Dell EMC™ Unisphere for PowerMax™

Version 9.0.0

REST API Concepts and Programmer's Guide

REV 01

Copyright © 2014-2018 Dell Inc. or its subsidiaries. All rights reserved.

Published May, 2018

Dell believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS-IS." DELL MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. USE, COPYING, AND DISTRIBUTION OF ANY DELL SOFTWARE DESCRIBED IN THIS PUBLICATION REQUIRES AN APPLICABLE SOFTWARE LICENSE.

Dell, EMC, and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be the property of their respective owners. Published in the USA.

Dell EMC
Hopkinton, Massachusetts 01748-9103
1-508-435-1000 In North America 1-866-464-7381
www.DellEMC.com

CONTENTS

TABLES

PREFACE

As part of an effort to improve its product lines, Dell EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your technical support professional if a product does not function properly or does not function as described in this document.

Note

This document was accurate at publication time. Go to Dell EMC support website or the technical documentation page to ensure that you are using the latest version of this document.

Purpose
This document describes how to configure and use Unisphere REST API. The term Unisphere incorporates "Unisphere for PowerMax" for the management of PowerMax and All Flash storage systems using PowerMaxOS 5978, and "Unisphere for VMAX" for the management of VMAX All Flash and VMAX storage systems using HYPERMAX OS 5977 and Enginuity OS 5876.

Audience
This guide is intended for programmers who want to create interfaces, using REST APIs, in any of the programming environments that support standard REST clients such as web browsers and programming platforms that can issue HTTP requests.

Related documentation
This section lists the resources available for the Unisphere REST API.

- The Unisphere for VMAX REST API Information Hub has information on how to get started with Unisphere REST API.
- Change log — This document lists the changes to the Unisphere REST API in each release. It can be found by going to the Dell EMC support website and searching for "REST API Change Log". It can also be found on the Dell EMC Community page in the "Developer Resources" area.
- Enunciate — This is the engine used to generate the Unisphere REST API. More information can be found here.
- Dell EMC Open Source Community
- Unisphere-specific examples:
- PYU4V

The following publications provide additional information:

- *Dell EMC Unisphere for PowerMax Release Notes*
- *Dell EMC Unisphere for PowerMax Installation Guide*

Where to get help
Dell EMC support, product, and licensing information can be obtained as follows:

Product information

For documentation, release notes, software updates, or information about Dell EMC products, go to Dell EMC support website.

Technical support

Go to the Dell EMC support website and click Service Center. You will see several options for contacting Technical Support. Note that to open a service request, you must have a valid support agreement. Contact your sales representative for details about obtaining a valid support agreement or with questions about your account.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to content feedback.

CHAPTER 1

Introduction

This chapter contains the following:

Overview and Architecture

This guide is intended for programmers who want to create interfaces, using REST APIs, in any of the programming environments that support standard REST clients such as web browsers and programming platforms that can issue HTTP requests.

The term Unisphere incorporates "Unisphere for PowerMax" for the management of PowerMax and All Flash storage systems using PowerMaxOS 5978, and "Unisphere for VMAX" for the management of VMAX All Flash and VMAX storage systems using HYPERMAX OS 5977 and Enginuity OS 5876.

