemplateEx

The **getConfigurationViaTemplateEx** method returns a map of template variables to current values. Given a Template and a destination group under which the template has been deployed, returns the values that replace the template variables as the template is deployed in that group.

Usage	`public HashMap` **`getConfigurationViaTemplateEx`**`(String[] fullPathToTemplate, String[] fullPathToTargetGroup, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplate` - A String array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. `fullPathToTargetGroup` - A String array specifying the full path to the group. The path starts with the first group under SiteScope root and ends with the group the template was deployed under. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of template variables to current values.
Throws	`ExternalServiceAPIException`

getConfigurationViaSourceTemplateEx

The **getConfigurationViaSourceTemplateEx** method returns a map of template variables to current values. Given a Template and a destination group under which the template has been deployed, returns the values that replace the template variables as the template is deployed in that group.

Usage	`public HashMap` **`getConfigurationViaSourceTemplateEx`**`(String[] fullPathToTargetGroup, String username, String password) throws RemoteException, ExternalServiceAPIException`
Parameters	`fullPathToTargetGroup` - A String array specifying the full path to the group. The path starts with the first group under SiteScope root and ends with the group the template was deployed under. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of template variables to current values.
Throws	`ExternalServiceAPIException`

getFullConfigurationSnapshot

The **getFullConfigurationSnapshot** method returns a map of the currently deployed entities in SiteScope together with all the entity's properties.You can use the **SnapshotConfigurationVisitor** method to convert the map representation back to a tree-like representation of the result.

Usage	`public HashMap` **`getFullConfigurationSnapshot`**`(String username, String password) throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of the currently deployed entities in SiteScope.
Throws	`ExternalServiceAPIException`

getGroupsConfigurationSnapshot

The **getGroupsConfigurationSnapshot** method returns the corresponding snapshots for the group.

Usage	`public HashMap` **`getGroupsConfigurationSnapshot`**`(String[] fullPathsToGroups, boolean isFullConfig, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathsToGroups` - An array of the groups to which to return snapshots. The path to each alert will be delimited using "_sis_path_delimiter_". For example: group1_sis_path_delimiter_group2_sis_path_delimiter_group3. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of the snapshots for the given group paths.
Throws	`ExternalServiceAPIException`

getHostsMap

The **getHostsMap** method returns a map of the hosts monitored by SiteScope.

Usage	`public HashMap getHostsMap(String username, String password) throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A map of hosts monitored by SiteScope. Host name is used as a key and data is Map object of host data containing the number of monitors that are monitoring this host.
Throws	`ExternalServiceAPIException` - on failure

getMonitorSnapshots

The **getMonitorSnapshots** method returns the corresponding snapshots for the given monitors.

Usage	`public HashMap` **`getMonitorSnapshots`**`(String[] fullPathsToMonitors,`
	`HashMap propertiesFilter,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathsToMonitors` - An array of the monitor paths to which to return snapshots. The path to each monitor is be delimited using "_sis_path_delimiter_". For example: group_sis_path_delimiter_monitor `propertiesFilter` - Properties to filter. Each key stored in map will be filtered and not included in returned snapshot. Allowed filter values: `name, full_path, type, target_ip, target_name, target_display_name, updated_date, description, is_disabled_permanently, disable_description, disable_start_time, disable_end_Time, is_associated_alerts_disabled, associated_alerts_disable_description, associated_alerts_disable_start_time, associated_alerts_disable_end_time, acknowledgment_comment, status, availability, availability_description, summary, configuration_snapshot, runtime_snapshot`.  `username` - SiteScope user name, either plain text or encrypted.  `password` - Either plain text or encrypted.  `identifier` - Identifier to be written to audit log.
Returns	A map of the snapshots for the given monitor paths.
Throws	`ExternalServiceAPIException`

getQuickReport

The **getQuickReport** method returns the Quick Report URL for the monitor or group.

Usage	`public String` **`getQuickReport`**`(String[] fullPathToEntity,`
	`HashMap reportProperties,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToEntity` - A String array specifying the full path to the entity. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the entity. `reportProperties` - Report properties. must contain the following keys: • start_time - The time difference in milliseconds from the [current time] and the required [start time]. For example: If the current time is 15:00:00 and the required start time is 14:50:00, the value that should be sent is [14:00:00] - [15:00:00] = -60*60*1000 (-3600000 milliseconds). • end_time - The time difference in milliseconds from the [current time] and the required [end time]. For example: If the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [14:30:00] - [15:00:00] = -30*60*1000 (-1800000 milliseconds). `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	Report URL without base part.
Throws	`ExternalServiceAPIException`

getReadOnlyMode

The **getReadOnlyMode** method returns true if SiteScope APIs are in read-only mode; otherwise it returns false.

Usage	`public boolean` **`getReadOnlyMode`**`(String username, String password) throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	Whether SiteScope APIs are in read-only mode or not.
Throws	`ExternalServiceAPIException` - on failure

getSiteScopeMonitoringStatus

The **getSiteScopeMonitoringStatus** method returns the SiteScope monitoring status string. The returned value is one of:

- MONITORING_PASSIVE__STARTUP. The initial state from the beginning of SiteScope startup until the monitoring engine starts.
- MONITORING_ACTIVE. From the time the monitoring engine is active and monitors are running until SiteScope starts to shutdown.
- MONITORING_PASSIVE__SHUTDOWN. From the beginning of SiteScope shutdown until the process exits.

Usage	`public String` **`getSiteScopeMonitoringStatus`**`(String username, String password) throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	SiteScope monitoring status string.
Throws	`ExternalServiceAPIException`

getSiteScopeMonitoringStatusWithIdentifier

The **getSiteScopeMonitoringStatusWithIdentifier** method returns the SiteScope monitoring status string. The returned value is one of:

- MONITORING_PASSIVE__STARTUP. The initial state from the beginning of SiteScope startup until the monitoring engine starts.
- MONITORING_ACTIVE. From the time the monitoring engine is active and monitors are running until SiteScope starts to shutdown.
- MONITORING_PASSIVE__SHUTDOWN. From the beginning of SiteScope shutdown until the process exits.

Usage	`public String` **`getSiteScopeMonitoringStatusWithIdentifier`**`(String username, String password, String identifier) throws ExternalServiceAPIException`
Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	SiteScope monitoring status string.
Throws	`ExternalServiceAPIException`

getSchedulePreferencesSnapshot

The **getSchedulePreferencesSnapshot** method retrieves all schedule preferences that are available in SiteScope.

Usage	`public HashMap` **`getSchedulePreferencesSnapshot`**`(String username, String password`
	`String identifier)`
	`throws RemoteException,ExternalServiceAPIException`

Parameters	`username` - SiteScope user name, either plain text or encrypted. `password` - SiteScope password, either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	An array of schedule preference details such as schedule type, ID, name, description, range, related entities, and related tags.
Throws	`ExternalServiceAPIException`

importSSHKey

The **importSSHKey** method imports the given SSH key file to SiteScope.

Usage	`public String importSSHKey(byte[] sshKeyFileBinary,`
	`String sshKeyFileName,`
	`boolean override,`
	` String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`sshKeyFileBinary` - SSH key file binary `sshKeyFileName` - SSH key file name `override` - If override allowed or not `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	The relative path to imported file.
Throws	`ExternalServiceAPIException` - If errors occurred while importing file.

importTemplate

The **importTemplate** method imports a template to SiteScope.

Usage	`public void importTemplate(String templateDestinationFullPath, byte[] templateData, String username, String password) throws ExternalServiceAPIException`
Parameters	`templateDestinationFullPath` - A String specifying the full path to the template container to import the template under. The path should start with the name of the first template container name under the SiteScope root and be separated by forward slashes (/). For example: "tc1/tc2" `templateData` - Binary template representation. Exported template via SiteScope user interface. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

importTemplateWithOverride

The **importTemplateWithOverride** method imports an external template.

Usage	`public void importTemplateWithOverride(String templateDestinationFullPath, byte[] templateData, String username, String password, boolean override) throws ExternalServiceAPIException`
Parameters	`templateDestinationFullPath` - Path to import template `templateData` - Binary array with template data `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `override` - If override allowed or not
Throws	`ExternalServiceAPIException`

publishTemplateChanges

The **publishTemplateChanges** method publishes template changes to all deployed groups associated with the selected template.

Usage	`public String` **`publishTemplateChanges`**`(String templatePath, HashMap selectedGroupsWithVariables, boolean connectToServer, boolean deleteOnUpdate, String username, String password, String identifier) throws ExternalServiceAPIException`
Parameters	`templatePath` - Path to template. `selectedGroupsWithVariables` - This can be empty. In this case, the function searches all groups associated with the selected template, and publishes changes to these groups. It can also include `HashMap` which contains the key's path to groups affected by publishing changes as values HashMap's of variables. If `HashMap` variables are empty, the default template variables values are used. You can specify the variables to update by sending `HashMap` variables in the format *Variable Name - > Variable Value*. `connectToServer` - If set to true, the connection to the remote server is established while publishing changes. `deleteOnUpdate` - If set to true, the delete on update functionality is allowed (SiteScope deleted all objects from the deployed groups that are not in the source template). `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	Publish result reports.
Throws	`ExternalServiceAPIException` - If errors occurred while publishing template changes.

removeTagValue

The **removeTagValue** method removes tag value by the name `tagValueName` for a tag with the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists, but a tag value by the name `tagValueName` does not exist, or (iii) an entity depends on it.

Usage	`public void removeTagValue(String tagName,`
	`String tagValueName,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`tagName` - The tag's name. `tagValueName` - The tag's value name. `tagValueDescription` - The tag's value description. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

removeTagValuesFromMonitor

The **removeTagValuesFromMonitor** method removes tag values from a monitor.

Usage	`public void removeTagValuesToMonitor(String[] fullPathToMonitor,`
	`String tagName,`
	`String[] tagValueNames,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - Full path from SiteScope root to monitor as sequence of groups and monitor in array format. `tagName` - Name of tag that holds the values. `tagValueNames` - Names of values to be checked in monitor. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

runExistingMonitorEx

The **runExistingMonitorEx** method runs the monitor. The monitor must be deployed before invoking this method.

Usage	`public HashMap` **`runExistingMonitorEx`**`(String[] fullPathToMonitor,`
	`long timeout,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to run. The path starts with the name of the first child under SiteScope's root and ends with the name of the monitor. `timeout` - Timeout in seconds. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A hashmap representation of the status of the run and the status message as it would appear in the user interface. You can use **SnapshotConfigurationVisitor** to convert the hashmap to a class representation of the result.
Throws	`ExternalServiceAPIException` - on failure

runExistingMonitorExWithIdentifier

The **runExistingMonitorExWithIdentifier** method runs the monitor. The monitor must be deployed before invoking this method.

Usage	`public HashMap` **`runExistingMonitorExWithIdentifier`**`(String[] fullPathToMonitor, long timeout, String username, String password, String identifier) throws ExternalServiceAPIException`
Parameters	`fullPathToMonitor` - A String array specifying the full path to the monitor to run. The path starts with the name of the first child under the SiteScope's root and ends with the name of the monitor. `timeout` - In seconds. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	A hashmap representation of the status of the run and the status message as it would appear in the user interface. You can use the **SnapshotConfigurationVisitor** method to convert the hashmap to a class representation of the result.
Throws	`ExternalServiceAPIException`

runExistingMonitorsInGroup

The **runExistingMonitorsInGroup** method runs existing monitors in group.

Usage	`public void runExistingMonitorsInGroup(String[] fullPathToGroup,`
	`boolean recursively,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToGroup` - A String array specifying the full path to the group. The path starts with the name of the first child under the SiteScope's root, and ends with the name of the group. `recursively` - Should it run all sub monitors. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

runMonitorFromTemplate

The **runMonitorFromTemplate** method creates a temporary monitor instance from the template (it replaces variables), and runs the monitor.

Usage	`public HashMap` **`runMonitorFromTemplate`**`(String templateName, HashMap actualVariablesValuesHashMap, long timeOut, String userName, String password) throws ExternalServiceAPIException`
Parameters	`templateName` - A String array specifying the full path to the template name to deploy. The path starts with the name of the first child under SiteScope's root and ends with the name of the template. `actualVariablesValuesHashMap` - A String->String Hash Map of all variables in the template and their values. `timeOut` - Timeout in seconds. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Returns	A hashmap representation of the status of the run and the status message as it would appear in the user interface. You can use the **SnapshotConfigurationVisitor** method to convert the hashmap to a class representation of the result.
Throws	`ExternalServiceAPIException` - on failure

runToolOnMonitorEx

The **runToolOnMonitorEx** method runs the monitor configuration tool for specific monitors to help configure the monitor settings.

Usage	`public String` **`runToolOnMonitorEx`**`(String[] fullPathToMonitor, boolean returnResultAsHtml, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToMonitor` - Full path to the requested monitor. `returnResultAsHtml` - Returns the result as HTML or as plain String for the relevant monitors only. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - in case of authentication/authorization failure

search

The **search** method gets the relevant elements (monitors, groups or tags) according to the given search criteria. You can specify regular expressions in addition to plain text search strings. The method also allows to search for monitors and groups based on their tag names and values. The returned results include the entities of the selected entity_type (Monitors, Groups or Tags) that match ANY of the search criteria that are passed in the parameters name, path, target_name, target_display_name, status OR tags.

Usage	`public HashMap search(HashMap searchCriteria,`
	` int maxNumOfResults,`
	`String username,`
	`String password, String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`searchCriteria` - Use the following keys: • target_name - Monitor's target name • target_display_name - Monitor's target display name • name - Monitor's name • path - Monitor's full path (use "_sis_path_delimiter_" as path delimiter) • entity_type - monitor/group/tag/empty string (for both monitors and groups) • status - good/warning/error/empty string (for both monitors and groups) • searchregex - "true" or "false". If set to "true", all values passed in other search parameters are treated as regular expressions and the method searches for regular expression matches. The default value is false and in such a case, all parameter values will be treated as plain text. • tags - Map of tag name value pairs to search monitor and groups. It is in the format `tagName:tagValue` with multiple tag name value pairs separated by commas. `maxNumOfResults` - Maximum number of returned search results. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted. `identifier` - Identifier to be written to audit log.
Returns	A map of path->entity's type. The key is the entity's path with `_sis_path_delimiter_` as the delimiter. The value is the entity's type (Monitor, Group, or Tag) **Note:** None of the keys are mandatory.
Throws	`ExternalServiceAPIException`

setReadOnlyMode

The **setReadOnlyMode** method sets SiteScope API to read-only mode. The only configuration changes allowed in this mode are **getConfiguration** and **runExistingMonitors**.

Usage	`public void setReadOnlyMode(boolean isReadOnlyMode, String username, String password) throws ExternalServiceAPIException`
Parameters	`isReadOnlyMode` - true/false. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

updateMonitorViaTemplateEx

The **updateMonitorViaTemplateEx** method updates a single monitor deployed by a template with new variables.

Usage	`public void` **`updateMonitorViaTemplateEx`**`(String[] fullPathToTemplate, String[] fullPathToDeployedMonitor, HashMap actualValuesToUpdate, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplate` - A String array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. `fullPathToDeployedMonitor` - A String array specifying the full path to the monitor. The path starts with the first group under SiteScope root and ends with the deployed monitor. `actualValuesToUpdate` - A map of variables to the new values. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

updateTemplate

The **updateTemplate** method enables you to update a template.

Usage	`public void` **`updateTemplate`**`(String fullPathToTemplate,`
	`HashMap properties,`
	`String username,`
	`String password,`
	`String identifier)`
	`throws ExternalServiceAPIException`

Parameters	`fullPathToTemplate` – A string specifying the full path to the template. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template with path elements separated by forward slashes (/). `properties` – contains the properties to be updated. "templateName" – string, the name of the template to be updated. **Note**: Currently only the `templateName` property is supported. `username` – SiteScope user name, either plain text or encrypted. `password` – Either plain text or encrypted. `identifier` – Identifier to be written to audit log.
Throws	`ExternalServiceAPIException`

updateViaSourceTemplateEx

The **updateViaSourceTemplateEx** method updates a group of entities that were created with a template deployment operation.

Usage	`public void` **`updateViaSourceTemplateEx`**`(String[] fullPathToDeployedGroup, HashMap actualValuesToUpdate, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToDeployedGroup` - A String array specifying the full path to the group. The path starts with the first group under SiteScope root and ends with the group the template was deployed under. `actualValuesToUpdate` - A map of variables to the new values. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

updateViaTemplateEx

The **updateViaTemplateEx** method updates a group of entities that were created with a template deployment operation.

Usage	`public void updateViaTemplateEx(String[] fullPathToTemplate, String[] fullPathToDeployedGroup, HashMap actualValuesToUpdate, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplate` - A String array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. `fullPathToDeployedGroup` - A String array specifying the full path to the group. The path starts with the first group under SiteScope root and ends with the group the template was deployed under. `actualValuesToUpdate` - A map of variables to the new values. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException`

updateViaTemplateWithRootGroupEx

The **updateViaTemplateWithRootGroupEx** method updates the template deployment to use the new variables. The full path to the deployed group should point to a root group.

Usage	`public void` **`updateViaTemplateWithRootGroupEx`**`(String[] fullPathToTemplate, String[] fullPathToDeployedRootGroup, HashMap actualValuesToUpdate, String username, String password) throws ExternalServiceAPIException`
Parameters	`fullPathToTemplate` - A String array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. `fullPathToDeployedRootGroup` - A String array specifying the full path to the group. The path starts with the first group under SiteScope root and ends with the deployed root group. `actualValuesToUpdate` - A map of variables to the new values. `username` - SiteScope user name, either plain text or encrypted. `password` - Either plain text or encrypted.
Throws	`ExternalServiceAPIException` - on failure

Chapter 2: Data Acquisition APIs

The following data acquisition actions are supported using the SiteScope Data Acquisition API:

Method	Description
getData	Retrieves historical metrics data for monitor runs matching the specified query parameters. For details, see "getData" on the next page
getDataWithTopology	Retrieves historical metrics data for monitor runs matching the specified query parameters and VMware reconciliation topology collected by VMware monitors currently running on SiteScope. • Supports given time interval, credentials, and filter (monitor type(s), name, etc…) • Returns XML similar to the XML sent with generic data integration that contains the (historical) metrics data For details, see "getDataWithTopology" on page 86.
getMonitorTypesWithMetricNames	Scans all the monitors in this SiteScope instance for which the user has view permissions, and returns a list of their types together with the metric names per monitor type. The list of metric names is merged from all the monitors of each type (repeated occurrences are removed). Where `enabledMonitorsOnly` is true, it scans enabled monitors only. Where `enabledMonitorsOnly` is false, it scans all monitors (enabled/disabled) in the SiteScope instance. For details, see "getMonitorTypesWithMetricNames" on page 88.

getData

The **getData** method gets historical data for monitor runs matching the specified query parameters. The data is taken from the SiteScope daily log.

Usage	`public byte[] getData (String [] query,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`query` - Array of parameters by which to filter the SiteScope daily log data. Parameters should be specified in the following order, and separated by commas: [START_TIME, END_TIME, MONITOR_TYPE, TARGET_SERVER, BSM_ID, MONITOR_NAME, DATA_GRANULARITY] where: START_TIME - Start of time frame in which to get historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). Mandatory. END_TIME - End of time frame in which to get historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). Mandatory. MONITOR_TYPE - Monitor type(s) for which to get data. A monitor type is its 'Topaz name' as detailed in SiteScope documentation. TARGET_SERVER - Server name(s) monitored by this SiteScope for which to get data. BSM_ID - Monitor BSM ID(s) for which to get data. MONITOR_NAME - Monitor name(s) for which to get data. Monitor name appears in the general settings of the monitor properties. DATA_GRANULARITY - Granularity of the data in seconds. Data samples for every [DATA_GRANULARITY] seconds will be listed in the response. To pass several monitor types, monitor names, monitor BSM ID's or target servers, separate them with a #,# token. For example: [START_TIME,END_TIME,MONITOR_TYPE1#,#MONITOR_TYPE2,TARGET_SERVER1#,#TARGET_SERVER2,DATA_GRANULARITY] `username` - User name for authentication `password` - Password for authentication
Returns	byte array of a compressed (gzip) XML with the requested data For an example of requested and retrieved data for all URL monitors that ran between a specified start and end time, see "Example: SOAP Query for Data Acquisition API" on page 89.