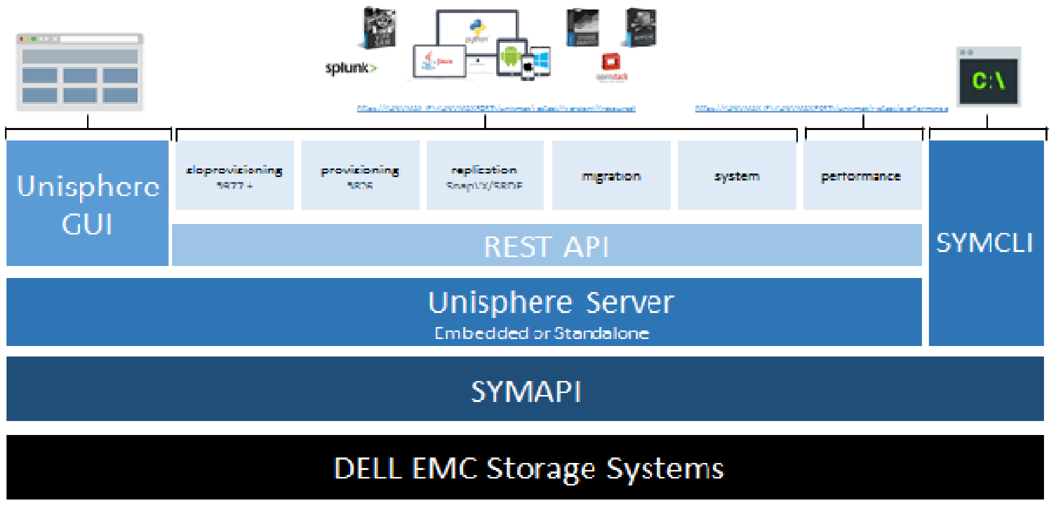
The REST (Representational State Transfer) API allows for accessing diagnostic performance data, accessing configuration data, and performing storage provisioning operations for storage system hardware through robust APIs. These APIs can be used in any of the programming environments that support standard REST clients such as web browsers and programming platforms that can issue HTTP requests. The REST API supports both XML and JavaScript Object Notation (JSON) MIME types.

Unisphere is designed around the concept of the storage group for management of applications. A storage group is a container for a set of devices that are related in some way and are usually representative of an application or a tenant with data on the storage array. The majority of operations using REST API are built to take advantage of the storage group as a control entity, for example, SnapVX, and SRDF. Using a storage group for management reduces the number of objects that storage administrators need to manage. A storage group must exist for masking devices associated with a host (see Provisioning and managing storage using REST API on page 12).

REST API utilizes numbered versioning. The version of the API call is specified as part of the URL, for example, *https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/90* is the base URL for the Unisphere 9.0 array management and monitoring calls. If no version is specified and the endpoint URL is valid, it is processed using the oldest supported version of the REST API.

When beginning to use REST API, it is recommended that you use the most recent version of the REST API supported by your Unisphere server.

As outlined in the following figure, REST API resides at the same level in the application stack as the Unisphere GUI above the Unisphere server.



REST API is subdivided into a number of distinct resources which provide distinct endpoints for operations that execute on the Unisphere server and the storage systems.

The resources and their functionality are:

Service level (SLO) provisioning for storage systems and all flash storage systems running HYPERMAX OS 5977 and higher (for Open Systems and Mainframe)

Provisioning for storage systems and all flash storage systems running HYPERMAX OS 5977 and higher (for Open Systems and Mainframe)

System - management of alerts and jobs

Replication - Provides Storage Group Level Management of SRDF and TimeFinder SnapVX

Migration - Control of Non-Disruptive Migration functionality, for live migration of running workloads from storage groups running Enginuity OS 5876 and storage groups running HYPERMAX OS 5977 to storage groups running HYPERMAX OS 5977 and PowerMaxOS 5978.

Performance - Gathering of performance statistics of storage array components, storage groups, and configuration of performance threshold alerts.

REST model

REST is a manner of building networked software systems that allows merging of documents, data and information services into an ecosystem of named resources.

- Resource - Anything important enough to be referenced as a noun.
- Resource name - Unique identification of the resource.
- Resource representation - Useful information about the current state of resource. Resource representation is what is used to transfer its information to the client using the HTTP protocol.
- Resource link - Link to another representation of the same resource.
- Resource interface - Uniform interface for accessing the resource and manipulating its state.

The following table describes the standard resource methods.

Table 1 REST resource methods

HTTP operation	Description	Access to Method
PUT	Modify resource	Administrator Storage Administrator
GET	Retrieve state of resource	All roles
POST	Create a new resource	Administrator Performance Monitor (for Performance URLs only) Storage Administrator
DELETE	Delete resource	Administrator Storage Administrator

For assigning user roles for each array, refer to Setting up user authentication on page 28.

REST client

The REST client provides both a front-end GUI for the REST API documents and a Unisphere-specific REST client based on the WADL file received from the Unisphere host. The REST client enables you to visualize the REST schema and helps to identify the calls you need to build scripts and code functions. The REST client, while extremely useful is not a replacement for the documentation, as the documentation has a greater level of detail.

For more information, refer the REST Client information which can be accessed by clicking here and searching for "REST client".

Provisioning and managing storage using REST API

Provisioning storage using the REST API follows the same rules and logic as provisioning using the Unisphere GUI. All end points for storage provisioning on storage systems running HYPERMAX OS 5977 and higher are located under

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/
```

You do not need to create storage volumes prior to provisioning the storage, because REST API is using the logic of the Unisphere server. When storage volumes are requested for a given storage group, the Unisphere server first checks for unused volumes in the system that match the capacity and type and uses these. If no volumes exist, new volumes are created to match the required capacity.

Storage is provisioned to a host via a masking view. A masking view is comprised of the following three components which need to exist already or be created before being combined in the masking view.

- Storage Group
- Host or Host Group
- Port Group

The REST API caters for all of the above steps in a single REST API call with a payload package that creates each component and combines them into a masking view.

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/symmetrix/{symmetrixId}/maskingview
```

Adding additional storage volumes to a host or system once the masking view is already in place is perfomed using a PUT call to the storage group resource object. This call also has a number of action parameters that allow the expansion of a specific volume in the group or that allow expansion of all volumes in the group to a specified size.

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/symmetrix/{symmetrixId}/storagegroup/
{storageGroupId}
```

Device expansion is also supported on per volume level with a PUT call.

```
 https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/symmetrix/{symmetrixId}/volume/{volumeId}
```

Snapshot management using REST API

REST API supports Timefinder SnapVX Snapshots. These are space efficient with a low footprint. Snapshots are managed at a storage group level.

All of the endpoints for SnapVX are under

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
replication/symmetrix/{symmetrixId}/storagegroup/{storageGroupId}/
snapshot
```

Creating a snapshot is a POST call to the endpoint. If you send multiple POST calls with the same payload, multiple generations of the same snapshot are created. The latest version of a SnapVX snapshot will always be generation 0.

Managing Snapshots with REST API requires the use of the snapshot generation. If you want to do any PUT or DELETE operations, you need to consider this. These operations are located under the following resource endpoint:

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{vesion}/
replication/symmetrix/{symmetrixId}/storagegroup/{storageGroupId}/
snapshot/{snapshotId}/generation/{generationNumber}
```

Linking a snapshot is the operation that makes snapshot data visible via a host accessible volume, and is a PUT call. REST API has automated the process of volume creation and storage group creation if you specify a storage group for the link that doesn't already exist. This helps simplify workflows.

Workflow to link a snapshot:

- Identify snapshot on the source storage group (GET)

  ```
  https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{vesion}/
  replication/symmetrix/{symmetrixId}/storagegroup/
  {storageGroupId}/snapshot
  ```

- If multiple generations identify the correct timestamp (GET)

  ```
  https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{vesion}/
  replication/symmetrix/{symmetrixId}/storagegroup/
  {storageGroupId}/snapshot/{snapshotId}/generation/
  {generationNumber}
  ```

- Link to new storage group (PUT)

  ```
  https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{vesion}/
  replication/symmetrix/{symmetrixId}/storagegroup/
  {storageGroupId}/snapshot/{snapshotId}/generation/
  {generationNumber}
  ```

- Create Masking View for the target storage group (POST)