Throws	RemoteException `ExternalServiceAPIException` - • Start time or end time are null or empty. • Start time is not chronologically earlier than end time. • The amount of memory required by the server to carry out this request violates the memory limits specified in the configuration preferences of the server.

getDataWithTopology

The **getDataWithTopology** method gets historical data for monitor runs matching the specified query parameters, with reconciliation topology for VMware monitors. The data is taken from the SiteScope daily log. The reconciliation topology is collected by VMware monitors currently running on SiteScope. Reconciliation topology for monitors that existed in the specified time frame but no longer exist at the time the request is made, is not available in the response.

Reconciliation topology matching the above constraints includes:

- Details of VMware objects referenced in the counters of the VMware monitors whose run data is within the specified time frame.
- Links between the above VMware objects.
- References between the VMware objects and the counters in the run data.

Usage	`public byte[] getDataWithTopology(String[] query,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`query` - Array of parameters by which to filter the SiteScope daily log data. Parameters should be specified in the following order, and separated by commas: [START_TIME, END_TIME, MONITOR_TYPE, TARGET_SERVER, BSM_ID, MONITOR_NAME, DATA_GRANULARITY] where: • START_TIME - Start of time frame in which to get historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). Mandatory. • END_TIME - End of time frame in which to get historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). Mandatory. • MONITOR_TYPE - Monitor type(s) for which to get data. A monitor type is its 'Topaz name' as detailed in SiteScope documentation. • TARGET_SERVER - Server name(s) monitored by this SiteScope for which to get data. • BSM_ID - Monitor BSM ID(s) for which to get data. • MONITOR_NAME - Monitor name(s) for which to get data. Monitor name appears in the general settings of the monitor properties. • DATA_GRANULARITY - Granularity of the data in seconds. Data samples for every [DATA_GRANULARITY] seconds will be listed in the response. • To pass several monitor types, monitor names, monitor BSM ID's or target servers, separate them with a #,# token. For example: [START_TIME,END_TIME,MONITOR_TYPE1#,#MONITOR_TYPE2,TARGET_SERVER1#,#TARGET_SERVER2,DATA_GRANULARITY] `username` - User name for authentication `password` - Password for authentication

Returns	Byte array of a compressed (gzip) XML with the requested data For an example of requested and retrieved data for all URL monitors that ran between a specified start and end time, see "Example: SOAP Query for Data Acquisition API" on page 89.
Throws	RemoteException ExternalServiceAPIException - • Start time or end time are null or empty. • Start time is not chronologically earlier than end time. • The amount of memory required by the server to carry out this request violates the memory limits specified in the configuration preferences of the server. • SiteScope is not set up to collect topology in the background: either it is not integrated with BSM, or topology collection is disabled (**Enable topology collection in standalone deployment** is not selected in **Infrastructure Preferences > General Settings**). When this exception is thrown, select the **Enable topology collection in standalone deployment** check box.

getMonitorTypesWithMetricNames

The **getMonitorTypesWithMetricNames** method scans all the monitors in this SiteScope instance for which the user has view permissions, and returns a list of their types together with the metric names per monitor type.

The list of metric names is merged from all the monitors of each type (repeated occurrences are removed). Where `enabledMonitorsOnly` is true, it scans enabled monitors only. Where `enabledMonitorsOnly` is false, it scans all monitors (enabled/disabled) in the SiteScope instance.

Usage	`public byte[] getMonitorTypesWithMetricNames(boolean enabledMonitorsOnly,`
	`String username,`
	`String password)`
	`throws ExternalServiceAPIException`

Parameters	`enabledMonitorsOnly` - If true only enabled monitors are scanned `username` - User name for authentication `password` - Password for authentication
Returns	Byte array of a compressed (gzip) XML with the requested data
Throws	ExternalServiceAPIException

Example: SOAP Query for Data Acquisition API

The **getData** and **getDataWithTopology** methods get historical data for monitor runs matching the specified query parameters. The SOAP query requires epoch time to be in milliseconds.

Below is an example of a request for xml data for all URL monitors that run between a specified start and end time:

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:data="http://data.api.sitescope.mercury.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
 <soapenv:Body>
   <data:getData soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <in0 xsi:type="data:ArrayOf_xsd_strng" soapenc:arrayType="xsd:string[]">
              <string>1431204690000</string>
              <string>1431464690000</string>
              <string>URL Monitor</string>
              <string></string>
              <string></string>
              <string></string>
              <string></string>
          </in0>
       <in1 xsi:type="xsd:string">admin</in1>
       <in2 xsi:type="xsd:string">admin</in2>
   </data:getData>
  </soapenv:Body>
</soapenv:Envelope>
```

Where epoch time in the query is:

- `1431204690` for START_TIME. This is equivalent to 05/09/2015 @ 8:51pm (UTC).
- `1431464690` for END_TIME. This is equivalent to 05/12/2015 @ 9:04pm (UTC).

The result of the data is base64 gzip xml.

To extract the base64 encoded data, perform the following:

1. Decode the file using the following URL:
   http://www.motobit.com/util/base64-decoder-encoder.asp
2. Save the file to a .bin file, and open it using WinRAR.

Below is an example of the request output, which shows all URL monitors that ran between the specified start and end time:

```
<monitor type="URL Monitor" target="www8.hp.com" targetIP="184.25.56.101"
time="1431521465000"
quality="1" name="HP Software URL">
       <counter value="Cloud" quality="good" name="content match"/>
       <counter value="50205.0" quality="good" name="size (bytes)"/>
```

```
       <counter value="192.0" quality="good" name="roundtrip time (milliseconds)"/>
       <counter value="200.0" quality="good" name="overall status"/>
       <counter value="9546.0" quality="good" name="age (seconds)"/>
       <counter value="200.0" quality="good" name="status"/>
       <counter value="0.0" quality="good" name="total errors (errors)"/>
       <counter value="0.0" quality="good" name="certificate expiration days
remaining"/>
       <counter value="0.0" quality="good" name="dns time (milliseconds)"/>
       <counter value="17.0" quality="good" name="connect time (milliseconds)"/>
       <counter value="41.0" quality="good" name="response time (milliseconds)"/>
       <counter value="134.0" quality="good" name="download time (milliseconds)"/>
     </monitor>
      <monitor type="URL Monitor" target="192.168.57.128" targetIP="192.168.57.128"
  time="1431503285000" quality="1" name="SiteScope Home">
       <counter value="" quality="good" name="content match"/>
       <counter value="7077.0" quality="good" name="size (bytes)"/>
       <counter value="12.0" quality="good" name="roundtrip time (milliseconds)"/>
       <counter value="200.0" quality="good" name="overall status"/>
       <counter value="1.43150387E9" quality="good" name="age (seconds)"/>
       <counter value="200.0" quality="good" name="status"/>
       <counter value="0.0" quality="good" name="total errors (errors)"/>
       <counter value="0.0" quality="good" name="certificate expiration days
remaining"/>
      </monitor>
```

Chapter 3: Use-Case Scenario - Configuring SiteScope APIs Calls

SiteScope configuration and data acquisition APIs enable you to run various scenarios automatically without using the SiteScope user interface.

This use-case scenario describes how the SiteScope administrator can automate the process of configuring and deploying a monitor. It includes the steps and APIs required to:

1. Import a monitoring configuration template to a specific template container (if the container does not exist, the code will create it).
2. Deploy the imported configuration template to a specified group path with parameters specified by user.

API Usage:

To perform this scenario, the SiteScope administrator needs to:

1. Create a template container using the **createTemplateContainer** API method (performed only once; ignore this step if the template container already exists).

   For method details, see "createTemplateContainer" on page 21.
2. Import a template using the **importTemplateWithOverride** API method.

   For method details, see "importTemplateWithOverride" on page 67.
3. Deploy a template for a server using the **deploySingleTemplateWithResult** API method.

   For method details, see "deploySingleTemplateWithResult" on page 32.

API Example:

For this scenario, we created an API example named **SiteScopeImportAndDeployTemplateWithResultCommandLineUtil.java** (available from **<SiteScope root>\examples\integrations\api\src**), and a batch file named **import_and_deploy_template.bat**, which calls the library that executes the API example.

Below is an example of how to fill the parameters for the batch file:

```
import_and_deploy_template.bat -host localhost -port 8080
-useSSL false -login admin -password admin
-templateContainerImportPath "TC" -templateFilePath SanityTemplates2
-deployGroupPath "MC" -deployTemplatePath "TC/sanity/basic OS monitors"
-testRemotes true -connectToServer true
-templateVariables SQLserver=sqlserver.hp.com,hostname=remotehost
-identifier "Template deploy":
```

The batch file is available from **<SiteScope root>\examples\integrations\api\bin\import_and_deploy_template.bat**.

To run that batch:

1. Make sure you have the latest Java version installed.
2. Open a command line and run:

   `C:\SiteScope\examples\integrations\api\bin\import_and_deploy_template.bat` with the below parameters and their values:

Parameter	Description
connectToServer	Selector to verify monitor measurements against the remote server during deployment.
deployGroupPath	Group of monitors on which the template is applied.
deployTemplatePath	Full path to the template (including template name) which would be deployed to the monitor group.
identifier	Identifier to be written to audit log.
overrideTemplate	Overrides template with identical path.
templateContainerImportPath	Parent container destination for new template, including template container name.
templateFilePath	Path in file system where the template file is located . This file is the import source.
templateVariables	A "Variable=Value" pairs all variables in the template with their values. Delimited by comma ",".
testRemotes	Selector to run a test on the deployed remote server.

API Configuration Used in this Example:

Below is the code used in the **SiteScopeImportAndDeployTemplateWithResultCommandLineUtil** API example, together with an explanation. You can find additional details in the java file.

Code	Explanation

`import java.rmi.RemoteException;`
`import com.mercury.sitescope.api.configuration.exception.`
`ExternalServiceAPIException;`
`import . . . (additional import classes i.e. HashMap, and etc.)`
	Imports:
	• `ExternalServiceAPIException` – If the API call fails for some reason, such as unable to find searched property, or unable to perform an action if server is in read only mode.
	• `RemoteException` – Is thrown from `apiCall` method that uses remote calls.

Code	Explanation
`public class SiteScopeImportAndDeployTemplateWithResultCommandLineUtil extends SiteScopeCommandLineUtil {`	A class that works with API should extend the `SiteScopeCommandLineUtil` which has the SiteScope server connection `apiConfiguration` and `apiDataAcquisition` objects for all the exposed public APIs.
`public static void main(String args[]) { try { SiteScopeImportAndDeployTemplateWithResultCommandLineUtil cmd = new SiteScopeImportAndDeployTemplateWithResultCommandLineUtil (); cmd.runCommand(args); }catch (ExternalServiceAPIException e){ System.err.println("\nFailed to run " + USAGE + " due to " + e.getErrorMessage()); System.exit(-1); } catch (Exception ex) { System.err.println("\nFailed to run " + USAGE + " due to " + ex); System.exit(-1); } }`	Your class should extend `SiteScopeCommandLineUtil` and must contain the `main` method, which is called by the batch file. The `runCommand` method is an inherited method that sequences the API call. It resolves parameters and their values from argument lines and calls the appropriate API method. It is important to filter the exceptions thrown from the API execution runtime—first by `ExternalServiceAPIException`, and then by other unexpected exceptions.
`protected void usage() { String generalUsage = createGeneralCmdUsage(); String usage = . . . String usageExp = . . . System.out.println(usage); System.out.println(generalUsage); System.out.println(usageExp); }`	The `usage` method generates usage rules and example text. Fill the strings to help users use your API via command line. If your implementation is intended for automation purposes, you can make it an empty method. For a detailed example, see the code in the API example. The method `createGeneralCmdUsage` is inherited from parent class.
`protected void apiCall() throws ExternalServiceAPIException, RemoteException {`	The `apiCall` method wraps the actual usage of `apiConfiguration` and `apiDataAcquisition`, and envelops them with pre- and post- execution messages. See the example in the following three sections of the code.

Code	Explanation

`final int NOT_FOUND = -1;`
 	
`String parentCont = "";`
`String contName = "";`
`Integer indexOfDelim =`
`pathToTemplateContainer.lastIndexOf("/");`
 	
`if (indexOfDelim==NOT_FOUND){`
`// if no delims, parent container is root and the path is the template name`
`contName = pathToTemplateContainer;`
`}else{`
`// if path supplied divide it to parent and suffix (template name)`
`parentCont = pathToTemplateContainer.substring (0,indexOfDelim);`
`contName = pathToTemplateContainerArr [pathToTemplateContainerArr.length-1];`
`}`
`System.out.println("\n\n\n Creating template container... " + pathToTemplateContainer);`
`try {`
`apiConfiguration.createTemplateContainer(parentCont, contName, login, password);`
`} catch (ExternalServiceAPIException e){`
`System.out.println("\n\nContainer creation skipped due " + e.getMessage() + "\n");`
`}`
	The goal in the current API example is to import a template to a specific template container. The code creates the container. If a template container already exists, the exception is filtered to prevent an API execution abort. For more details, see "createTemplateContainer" on page 21.

`System.out.println("\n\n\n Importing template... ");`
`System.out.println("Getting template file data file name is:"+`**`pathToTemplateFile`**`);`
 	
`templateBinary=SiteScopeFileUtil.getBytesFromFile (pathToTemplateFile);`
`System.out.println("\n Trying to import template");`
`System.out.println ("The deployment path is :" +pathToTemplateContainer);`
`apiConfiguration.importTemplateWithOverride (pathToTemplateContainer, templateBinary,login, password,override);`
	The template is a binary file that was exported from an existing SiteScope template. Enter the path to the template file. The code reads the path and translates it into actual bytes, which it sends to the target SiteScope server provided by the `host` parameter. For more details, see "importTemplateWithOverride" on page 67.

`//deploy`
`HashMap<String,String> result =`
`(HashMap<String,String>)`
`apiConfiguration.`**`deploySingleTemplateWithResult`**
`(pathToDeployTemplateArr, variables, pathToDeployGroupArr, connectToserver,testRemotes, login, password, identifier);`
`SiteScopeIOUtil.printMap(result,"");`
 	
`System.out.println("\nAction was successfully completed");`
`}`
	The `deploySingleTemplateWithResult` API call deploys the groups and monitors contained in the template. It takes the monitor properties and instantiates new active monitors, and then starts the monitors. It returns the deployment result in the form of new monitor properties. For more details, see "deploySingleTemplateWithResult" on page 32.

Code	Explanation

protected void **checkAdditionalParams**(Map<String, String> otherParams) {
variables = new HashMap<String, String>();
 for (String key : otherParams.keySet()) {
 if (key.equalsIgnoreCase(PATH_TO_TEMPLATE_CONTAINER)) {
 pathToTemplateContainer = otherParams.get(key);
 pathToTemplateContainerArr = pathToTemplateContainer.split(PATH_DELIM);
}
else if (key.equalsIgnoreCase(PATH_TO_DEPLOY_GROUP)) {
pathToDeployGroup = otherParams.get(key);
 pathToDeployGroupArr = pathToDeployGroup.split(PATH_DELIM);
}
else if (key.equalsIgnoreCase(PARAMETER_STRING)) {
. . .
}
else {
 System.out.println("\nUnknown argument " + key);
 usage();
 System.exit(-1);
 }
}
	The `checkAdditionalParams` method uses a map of parameters that was delivered in the command line used in the API call. The method iterates over the map, and for each key, it maps its value to the appropriate variable in your class. The following basic parameters should always be present: `host`, `port`, `username`, `password`, `useSSL`. They are deduced by the parent class from command line arguments. Other parameters need to be handled by the implementer as described in the code. If an unknown parameter is found, the method notifies the user with the usage rules print and stops execution.

Part 2: REST APIs

SiteScope supports a set of REST APIs that enable running of various scenarios automatically without using the SiteScope user interface. SiteScope REST APIs can be invoked using any known REST client framework or tool.

Common characteristics of SiteScope REST APIs are as follows:

- All SiteScope REST endpoints are accessed by sending an HTTP request to the SiteScope server. The server response contains either the data requested, or the status indicator, or both.
- All endpoints are located in a hierarchy starting from http(s)://<<SiteScope_Host>>:<<SiteScope_Port>>/api/
- All endpoints may return different HTTP status codes. The most common status codes and their description are as follows:

Status Code	Description
200	Success with the response containing requested data.
204	Success response where no data is expected to be returned.
400	Bad request; A message with details of the error condition is returned along with the response.
500	Error condition on the server along with a message indicating the details of the error condition.

- All SiteScope REST endpoints allow authentication through the following ways:
  - HTTP basic authentication by including the user name and password in the request header.
  - Client certificate authentication by sending a certificate along with the request when SiteScope APIs are configured for client certificate authentication as per hardening instructions available in the *SiteScope Deployment Guide*.
- For POST requests, the request media type must be "multipart/form-data" when the request contains binary or file upload parameters. For all other POST requests, the supported request media type is "application/x-www-form-urlencoded".

Chapter 4: Configuration APIs

SiteScope configuration APIs provide services for working with SiteScope templates, groups, monitors, and alerts. The following configuration actions are supported using the SiteScope REST APIs:

SiteScope Object	Action
Templates	• Template management (create/delete template, create/delete template container, import/export template, import templates and override them if they already exist in the given path, get snapshot of all templates). • Template deployment (monitor, group, alert), deploy a single template that gets back details of the deployment. • Publish template changes (groups, monitors, alerts, remote server); update templates deployed without a root (updates only a single monitor with new variables).
Groups	Enable/disable groups, delete groups, search groups by specific criteria.
Monitors	Enable/disable monitors, delete monitors, run monitors, search monitors by specific criteria.
Alerts	Enable/disable alerts.

f

addAcknowledgment

The **addAcknowledgment** method adds an acknowledgment comment to an entity (monitor or group), and enables or disables the entity's associated alerts.

REST End Point	/api/monitors/monitor/acknowledgement
Method	POST
Form Parameters	`fullPathToEntity`– A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the entity with the elements of the path separated by the string "_sis_path_delimiter_". `acknowledgeComment`– The acknowledgment comment to add. `associatedAlertsDisableStartTime`– The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `associatedAlertsDisableEndTime`– The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). `associatedAlertsDisableDescription`– Associated alerts disable description. `identifier`– Identifier to be associated with the operation and written to audit log.
Return Type	void
Status Codes	204 400 500

addLicense

The **addLicense** method adds a license to SiteScope.

REST End Point	/api/admin/licenses
Method	POST
Form Parameter	`{File} licenseFile`– Binary representation of the license file.
Returns	void
Status Codes	204 400 500

addTagValue

The **addTagValue** method adds a tag value by the name `tagValueName` and description `tagValueDescription` to an existing tag with the name `tagName`. An exception is thrown if the tag does not exist. If the tag does exist and also a tag value by the name `tagValueName` exists, a uniqueness valuation exception is thrown.

REST End Point	/api/admin/tags/tag/value
Method	PUT
Query Parameters	`tagName`– Name of the tag to for which a new value is to be added. `tagValueName`– Name of new tag value to be added. `tagValueDescription`– Description of the new tag value to be added.
Return Type	void
Status Codes	204 400 500

addTagValuesToMonitor

The **addTagValuesToMonitor** method adds tag values to a monitor.

REST End Point	/api/monitors/tags
Method	POST
Form Parameters	`fullPathToMonitor` – A string array specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with elements of the path separated by the string "_sis_path_delimiter_". `tagName`– The name of the tag that holds the values. `tagValueNames` – The names of values to be checked in monitor. `active`– Set to "true" to make the tag values active. If set to empty or "false" the tag values are made inactive.
Return Type	void
Status Codes	204 400 500

createNewTag