```
 https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/symmetrix/{symmetrixId}/maskingview
```

You should create snapshots with a time to live value to ensure that they expire automatically. Scripted snapshots on a rotating basis should also be monitored. Ensure alerting and notifications are setup as documented in the SnapVX technical notes.

If you are planning a high volume of SnapShots at frequent intervals, it is recommended to check with your local storage system performance expert to review prior to implementing.

Snapshots do not auto-expire if they are linked.

For releases Unisphere 9.0 and later, the local replication role is assigned at the storage group level. This privilege ensures that the assigned user can create and terminate their own snapshots. Link, unlink, and Restore operations require additional privileges.

For detailed information on SnapVX, refer to the SnapVX technical notes.

Protecting Storage with SRDF using REST API

In order to remotely replicate the storage system, SRDF emulations must exist and a pair of front end ports must be mapped to the SRDF emulation. Zoning or IP routes must already exist to the remote array.

A lot of automation has been put into the SRDF replication workflows in Unisphere to simplify, not only the GUI management, but also the management flows using REST API. REST API fully supports all modes of SRDF operation and storage groups can be protected in a single call. The workflow for protecting a storage group with SRDF is as follows:

- Verify source array is SRDF capable
- Verify target array is visible and connected over SRDF links
- Protect the source storage group using a POST call to

```
  https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
replication/symmetrix/{symmetrixId}/storagegroup/
{storageGroupId}/rdf_group
```

- Optional: Mask the storage group on the remote storage system to a host for visibility to remote storage once data has been synchronized.

The POST call to protect the storage group with SRDF performs a number of operations depending on the mode of SRDF operation chosen for the protection.

- Automatic creation of devices on the target array to be paired with the sources.
- If Metro or Asynchronous mode is specified in the payload, a new SRDF group is created between the source and destination arrays if an unused one doesn't already exist. This ensures that all devices are managed together.
- If synchronous mode is set, the devices are added to the least loaded SRDF group between the source and target arrays.
- If protecting with SRDF Synchronous mode and you want to ensure that the devices are placed in a separate group, the POST call provides optional parameters to do this. Refer to the documentation or the REST Client for more details.

Note

There are 250 SRDF groups in total per array, so design your system with these limits in mind. Each SRDF group can have up to 64K volumes.

Managing SRDF replicated storage using REST API

Once storage has been protected with SRDF, there is little interaction with SRDF Replication other than to perform disaster recovery tests. However there are a number of control operations which are possible and fully supported using REST API. In order to manage SRDF protected storage, you need to know the SRDF group that your storage group is in.

Workflow for adding new volumes to existing SRDF session in Unisphere 9.0 and higher (Refer to REST API change log or REST API client for payload parameters):

1. Create a new storage group adding the required number of new devices.
2. Use the POST call to protect

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
replication/symmetrix/{symmetrixId}/storagegroup/staging_sg/
rdf_group
```

3. Perform lookup on the staging_sg created to find the SRDF group number (GET)

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/replication/symmetrix/
{symmetrixId}/storagegroup/{storageGroupId}/rdf_group
```

4. Perform lookup on the target_sg to find the SRDF group number (GET)

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/replication/symmetrix/
{symmetrixId}/storagegroup/{storageGroupId}/rdf_group
```

5. Suspend the replication on the staging_sg devices with a PUT call

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
replication/symmetrix/{symmetrixId}/storagegroup/
{storageGroupId}/rdf_group/{rdfgNumId}
```

6. Move the volumes from the staging SRDF group number to the existing one with a PUT call using consistency_exempt if target group is SRDF/A.

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
replication/symmetrix/{symmetrixId}/rdf_group/{rdfgNumId}
```

7. Move volumes from staging storage group to production storage group with a PUT call.

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
sloprovisioning/symmetrix/{symmetrixId}/storagegroup/
{storageGroupId}
```

8. Repeat step 7 for the storage group on the target array

9. Remove storage group created for staging SRDF volumes on source and target array with a Delete call.

    ```
    https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/{version}/
    sloprovisioning/symmetrix/{symmetrixId}/storagegroup/
    {storageGroupId}
    ```

The recommended practices for REST API and SRDF are as follows:

SRDF protection can take time as there is a lot going on in the background. SRDF POST calls are ideal candidates to be executed asynchronously.

Large data transfers and initial sync should be completed in Adaptive copy mode; switch to primary mode SRDF/Sync or Async when close to synchronized. Specify the optional parameter ,"establish":false when protecting with a POST call. If protecting storage groups at the time of creation, since data doesn't already exist, there should be no reason to delay the synchronization or switch mode of operation.

To actually start the synchronization, or modify the groups state of the data, you can use a PUT call to modify SRDF mode to adaptive copy and begin synchronization; when close to synchronized, a second PUT call can be used to switch SRDF modes back to Sync or Async. This will ensure normal operation.

Recommended Practices for Gathering Performance Metrics using REST API

Performance calls in Unisphere 8.4.0 and greater are no longer versioned. All performance REST API calls are under