The **createNewTag** method creates a new tag with the name tagName. An exception is thrown if a tag by this name already exists.

REST End Point	/api/admin/tags/tag
Method	PUT
Query Parameters	`tagName`– Name of the tag to be created. `tagDescription`– Description of the new tag to be created. `valueNames`– Names of values separated by commas (,). `valueDescs`– Descriptions of the tag values separated by commas (,).
Return Type	void
Status Codes	204 400 500

createTemplateContainer

The **createTemplateContainer** method creates a template container. The method throws an exception if a template container with the requested name already exists.

REST End Point	/api/templates/templateContainer
Method	PUT
Query Parameters	`templateContainerFullPath` – A string specifying the full path to the template container to be created. If the container is to be created under the SiteScope root, this parameter must have only the name of the new container to be created, for example, "tc1". Else this parameter must start with the name of the first template container under the SiteScope root and contain the full path to the desired new template with the path elements separated by the string "_sis_path_delimiter_". , for example "tc1_sis_path_delimiter_tc2".
Return Type	void
Status Codes	204 400 500

deleteGroupByExternalId

The **deleteGroupByExternalId** method deletes a group by its external ID.

REST End Point	/api/monitors/group
Method	DELETE
Query Parameters	`externalId`– (Optional) External ID of the group. `identifier` – (Optional) Identifier to be written to the audit log. `fullPathToGroup`– A string array specifying the full path to the group to be deleted. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to delete with the elements separated by the string "_sis_path_delimiter_".
Return Type	void
Status Codes	204 400 500

deleteGroupEx

The **deleteGroupEx** method deletes a group from SiteScope.

REST End Point	/api/monitors/group
Method	DELETE
Query Parameters	`externalId`– (Optional) External ID of the group. `identifier`– (Optional) Identifier to be written to audit log. `fullPathToGroup`– A string array specifying the full path to the group to be deleted. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to delete with the elements separated by the string "_sis_path_delimiter_".
Return Type	void
Status Codes	204 400 500

deleteMonitorEx

The **deleteMonitorEx** method deletes a monitor.

REST End Point	/api/monitors/monitor
Method	DELETE
Query Parameter	`fullPathToMonitor` – A string array specifying the full path to the monitor to delete. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor to be deleted with the elements separated by the string "_sis_path_ delimiter_".
Returns	void
Throws	204 400 500

deleteRemote

The **deleteRemote** method deletes a SiteScope remote server.

REST End Point	/api/admin/remote
Method	DELETE
Query Parameters	`platform` – Specify "Windows" for Windows remote servers or "UNIX" for Unix remote servers. `remoteName` – Name of the remote server to be deleted.
Return Type	`ExternalServiceAPIException` - on failure
Status Codes	204 400 500

deleteTag

The **deleteTag** method deletes a tag by the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) an entity depends on it.

REST End Point	/api/admin/tags/tag
Method	DELETE
Query Parameter	`tagName` – Name of the tag to be deleted.
Return Type	void
Status Codes	204 400 500

deleteTemplate

The **deleteTemplate** method deletes a template.

REST End Point	/api/templates/template
Method	DELETE
Query Parameters	`templateFullPath` – A string specifying the full path to the template to be deleted. If the template is to be deleted from the SiteScope root directory the parameter must have only the name of the template to be deleted, for example, "templateToDelete". Else this parameter must start with the first template container under the SiteScope root and contains the full path to the desired template with the path elements separated by the string "_sis_path_ delimiter_". For example, "tc1_sis_path_delimiter_ tc2_sis_path_delimiter_ templateToDelete".
Return Type	void
Status Codes	204 400 500

deleteTemplateContainer

The **deleteTemplateContainer** method deletes a template container.

REST End point	/api/templates/templateContainer
Method	DELETE
Query Parameters	`templateContainerFullPath` – A string specifying the full path to the template container to be deleted. If the container to be deleted is under the SiteScope root, then this parameter must have only the name of the container to be deleted, for example, "tc1". Else this parameter must start with the first template container under the SiteScope root and contain the full path to the desired template with the path elements separated by the string "_sis_ path_delimiter_", for example: "tc1_sis_path_delimiter_ tc2".
Return Type	void
Status Codes	204 400 500

deploySingleTemplateEx

The **deploySingleTemplateEx** method deploys a single template. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

REST End Point	/api/templates/templateDeployment
Method	POST
Form Parameters	`pathToTemplate` – A string array specifying the full path to the template name to deploy. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template. The elements of the path are separated by the string "_sis_path_ delimiter_". `pathToTargetGroup` – A string array specifying the full path to the group where the template is to be deployed. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. The elements of the path are separated by the string "_sis_path_delimiter_". The elements of the path are separated by the string "_sis_ path_delimiter_". `connectToServer` – Set it to "true" to verify monitor measurements against the remote server during deployment. If set to "false" monitor measurements are not verified against the remote server. `testRemotes` – Set it "true" to run the test on deployed remote server. All other parameters required for the deployment as specified by the template being deployed.
Return Type	void
Status Codes	204 400 500

deploySingleTemplateWithConnectToServer

The **deploySingleTemplateWithConnectToServer** method deploys a single template, with option to verify monitor measurements against the remote server during deployment. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

REST End Point	/api/templates/templateDeployment
Method	POST
Form Parameters	`pathToTemplate` – A string array specifying the full path to the template name to deploy. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template. The elements of the path are separated by the string "_sis_path_ delimiter_". `pathToTargetGroup` – A string array specifying the full path to the group where the template is to be deployed. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. The elements of the path are separated by the string "_sis_path_delimiter_". `connectToServer` – Set it to "true" to verify monitor measurements against the remote server during deployment. If set to "false" monitor measurements are not verified against the remote server. `testRemotes` – Set it "true" to run the test on deployed remote server. All other parameters required for the deployment as specified by the template being deployed.
Return Type	void
Status Codes	204 400 500

deploySingleTemplateWithConnectToServerAndTestRemotes

The **deploySingleTemplateWithConnectToServerAndTestRemotes** method deploys a single template, with option to test deployed remote server and verify monitor measurements against the remote server during deployment. If there is a non-recoverable failure, either all the entities under the template are deployed or none of them are deployed.

REST End Point	/api/templates/templateDeployment
Method	POST
Form Parameters	`pathToTemplate` – A string array specifying the full path to the template name to deploy. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template. The elements of the path are separated by the string "_sis_path_ delimiter_". `pathToTargetGroup` – A string array specifying the full path to the group where the template is to be deployed. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. The elements of the path are separated by the string "_sis_path_delimiter_". `connectToServer` – Set it to "true" to verify monitor measurements against the remote server during deployment. If set to "false" monitor measurements are not verified against the remote server. `testRemotes` – Set it "true" to run the test on deployed remote server. All other parameters required for the deployment as specified by the template being deployed.
Return Type	void
Status Codes	204 400 500

deploySingleTemplateWithResult

The **deploySingleTemplateWithResult** method deploys a single template and provides details of the template deployment results.

REST End Point	/api/templates/templateDeploymentWithResult
Method	POST
Form Parameters	`pathToTemplate` – A string array specifying the full path to the template name to deploy. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template. The elements of the path are separated by the string "_sis_path_ delimiter_". `pathToTargetGroup` – A string array specifying the full path to the group where the template needs to be deployed. If the last element in the path does not exist, the function creates it and deploys the template under the new path element. The elements of the path are separated by the string "_sis_path_delimiter_". `connectToServer` – Set it to "true" to verify monitor measurements against the remote server during deployment. `testRemotes` – Set it "true" to run the test on deployed remote server. `identifier` – Identifier to be associated with deployment and written to audit log. All other parameters required for the deployment as specified by the template being deployed.
Returns	Map containing actual details of deployment.
Status Codes	200 400 500

disableAlertEx

The **disableAlertEx** method disables the specified alert.

REST End Point	/api/monitors/alert/status
Method	POST
Form Parameters	`fullPathToAlert`– A string specifying the full path to the alert to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the alert to enable/disable with elements of the path separated by the string "_ sis_path_delimiter_". `enable`– Set it to "true" for enabling the alert and set it to "false" or empty for disabling the alert.
Return Type	void
Status Codes	204 400 500

disableAssociatedAlerts

The **disableAssociatedAlerts** method disables the alerts associated with the given entity (Group or Monitor).

REST End Point	/api/monitors/monitor/alerts
Method	POST
Form Parameters	`fullPathToEntity` – A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the entity with elements of the path separated by the string "_sis_path_delimiter_". `enable` – Set to "true" to enable alerts and set to "false" or empty to disable alerts. `disableStartTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Required only for disabling the alerts. `disableEndTime`– The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Required only for disabling the alerts. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

disableGroupFullPathEx

The **disableGroupFullPathEx** method disables all monitors under the specified group. If the group contains subgroups, their monitors are also disabled, and so on recursively to the bottom of the tree. Disabling a group that is already disabled has no effect.

REST End Point	/api/monitors/group/status
Method	POST
Form Parameters	`fullPathToGroup`– A string specifying the full path to the group to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to be enabled/disabled with elements of the path separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Group is enabled if set to "true" and group is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the group should be disabled. If set to 0, group is disabled until explicitly enabled. Applicable only for disabling a group. `fromTime`– The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a group. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a group. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

disableGroupWithDescription

The **disableGroupWithDescription** method disables a group with given time period and description.

REST End Point	/api/monitors/group/status
Method	POST
Form Parameters	`fullPathToGroup`– A string specifying the full path to the group to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to be enabled/disabled with elements of the path separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Group is enabled if set to "true" and group is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the group should be disabled. If set to 0, group is disabled until explicitly enabled. Applicable only for disabling a group. `fromTime`– The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a group. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a group. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

disableMonitorEx

The **disableMonitorEx** method disables a monitor.

REST End Point	/api/monitors/monitor/status