```
https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/performance
```

If new metrics are added and you wish to collect them, the calls are added under a new endpoint under this base URL.

In order to query performance metrics, the array must first be registered for performance data collection using the Unisphere user interface. To find out what arrays are registered for performance statistics with the instance of Unisphere, a GET call can be made to

```
 https://{UNIVMAX_IP}:{UNIVMAXPORT}/univmax/restapi/performance/
Array/keys
```

Each sub-resource under the performance API calls also provides a "keys" endpoint to return a list of valid values. For example, to get a list of the front end directors for a particular array in order to gather metrics for each, you can make a single call to the

```
/performance/FEDirector/keys
```

endpoint and pass the serial number for the storage array. This returns a list that can be parsed to extract the directorID which can be used to build the URL for additional calls. Using keys to dynamically build your performance calls ensures that you do not make calls for objects that will return no data.

All performance calls accept a payload that requests multiple metrics. This is more efficient as there is no need to make multiple requests to the same resource for multiple metric values.

Gathering performance statistics from Unisphere REST API is always a POST (Create) call. This may seem counter intuitive, however the REST call constructs a return object based on the JSON payload parameters and the server responds with this object as a JSON object containing the requested data.

Performance metrics are calculated and stored at 5 minute intervals, so any REST API scripts should be run at an interval of 5 minutes or higher. Device level performance stats are not accessible via REST API calls; if device level metrics are required, these can be accessed using the Unisphere user interface.

REST API calls for performance data require that date/times are specified in epoch/ Unix time (the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, not counting leap seconds). All timestamps are in milliseconds, so you need to take this into account when building the payload start_date and end_date.

Code Examples

Sample code for the main use cases have been posted here.

New in this release

The complete Unisphere REST API change log can be found by going to the Dell EMC support website and searching for "REST API Change Log". It can also be found on the Dell EMC Community page in the "Developer Resources" area.

Unisphere REST API for 9.0 has the following changes:

Performance - Addition of 9.0 metrics to the following categories: IP interface, iSCSI target, Storage Group and Array.

Service Level (SLO) Provisioning - Additional, URL, or parameters, or both. Removal of Director and Director-port implementations.

Replication:

- Addition of filtering for GET list lookup of SRDF groups
- Addition of PUT action to allow Move Pairs functionality for SRDF Groups
- Addition of WWNs for SRDF ports
- Modification of StorageGroupRdfg attribute names

System:

- Addition of iSCSI IP Route implementation for GET(Object), GET(List), POST and DELETE
- Addition of Director implementation for GET(Object), GET(List)
- Addition of iSCSI Target (Port) implementation for GET(Object), GET(List), POST, PUT and DELETE
- Addition of IP Interface implementation for GET(Object), GET(List), POST, PUT and DELETE

CHAPTER 2

Documentation

This chapter contains the following:

Downloading documentation

Before you begin

Before downloading REST API documentation, Unisphere must be installed and user access roles must be setup (see Setting up user authentication on page 28).

Procedure