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor to enable/disable. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor to enable/disable. The elements of the path are separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Monitor is enabled if set to "true" and monitor is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the monitor should be disabled. If set to 0, monitor is disabled until it is explicitly enabled. If `fromTime` and `toTime` are specified for disabling the monitor, the timePeriod value is ignored. Applicable only for disabling a monitor. `fromTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a monitor. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a monitor. `description` – Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

disableMonitorWithDescription

The **disableMonitorWithDescription** method disables a monitor with given time period and description.

REST End Point	/api/monitors/monitor/status
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor to enable/disable. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor to enable/disable. The elements of the path are separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Monitor is enabled if set to "true" and monitor is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the monitor should be disabled. If set to 0, monitor is disabled until it is explicitly enabled. If `fromTime` and `toTime` are specified for disabling the monitor, the timePeriod value is ignored. Applicable only for disabling a monitor. `fromTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a monitor. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a monitor. `description` – Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

editTagDescription

The **editTagDescription** method changes the description value to tagDescription for a tag with the name tagName. An exception is thrown if a tag by this name does not exist.

REST End Point	/api/admin/tags/tag
Method	POST
Query Parameters	`tagName` – Name of the tag to be modified. `tagDescription` – New value for description of the tag.
Return Type	void
Status Codes	204 400 500

editTagValueDescription

The **editTagValueDescription** method changes the tag description value to `tagValueDescription` for a tag with the name `tagName` for the value with the name `tagValue`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists, but a tag value by the name `tagValueName` does not exist.

REST End Point	/api/admin/tags/tag/value/description
Method	POST
Form Parameters	`tagName`– Name of the tag to be modified. `tagValueName` – Tag value name whose description needs to be changed. `tagValueDescription`– New tag value description to be set.
Return Type	void
Status Codes	204 400 500

editTagValueName

The **editTagValueName** method changes the tag value name from `oldTagValueName` to `newTagValueName` for a tag with the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists but a tag value by the name `oldTagValueName` does not exist.

REST End Point	/api/admin/tags/tag/value/name
Method	POST
Form Parameters	`tagName`– Name of the tag to be modified. `oldTagValueName`– Old tag value name. `newTagValueName`– New tag value name.
Return Type	void
Status Codes	204 400 500

enableAlertEx

The **enableAlertEx** method enables the specified alert.

REST End Point	/api/monitors/alert/status
Method	POST
Form Parameters	`fullPathToAlert`– A string specifying the full path to the alert to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the alert to enable/disable with elements of the path separated by the string "_ sis_path_delimiter_". `enable`– Set it to "true" for enabling the alert and set it to "false" or empty for disabling the alert.
Return Type	void
Status Codes	204 400 500

enableAssociatedAlerts

The **enableAssociatedAlerts** method enables the alerts associated with the given entity (Group or Monitor).

REST End Point	/api/monitors/monitor/alerts
Method	POST
Form Parameters	`fullPathToEntity` – A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the entity with elements of the path separated by the string "_sis_path_delimiter_". `enable` – Set to "true" to enable alerts and set to "false" or empty to disable alerts. `disableStartTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Required only for disabling the alerts. `disableEndTime`– The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Required only for disabling the alerts. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

enableGroupEx

The **enableGroupEx** method enables a group whether it was disabled indefinitely or for a specified time period. Enabling a group that is already enabled has no effect.

REST End Point	/api/monitors/group/status
Method	POST
Form Parameters	`fullPathToGroup`– A string specifying the full path to the group to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to be enabled/disabled with elements of the path separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Group is enabled if set to "true" and group is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the group should be disabled. If set to 0, group is disabled until explicitly enabled. Applicable only for disabling a group. `fromTime`– The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a group. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a group. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

enableGroupWithDescription

The **enableGroupWithDescription** method enables a group regardless of whether the group was disabled indefinitely, or for a specified time period.

REST End Point	/api/monitors/group/status
Method	POST
Form Parameters	`fullPathToGroup`– A string specifying the full path to the group to be enabled/disabled. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group to be enabled/disabled with elements of the path separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Group is enabled if set to "true" and group is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the group should be disabled. If set to 0, group is disabled until explicitly enabled. Applicable only for disabling a group. `fromTime`– The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a group. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a group. `description`– Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

enableMonitorEx

The **enableMonitorEx** method enables a monitor whether it was disabled indefinitely or for a specified time period. Enabling a monitor that is already enabled has no effect.

REST End Point	/api/monitors/monitor/status
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor to enable/disable. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor to enable/disable. The elements of the path are separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Monitor is enabled if set to "true" and monitor is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the monitor should be disabled. If set to 0, monitor is disabled until it is explicitly enabled. If `fromTime` and `toTime` are specified for disabling the monitor, the timePeriod value is ignored. Applicable only for disabling a monitor. `fromTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a monitor. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a monitor. `description` – Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

enableMonitorWithDescription

The **enableMonitorWithDescription** method enables a monitor with given description regardless of whether the monitor was disabled indefinitely, or for a specified time period.

REST End Point	/api/monitors/monitor/status
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor to enable/disable. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor to enable/disable. The elements of the path are separated by the string "_sis_path_delimiter_". Required for both enable and disable operations. `enable`– Monitor is enabled if set to "true" and monitor is disabled if set to "false" or if the string is empty. `timePeriod`– The duration (in seconds) for which the monitor should be disabled. If set to 0, monitor is disabled until it is explicitly enabled. If `fromTime` and `toTime` are specified for disabling the monitor, the timePeriod value is ignored. Applicable only for disabling a monitor. `fromTime` – The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). Applicable only for disabling a monitor. `toTime` – The time difference in milliseconds from the [current time] and the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). Applicable only for disabling a monitor. `description` – Description to be associated with enable/disable operation. `identifier`– Identifier to be associated with enable/disable operation and written to audit log.
Return Type	void
Status Codes	204 400 500

exportTemplate

The **exportTemplate** method exports the template.

REST End Point	/api/templates/template/export
Method	GET
Query Parameters	`templateFullPath` – A string specifying the full path to the template to be exported. If the template is to be exported from the SiteScope root directory the parameter must have only the name of the template to be exported, for example, "templateToExport". Else this parameter must start with the first template container under the SiteScope root and contain the full path to the template with the path elements separated by the string "_sis_path_ delimiter_", for example, "tc1_sis_path_delimiter_tc2_sis_path_delimiter_ templateToExport". `identifier=""` – Identifier to be associated with export and written to audit log.
Return Type	Base64 encoded and compressed (gzip) byte array containing exported template data.
Status Codes	200 400 500

getAcknowledgments

The **getAcknowledgments** method returns the acknowledgment data log of the given Entity.

REST End Point	/api/monitors/monitor/acknowledgements
Method	GET
Query Parameters	`fullPathToEntity`– A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the entity with elements of the path separated by the string "_sis_path_delimiter_". `identifier`– Identifier to be associated with the operation and written to audit log.
Return Type	A list of acknowledgments.
Status Codes	200 400 500

getAlertReport

The **getAlertReport** method returns the Alert Report URL for the monitor or group.

REST End Point	/api/admin/alertReport
Method	GET
Query Parameter	`fullPathToEntity`– A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root, and ends with the name of the entity. The individual path elements are separated by the string "_sis_path_delimiter_". `startTime` – Start time for the report. The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `endTime`– End time for the report. The time difference in milliseconds from the [current time] to the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). `identifier`: Identifier to be associated with the operation and written to audit log.
Returns	Alert report URL without base part.
Status Codes	200 400 500

getAlertSnapshots

The **getAlertSnapshots** method returns the corresponding snapshots for the alerts.

REST End Point	/api/monitors/alerts/snapshots
Method	GET
Query Parameters	`fullPathsToAlerts`– An array of alert paths to which snapshots are to be returned. The path to each alert is delimited using a semicolon (;), for example, alert1;alert2;alert3. Within each alert, multiple path elements must be separated by the string "_sis_path_delimiter_". For example: path1_sis_path_delimiter_path2_sis_path_delimiter_monitor1_sis_path_delimiter_alert1. `identifier`– Identifier to be associated with enable/disable operation and written to audit log. `propertiesToFilter`– Properties to filter. Comma separated list of properties to be filtered from returned snapshot response. Allowed properties for filtering: name, full_path, is_disabled.
Return Type	A map of the snapshots for the given alert paths
Status Codes	200 400 500

getAllTemplates

The **getAllTemplates** method gets all the templates.

REST End Point	/api/templates/export
Method	GET
Query Parameter	`identifier=""`– Identifier to be associated with export and written to audit log.
Return Type	A map containing snapshot of all templates.
Status Codes	200 400 500

getConfigurationSnapshotEx

The **getConfigurationSnapshotEx** method returns a map of the currently deployed entities in SiteScope together with basic properties for each entity. You can use the **SnapshotConfigurationVisitor** method to convert the map representation back to a tree-like representation of the result.

REST End Point	/api/admin/config/snapshot
Method	GET
Query Parameter	`fetchFullConfig` – Set to "true" to fetch full config with all entity properties; if set to "false" only basic config with basic entity properties are fetched.
Return Type	A map of the currently deployed entities in Sitescope with basic entity properties.
Status Codes	200 400 500

getConfigurationViaSourceTemplateEx

The **getConfigurationViaSourceTemplateEx** method returns a map of template variables to current values. Given a Template and a destination group under which the template has been deployed, returns the values that replace the template variables as the template is deployed in that group.

REST End Point	/api/templates/group/template/configuration
Method	GET
Query Parameters	`fullPathToDeployedGroup` – A string array specifying the full path to the group. The path starts with the first group under the SiteScope root directory and ends with the name of the deployed root group.
Return Type	A map of variables to values.
Status Codes	200 400 500

getConfigurationViaTemplateEx

The **getConfigurationViaTemplateEx** method returns a map of template variables to current values. Given a Template and a destination group under which the template has been deployed, returns the values that replace the template variables as the template is deployed in that group.

REST End Point	/api/templates/template/configuration
Method	GET
Query Parameters	`fullPathToTemplate` – A string array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. The elements of the path are separated by the string "_sis_path_delimiter_". `fullPathToDeployedGroup`– A string array specifying the full path to the group. The path starts with the first group under the SiteScope root directory and ends with the name of the deployed root group. The elements of the path are separated by the string "_sis_path_ delimiter_".
Returns	A map of variables to values.
Status Codes	200 400 500

getFullConfigurationSnapshot

The **getFullConfigurationSnapshot** method returns a map of the currently deployed entities in SiteScope together with all the entity's properties.You can use the **SnapshotConfigurationVisitor** method to convert the map representation back to a tree-like representation of the result.

REST End Point	/api/admin/config/snapshot
Method	GET
Query Parameter	`fetchFullConfig` – Set to "true" to fetch full config with all entity properties; if set to "false" only basic config with basic entity properties are fetched.
Return Type	A map of the currently deployed entities in Sitescope with basic entity properties.
Status Codes	200 400 500

getGroupsConfigurationSnapshot

The **getGroupsConfigurationSnapshot** method returns the corresponding snapshots for the group.

REST End Point	/api/admin/groups/config/snapshot
Method	GET
Query Parameters	`fullPathsToGroups`– An array of group paths to which snapshots should be returned. The path to each group is delimited with a semicolon (;), for example, group1;group2;group3. Within each group, multiple path elements should be separated by the string "_sis_path_ delimiter_", for example, path1_sis_path_delimiter_path2_sis_path_delimiter_path3. `isFullConfig`– Set it "true" if full group config is required else set to "false".
Return Type	A map of the currently deployed entities in selected group with basic entity properties.
Status Codes	200 400 500

getGroupSnapshots

The **getGroupSnapshots** method returns the corresponding snapshots for the given groups.

REST End Point	/api/monitors/groups/snapshots
Method	GET
Query Parameters	`fullPathsToGroups` – An array of group paths to which snapshots are to be returned. The path to each group is delimited using a semicolon (;), for example, group1;group2;group3. Within each group, multiple path elements must be separated by the string "_sis_path_ delimiter_", for example, path1_sis_path_delimiter_path2_sis_path_delimiter_path3. `identifier` – Identifier to be associated with enable/disable operation and written to audit log. `propertiesToFilter` – Properties to filter. Comma separated list of properties to be filtered from returned snapshot response. Allowed properties for filtering: name, full_path, type, description, updated_date.
Return Type	A map of the snapshots for the given group paths.
Status Codes	200 400 500

getHostsMap

The **getHostsMap** method returns a map of the hosts monitored by SiteScope.

REST End Point	/api/admin/hostsMap
Method	GET
Return Type	A map of the hosts monitored by SiteScope.
Status Codes	200 400 500

getMonitorSnapshots

The **getMonitorSnapshots** method returns the corresponding snapshots for the given monitors.

REST End Point	/api/monitors/snapshots
Method	GET
Query Parameters	`fullPathsToMonitors` – An array of monitor paths to which snapshots are to be returned. The path to each monitor is delimited using a semicolon (;), for example, alert1;alert2;alert3. Within each alert, multiple path elements must be separated by the string "_sis_path_ delimiter_". For example, path1_sis_path_delimiter_path2_sis_path_delimiter_monitor1_ sis_path_delimiter_alert1. `identifier` – Identifier to be associated with enable/disable operation and written to audit log. `propertiesToFilter` – Properties to filter. Comma separated list of properties to be filtered from returned snapshot response. Allowed properties for filtering: Allowed properties for filtering: name, full_path, type, target_ip, target_name, target_display_name, updated_date, description, is_disabled_permanently, disable_description, disable_start_time, disable_end_ Time, is_associated_alerts_disabled, associated_alerts_disable_description, associated_ alerts_disable_start_time, associated_alerts_disable_end_time, acknowledgment_ comment, status, availability, availability_description, summary, configuration_snapshot, runtime_snapshot.
Return Type	A map of the snapshots for the given monitor paths.
Status Codes	200 400 500

getQuickReport

The **getQuickReport** method returns the Quick Report URL for the monitor or group.

REST End Point	/api/admin/quickReport
Method	GET
Query Parameters	`fullPathToEntity`– A string specifying the full path to the entity. The path starts with the name of the first child under the SiteScope root directory, and ends with the name of the entity with individual path elements separated by the string "_sis_path_delimiter_". `startTime`– Start time for the report. The time difference in milliseconds from the [current time] and the required [start time]. For example, if the current time is 15:00:00 and the required start time is 15:10:00, the value that should be sent is [15:10:00] - [15:00:00] = 10*60*1000 (600000 milliseconds). `endTime`– End time for the report. The time difference in milliseconds from the [current time] to the required [end time]. For example, if the current time is 15:00:00 and the required end time is 15:30:00, the value that should be sent is [15:30:00] - [15:00:00] = 30*60*1000 (1800000 milliseconds). `identifier`– Identifier to be associated with the operation and written to audit log.
Return Type	Quick Report URL without base part.
Status Codes	200 400 500

getReadOnlyMode

The **getReadOnlyMode** method checks if SiteScope APIs are in read-only mode.

REST End Point	/api/admin/readMode
Method	GET
Return Type	"True" if SiteScope APIs are in read-only mode; otherwise it returns "False".
Status Codes	200 400 500

getSiteScopeMonitoringStatus

The **getSiteScopeMonitoringStatus** method returns the SiteScope monitoring status string.

REST End Point	/api/admin/monitors/status
Method	GET
Query Parameter	`identifier`– Identifier to be associated with the operation and written to audit log.
Returns	SiteScope monitoring status string. The returned value is one of: • MONITORING_PASSIVE__STARTUP. The initial state from the beginning of SiteScope startup until the monitoring engine starts. • MONITORING_ACTIVE. From the time the monitoring engine is active and monitors are running until SiteScope starts to shutdown. • MONITORING_PASSIVE__SHUTDOWN. From the beginning of SiteScope shutdown until the process exits.
Status Codes	200 400 500

getSiteScopeMonitoringStatusWithIdentifier

The **getSiteScopeMonitoringStatusWithIdentifier** method returns the SiteScope monitoring status string.

REST End Point	/api/admin/monitors/status
Method	GET
Query Parameter	`identifier`– Identifier to be associated with the operation and written to audit log.
Returns	SiteScope monitoring status string. The returned value is one of: • MONITORING_PASSIVE__STARTUP. The initial state from the beginning of SiteScope startup until the monitoring engine starts. • MONITORING_ACTIVE. From the time the monitoring engine is active and monitors are running until SiteScope starts to shutdown. • MONITORING_PASSIVE__SHUTDOWN. From the beginning of SiteScope shutdown until the process exits.
Status Codes	200 400 500

getSchedulePreferencesSnapshot

The **getSchedulePreferencesSnapshot** method retrieves all schedule preferences that are available in SiteScope.

REST End Point	/api/admin/config/snapshot/schedules
Method	GET
Query Parameters	`identifier` - Identifier to be written to audit log.
Return Type	A list of schedule preference details such as schedule type, ID, name, description, range, related entities, and related tags.
Status Codes	200 400 500

importSSHKey

The **importSSHKey** method imports the given SSH key file to SiteScope.

REST End Point	/api/admin/sshKeys
Method	POST
Form Parameter	`{File} sshKeyFile` – Binary representation of the SSH key file. `sshKeyFileName`– Identifier to be associated with the operation and written to audit log. `override`– Indicates if the existing SSH key with the given name should be overridden or not. `identifier`– Identifier to be associated with the operation and written to audit log.
Returns	void
Status Codes	200 400 500

importTemplate

The **importTemplate** method imports a template to SiteScope

REST End Point	/api/templates/templateImport
Method	POST
Form Parameters	`templateDestinationFullPath` – A string specifying the full path to the template container where the template needs to be imported. The path must start with the name of the first template container name under the SiteScope root and contain the full path with the path elements separated by the string "_sis_path_delimiter_", for example "tc1_sis_path_delimiter_tc2". `override` – Set it to "true" for the template at the specified path to be overridden; if set to "false" template is not overridden. `templateFile` – Binary template representation. Exported template via SiteScope UI.
Return Type	void
Status Codes	204 400 500

importTemplateWithOverride

The **importTemplateWithOverride** method imports an external template.

REST End Point	/api/templates/templateImport
Method	POST
Form Parameters	`templateDestinationFullPath` – A string specifying the full path to the template container where the template needs to be imported. The path must start with the name of the first template container name under the SiteScope root and contain the full path with the path elements separated by the string "_sis_path_delimiter_", for example "tc1_sis_path_delimiter_tc2". `override` – String ("true" or "false") that indicates if the template at the specified path should be overridden or not. `templateFile` – Binary template representation. Exported template via SiteScope UI.
Return Type	void
Status Codes	204 400 500

publishTemplateChanges

The **publishTemplateChanges** method publishes template changes to all deployed groups associated with the selected template.

REST End Point	/api/templates/publishedTemplate
Method	POST
Form Parameters	`pathToTemplate` – Path to template `connectToServer=false` – If set to true, the connection with server will be established during the publish. `deleteOnUpdate=false` – If set to true, delete on update will be allowed. `identifier` – Identifier to be associated with deployment and written to audit log. `Other parameters` – Group name as parameter and a map of template variables as the value. The map represents templateVariables specifying pairs of variableName and variableValue, separated with an equality sign (=), and each pair should be separated with a comma (,). These values will replace the names in the deployment.
Return Type	Publish result report
Status Codes	200 400 500

removeTagValue

The **removeTagValue** method removes tag value by the name `tagValueName` for a tag with the name `tagName`. An exception is thrown if: (i) the tag does not exist, or (ii) the tag exists, but a tag value by the name `tagValueName` does not exist, or (iii) an entity depends on it.

REST End Point	/api/admin/tags/tag/value
Method	DELETE
Query Parameters	`tagName`– Name of the tag from which a value is to be deleted. `tagValueName`– Name of the tag value that needs to be deleted.
Return Type	void
Status Codes	204 400 500

removeTagValuesFromMonitor

The **removeTagValuesFromMonitor** method removes tag values from a monitor.

REST End Point	/api/monitors/tags
Method	POST
Form Parameters	`fullPathToMonitor` – A string array specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with path elements separated by the string "_sis_path_delimiter_". `tagName` – The name of tag that holds the values. `tagValueNames` – The names of values to be checked in monitor. `active`– Set to "true" to make the tag values active. If set to empty or "false", the tag values are made inactive.
Return Type	void
Status Codes	204 400 500

runExistingMonitorEx

The **runExistingMonitorEx** method runs the monitor. The monitor must be deployed before invoking this method.

REST End Point	/api/monitors/monitor/run
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with path elements separated by the string "_sis_path_delimiter_". `timeOut`– Timeout value in milliseconds. `identifier` – Identifier to be associated with the operation and written to audit log.
Return Type	A HashMap representation of the status of the run and the status message as it would appear on the UI.
Status Codes	200 400 500

runExistingMonitorExWithIdentifier

The **runExistingMonitorExWithIdentifier** method runs the monitor. The monitor must be deployed before invoking this method.

REST End Point	/api/monitors/monitor/run
Method	POST