1. Point the browser to: `https://{UNIVMAX_IP}:{UNIVMAX_PORT}/univmax/restapi/docs`

   where *UNIVMAX_IP* is the IP address and *UNIVMAX_PORT* is the port of the host running Unisphere.
2. Copy the Zip file (restapi-docs.zip) locally, unzip the file, and navigate to target/docs/index.html.
3. To access the documented resources, open the html file.

Transport protocol

REST API uses the Hypertext Transfer Protocol Secure (HTTPS) version 1.1 as the transport for API requests. For PUT and POST requests, method arguments are passed in the HTTP Request message body. API methods return standard HTTP status codes and result content in the HTTP Response message body.

REST API supports two media-types:

- application/xml—Allows marshalling/un-marshalling using XML
- application/json—Allows marshalling/ un-marshalling using JSON

Product version and compatibility

The Unisphere REST API is the same version as the Unisphere application.

Backward compatibility

Since the REST API is a method-based API, the methods remain backwards compatible with older versions as long as the argument and result types have only optional properties added to newer versions of the API.

If additional properties are required or existing properties require modification, an additional method should be created. Deprecated methods will be maintained to service older clients, and eventually removed. New REST API methods are backwards compatible.

The latest versioning endpoints are accessed through /restapi/90 and requests made to URI's at the earlier endpoint /84 will continue to operate. Your scripts will work with updates to access the newer versions however if you want to take advantage of new data returned in /90/ rest calls then scripts will need to be updated to access the new keys. Please see the Unisphere REST API change log for full details.

REST API supports backwards compatibility for two major releases.

Versioning is handled in two different ways:

- For the following resources, REST API guarantees backwards compatibility for two major releases by versioning the URIs:

- DSA
- Migration
- WLP
- VVol
- System
- Provisioning
- Sloprovisioning
- Replication
- Migration

- For the following resources, REST API guarantees backwards compatibility for two major releases by versioning the attributes:
  - Common
  - Performance

Client/Server compatibility

If the Unisphere REST client API version is older than the server version, interaction proceeds as designed. If the client REST API version is newer than the server version, the server delivers an HTTP status 400 or HTTP status 404, indicating an illegal request.

Making the JSON root element backwards compatible

For prior versions of the REST API, the JSON root is included in the API request and response. The current version of REST API does not include the JSON root in the API request and response. To support backwards compatibility, the *JSON root* property must be modified as follows:

Procedure

1. Stop the SMAS service.
2. Remove `<location of Unisphere installation>\jboss\domain\deploy0` directory (and deploy1, deploy2 etc if they exist).

   Note

   Do not remove deploy directory.

3. Locate `<location of Unisphere installation>\jboss\domain\deploy\smc.ear\smc.war\WEB-INF\web.xml`.
4. Open the file and set the property *com.emc.em.restapi.WrapJsonRoot* to "true".
5. Start SMAS.

Example 1 REST API request and response example

The following example shows a REST API request and response with and without the JSON root:

Example 1 REST API request and response example (continued)

Request:

`https://<`*`UnimaxIP`*`:`*`UnimaxPort`*`>/univmax/restapi/90/sloprovisioning/symmetrix/000195900063/storagegroup`

`UNIVMAX_IP` is the IP address and *`UNIVMAX_PORT`* is the port of the host running Unisphere.

Response without JSON root (default):