Form Parameters	`fullPathToMonitor`– A string specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with path elements separated by the string "_sis_path_delimiter_". `timeOut`– Timeout value in milliseconds. `identifier` – Identifier to be associated with the operation and written to audit log.
Return Type	A HashMap representation of the status of the run and the status message as it would appear on the UI.
Status Codes	200 400 500

runExistingMonitorsInGroup

The **runExistingMonitorsInGroup** method runs existing monitors in group.

REST End Point	/api/monitors/group/run
Method	POST
Form Parameters	`fullPathToGroup` – A string specifying the full path to the group whose monitors are to be run. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the group with path elements separated by the string "_sis_path_ delimiter_". `recursive`– Indicates if monitors in child groups are to be run recursively. `identifier`– Identifier to be associated with the operation and written to audit log.
Return Type	void
Status Codes	204 400 500

runToolOnMonitorEx

The **runToolOnMonitorEx** method runs the monitor configuration tool for specific monitors to help configure the monitor settings.

REST End Point	/api/monitors/monitor/tool/run
Method	POST
Form Parameters	`fullPathToMonitor`– A string array specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with path elements separated by the string "_sis_path_delimiter_". `resultAsHtml`– Indicates if result should be returned as HTML or plain string (for the relevant monitors only).
Return Type	A string result of the operation.
Status Codes	200 400 500

search

The **search** method gets the relevant elements (monitors, groups, or tags) according to the given search criteria. You can specify regular expressions in addition to plain text search strings. The method also allows to search for monitors and groups based on their tag names and values. The returned results include the entities of the selected entity_type (Monitors, Groups or Tags) that match ANY of the search criteria that are passed in the parameters name, path, target_name, target_display_name, status OR tags.

REST End Point	/api/monitors
Method	GET
Query Parameters	`entity_type` – Monitor/Group/Tag/empty string (for both monitors and groups). `name`– Monitor/Group/Tag name. `path`– Full path to the monitor with path elements separated by the string "_sis_path_ delimiter_". *Note*: One of the parameters either `name` or `path` must be provided and must match, else no results will be returned. `searchregex` - "true" or "false". If set to "true", all values passed in other search parameters are treated as regular expressions and the method searches for regular expression matches. The default value is "false" and in such a case, all parameter values will be treated as plain text. `tags` - Map of tag name value pairs to search monitor and groups. It is in the format `tagName:tagValue` with multiple tag name value pairs separated by commas. `status`– good/warning/error/empty string (for both monitors and groups). `target_name`– Monitor/Group target name. `target_display_name`– Monitor/Group target display name. `target_ip`– Monitor/Group target IP address. `maxNumOfResults`– Maximum number of returned search results. `identifier` – Identifier to be associated with enable/disable operation and written to audit log.
Return Type	A map of the entity path. Key is the path of the entity with `_sis_path_delimiter_` as the delimiter. Value is the type of entity (Monitor, Group, or Tag).
Status Codes	200 400 500

setReadOnlyMode

The **setReadOnlyMode** method sets SiteScope API to read-only mode. The only configuration changes allowed in this mode are **getConfiguration** and **runExistingMonitors**.

REST End Point	/api/admin/readMode
Method	POST
Form Parameter	`readOnly`– Specify "True" to set SiteScope APIs to read-only mode or "False" otherwise.
Return Type	void
Status Codes	204 400 500

updateMonitorViaTemplateEx

The **updateMonitorViaTemplateEx** method updates a single monitor deployed by a template with new variables.

REST End Point	/api/monitors/monitor/properties
Method	POST
Form Parameters	`fullPathToTemplate` – A string specifying the full path to the template. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template with path elements separated by the string "_sis_path_delimiter_". `fullPathToDeployedMonitor`– A string specifying the full path to the monitor. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the monitor with path elements separated by the string "_sis_path_delimiter_". `Other Parameters` – All actual variable names versus values that need to be replaced in the deployed monitor.
Return Type	void
Status Codes	204 400 500

updateTemplate

The **updateTemplate** method enables you to update a template.

REST End Point	/api/templates/template
Method	POST
Form Parameters	`fullPathToTemplate` – A string specifying the full path to the template. The path starts with the name of the first child under the SiteScope root directory and ends with the name of the template with path elements separated by the string "_sis_path_delimiter_". `properties` – contains the properties to be updated. "templateName" – string, the name of the template to be updated. **Note**: Currently only the `templateName` property is supported. `identifier` – Identifier to be written to audit log.
Return Type	void
Status Codes	204 400 500

updateViaSourceTemplateEx

The **updateViaSourceTemplateEx** method updates a group of entities that were created with a template deployment operation.

REST End Point	/api/templates/group/template/configuration
Method	POST
Form Parameters	`fullPathToDeployedGroup` – A string array specifying the full path to the group. The path starts with the first group under the SiteScope root directory and ends with the group where the template was deployed. The path elements are separated by the string "_sis_path_ delimiter_". Other parameters – The templateVariables, with variable name as parameter and the variable value as the parameter value.
Return Type	void
Status Codes	204 400 500

updateViaTemplateEx

The **updateViaTemplateEx** method updates a group of entities that were created with a template deployment operation.

REST End Point	/api/templates/template/configuration
Method	POST
Form Parameters	`fullPathToTemplate` – A string array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. The elements of the path are separated by the string "_sis_path_delimiter_". `fullPathToDeployedGroup` – A string array specifying the full path to the group. The path starts with the first group under the SiteScope root and ends with the name of the deployed root group. The elements of the path are separated by the string "_sis_path_delimiter_". `Other parameters`– The templateVariables with variable name as parameter and the variable value as the parameter value.
Return Type	void
Status Codes	204 400 500

updateViaTemplateWithRootGroupEx

The **updateViaTemplateWithRootGroupEx** method updates the template deployment to use the new variables. The full path to the deployed group should point to a root group.

REST End Point	/api/templates/rootGroup/template/configuration
Method	POST
Form Parameters	`fullPathToTemplate` – A string array specifying the full path to the template. The path starts with the name of the root template container and ends with the name of the template. The elements of the path are separated by the string "_sis_path_delimiter_". `fullPathToDeployedGroup` – A string array specifying the full path to the root group. The elements of the path are separated by the string "_sis_path_delimiter_". `Other parameters` – The templateVariables with variable name as a parameter and the variable value as the parameter value.
Return Type	void
Status Codes	204 400 500

Chapter 5: Data Acquisition APIs

The data acquisition API can be used for querying historical data for the following:

- Retrieve monitor runs matching the specified query parameters.
- VMware reconciliation topology collected by VMware monitors currently running on SiteScope.
- Return a list of monitor types together with the metric names per monitor type for which user has view permissions.

getData

The **getData** method gets historical data for monitor runs matching the specified query parameters. The data is taken from the SiteScope daily log.

REST End Point	/api/data
Method	GET
Query Parameters	`startTime`(Mandatory) – Specify the start time to fetch historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). `endTime` (Mandatory) – Specify the end time to fetch historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). `monitorType` – Monitor types for which data needs to be fetched. Specify the Topaz name of the monitor. To specify multiple monitor types, separate them by commas, for example, "CPU,Memory,Directory". `targetServer` – Server names monitored by SiteScope for which data needs to be fetched. To specify multiple target servers, separat them by commas. `bsmId` – Monitor BSM IDs for which to get data. To specify multiple BSM IDs, separate them by commas. `monitorName` – Monitor names for which data needs to be fetched. Monitor name appears in the general settings of the monitor properties. To pass several monitor names, separate them by commas. `granularity` – Granularity of the data in seconds. Data samples for every [DATA_ GRANULARITY] seconds will be listed in the response. `vmwareMonitorTopology` – Whether to fetch reconciliation topology for VMware monitors. The data is taken from the SiteScope daily log. The reconciliation topology is collected by VMware monitors currently running on SiteScope. Reconciliation topology for monitors that existed in the specified time frame but no longer exist at the time the request is made is not available in the response.
Return Type	Base64 encoded and compressed (gzip) XML containing the requested data.
Status Codes	200 400 500

getDataWithTopology

The **getDataWithTopology** method gets historical data for monitor runs matching the specified query parameters, with reconciliation topology for VMware monitors. The data is taken from the SiteScope daily log. The reconciliation topology is collected by VMware monitors currently running on SiteScope. Reconciliation topology for monitors that existed in the specified time frame but no longer exist at the time the request is made is not available in the response.

Reconciliation topology matching the above constraints includes:

- Details of VMware objects referenced in the counters of the VMware monitors whose run data is within the specified time frame.
- Links between the above VMware objects.
- References between the VMware objects and the counters in the run data.

REST End Point	/api/data
Method	GET
Query Parameters	`startTime`(Mandatory) – Specify the start time to fetch historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). `endTime` (Mandatory) – Specify the end time to fetch historical data (in milliseconds since January 1, 1970, 00:00:00 GMT). `monitorType` – Monitor types for which data needs to be fetched. Specify the Topaz name of the monitor. To specify multiple monitor types, separate them by commas, for example, "CPU,Memory,Directory". `targetServer` – Server names monitored by SiteScope for which data needs to be fetched. To specify multiple target servers, separat them by commas. `bsmId` – Monitor BSM IDs for which to get data. To specify multiple BSM IDs, separate them by commas. `monitorName` – Monitor names for which data needs to be fetched. Monitor name appears in the general settings of the monitor properties. To pass several monitor names, separate them by commas. `granularity` – Granularity of the data in seconds. Data samples for every [DATA_ GRANULARITY] seconds will be listed in the response. `vmwareMonitorTopology` – Whether to fetch reconciliation topology for VMware monitors. The data is taken from the SiteScope daily log. The reconciliation topology is collected by VMware monitors currently running on SiteScope. Reconciliation topology for monitors that existed in the specified time frame but no longer exist at the time the request is made is not available in the response.
Return Type	Base64 encoded and compressed (gzip) XML containing the requested data.
Status Codes	200 400 500

getMonitorTypesWithMetricNames

The **getMonitorTypesWithMetricNames** method scans all the monitors in this SiteScope instance for which the user has view permissions, and returns a list of their types together with the metric names per monitor type.

The list of metric names is merged from all the monitors of each type (repeated occurrences are removed). Where enabledMonitorsOnly is true, it scans enabled monitors only. Where enabledMonitorsOnly is false, it scans all monitors (enabled/disabled) in the SiteScope instance.

REST End Point	/api/data/monitorTypes
Method	GET
Query Parameters	`enabledMonitorsOnly` – If true only enabled monitors are scanned.
Return Type	Base64 encoded and compressed (gzip) XML containing the requested data.
Status Codes	200 400 500

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on SiteScope Public API Reference Guide (SiteScope 11.32)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to sitescope-doc-feedback@hpe.com.

We appreciate your feedback!