```
{
"num_of_storage_groups": 64,
"storageGroupId":
[
"App2_SG",
"App2_SG",
"Finance2_SG",
"Database1_SG",
],
"success": true
}
```

Response with JSON root:

```
{
"listStorageGroupResult":
{
"num_of_storage_groups": 64,
"storageGroupId":
[
"App2_SG",
"App2_SG",
"Finance2_SG",
"Database1_SG",
],
"success": true
}
}
```

CHAPTER 3

Iterators

This chapter contains the following:

How iterators are used in REST API

An iterator is an object that allows you to cycle through the elements in a collection and display each element. REST API uses iterators to retrieve a collection of performance data only when the API interface defines that there are unbounded instances in the result. For example, the method to retrieve array metrics takes an ArrayParamType argument and returns an unbounded number of ArrayResultType results.

See Downloading documentation on page 20 for instructions on how to access the REST resources. From the REST API home page, select Performance, then any metrics resource, and under Response Body select Iterator to view how the iterator cycles through the elements to collect the results.

The REST API also uses iterators to control resource consumption on the server when multiple clients are querying for large sets of data. Each iterator has a unique ID and provides two ways to control excess server consumption: maximum page size, and an expiration time for the iterator instance. The page size and expiration time are dependent on server load and resource availability. In addition, Unisphere imposes limits on the number of concurrent requests it will accept. Refer to Sample iterator methods and typical use case on page 24.

Sample iterator methods and typical use case

The infrastructure provides three method endpoints for clients to interact with the iterators.

- Get iterator info
- Get iterator page
- Delete iterator

A typical use case is as follows:

Note

If a client becomes unresponsive, or disconnects from the Unisphere server, the iterator expires, based on a pre-configured timeout (10 minutes). When the iterator expires, server resources are released. If the iterator expires before the client has finished paging over all the results, the client must reissue the request.

1. The client invokes a REST API method that returns an iterator.
2. The client pages over the results of an existing iterator until all results are retrieved by issuing

   ```
   Get common /Iterator/{iteratorld}/page
   ```

   .
3. If the iterator has not expired, the client deletes the iterator by issuing

   ```
   Delete common/Iterator/{iteratorld}
   ```

CHAPTER 4

Sample

This chapter contains the following:

Setting up user authentication

Before using the REST API, user authorization must be assigned for each storage array that the user is permitted to access.

Procedure

1. Log in to Unisphere.
2. Select ⚙ to display the **Settings** panel.
3. Select **Users and Groups > Authorized Users and Groups** to open the **Authorized Users and Groups** list view.
4. Click **Create** to open the **Add New Authorization Rule** dialog box.
5. Select the **Roles** tab and create a user login profile for each storage array to be accessed by the user, and assign them to any of the following roles:
   - Users that are allowed storage array access:
     - Administrator—Can initiate GET, POST, PUT, and DELETE methods.
     - Storage Administrator—Can initiate GET, POST, PUT, and DELETE methods.
     - Performance Monitor—Can initiate GET and POST (for Performance URLS only) methods.
     - Monitor—Can initiate GET methods.
     - Auditor—Can initiate GET methods.
     - Security Administrator—Can initiate GET methods.
     - Local Replication (LREP)— A user with LREP can perform the following operations on storage groups for which they have been LREP rights, either all the storage groups on the storage array or a subset of storage groups:
       - POST - Create snapshots for a storage group (cannot specify 'Secure' - Storage Administrator rights are needed for that)).
       - PUT - Manage snapshots on a storage group as specified in the following list of rules with the exception of setting Secure (Storage Administrator rights are needed for that.)
         - a. If restoring the user must also have device manage rights on the storage group
         - b. If Linking, the storage group must already exist on the storage array and the user must also have device manage rights for that storage group .The LREP user cannot create a storage group (Storage Administrator rights are needed for that), similarly for ReLinking and Unlinking.
         - c. Set Mode - Must have device management rights on linked storage group

       DELETE on snapshot generation.
     - Remote Replication (RREP)— A user with RREP can peform the following operations on storage groups for which they have been RREP rights, either all the storage groups on the storage array or a subset of storage groups:

- PUT - Manage remote replication for a storage group - establish, suspend
- Delete storage group pairings

A user with RREP cannot create storage group pairings as this action requires the creation of a storage group and possibly a SRDF group and these actions need Storage Administrator rights.

All operations on an SRDF group also need (at a minimum) Storage Administrator rights.

- Users that are not allowed storage array access:
  - None

6. Click **OK**.

Client authentication

Note

The minimum supported TLS version is 1.2.

REST API uses HTTP Basic Access Authentication to authenticate API clients, described in RFC 2617. HTTP Basic Access Authentication is simple; it does not require cookies, session handling, or login pages. Instead, HTTP Basic Access Authentication uses static headers, requiring no handshakes.

Users of the REST API are assigned user credentials for associated storage arrays within Unisphere . A username and password is supplied with every request to REST API in the “Authorization” header, as specified in RFC 2617. Every request to REST API is authenticated and authorized.

Note

If X509 certificate-based client authentication is enabled, Unisphere ignores the username/password REST basic authentication credential. However, to meet REST standard’s basic authentication requirement, REST clients must provide a non-empty username/password credential of their own creation.

Sample client request and response examples

For sample client request and response examples, refer to the REST API documentation. Downloading documentation on page 20 describes how to obtain the documentation.