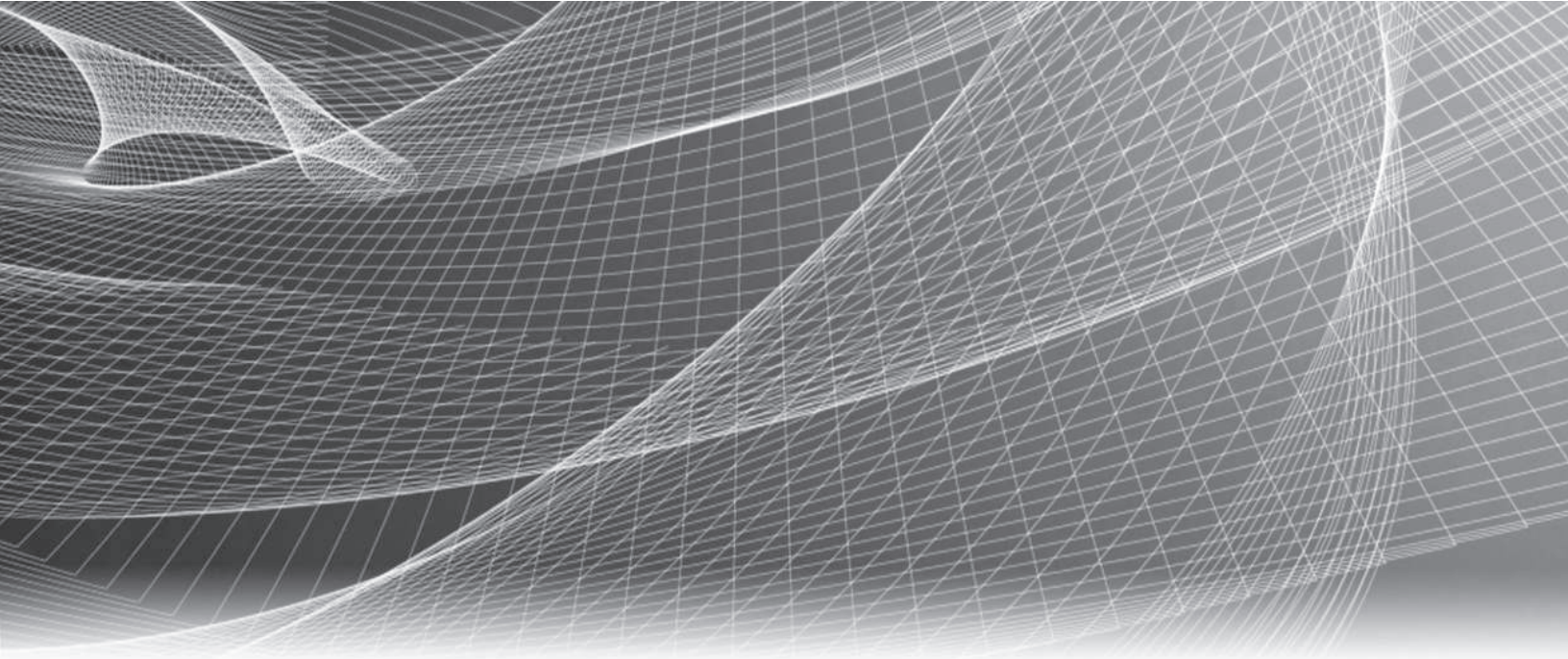




EMC® Unisphere for VMAX REST API

VERSION 1.6

Programmer's Guide

REV 01

Copyright © 2013 - 2013 EMC Corporation. All rights reserved. Published in the USA.

Published May, 2013

EMC 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. EMC Corporation 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 EMC software described in this publication requires an applicable software license.

All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to the technical documentation and advisories section on the EMC online support website.

CONTENTS

Preface	9
Chapter 1: Welcome the Unisphere for VMAX REST API user's guide	13
Chapter 2: REST API architecture	15
Architecture overview	16
Iterators	17
Chapter 3: REST API schemas	19
Downloading schemas	20
Transport protocol	21
Product version and compatibility	22
Backward compatibility	22
Client/Server compatibility	22
Chapter 4: Method examples	23
Sample method endpoint to get array metrics with iterator returned	24
Sample method to get array key with no iterator returned	25
Sample iterator methods and typical use case	26
Chapter 5: Setup tasks	29
Setting up user authentication	30
Chapter 6: Sample REST API client	31
Downloading sample REST API client	32
Client authentication	33
Sample client request and response examples in XML format	34
Sample client request and response examples in JSON format	36

FIGURES

Figure	Page
Figure 1: Client request to get array metrics, IO_RATE and HIT_PER_SEC	34
Figure 2: Client reponse to get array metrics, IO_RATE and HIT_PER_SEC	35
Figure 3: Client request to get array metric IO_RATE	36
Figure 4: Client response to get I/O rate	37

TABLES

Table	Page
Table 1: REST resource methods	13

Preface

As part of an effort to improve its product lines, 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 EMC representative if a product does not function properly or does not function as described in this document

Note

This document was accurate at publication time. New versions of this document might be released on the EMC Online Support site. Check the EMC Online Support site <http://support.EMC.com> to ensure that you are using the latest version of this document.

Purpose

This document describes how to configure and use EMC® Unisphere for VMAX REST API.

Audience

This guide is intended for programmers who want to create interfaces, using these API's, 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

The following related documents are available on the EMC Online Support Site at <https://support.EMC.com>.

- ◆ EMC Unisphere for VMAX Installation Guide
- ◆ EMC Solutions Enabler Release Notes
- ◆ EMC Solutions Enabler Array Management CLI Product Guide
- ◆ EMC Solutions Enabler Array Controls CLI Product Guide
- ◆ EMC Solutions Enabler Symmetrix CLI Command Reference
- ◆ EMC Solutions Enabler Installation Guide
- ◆ EMC Solutions Enabler Symmetrix SRM CLI Product Guide
- ◆ EMC Solutions Enabler Symmetrix TimeFinder Family CLI Product Guide
- ◆ EMC Solutions Enabler Symmetrix SRDF Family CLI Product Guide
- ◆ EMC Solutions Enabler Symmetrix Migration CLI Product Guide
- ◆ EMC Solutions Enabler Security Configuration Guide

Conventions used in this document

EMC uses the following conventions for special notices:



Caution is used with the safety alert symbol, indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.



Notice is used to address practices not related to personal injury.

Note A Note presents information that is important, but not hazard-related.



WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.

Typographical conventions

EMC uses the following type style conventions in this document:

Style	Where used
Normal	Used in running (nonprocedural) text for: - Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus - Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, functions, and utilities - URLs, pathnames, filenames, directory names, computer names, links, groups, service keys, file systems, and notifications
Bold	Used in running (nonprocedural) text for names of commands, daemons, options, programs, processes, services, applications, utilities, kernels, notifications, system calls, and man pages Used in procedures for: - Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus - What the user specifically selects, clicks, presses, or types
<i>Italic</i>	Used in all text (including procedures) for: - Full titles of publications referenced in text - Emphasis, for example, a new term - Variables

Style	Where used
<code>Courier</code>	Used for: • System output, such as an error message or script • URLs, complete paths, filenames, prompts, and syntax when shown outside of running text
Courier bold	Used for specific user input, such as commands
<i>Courier italic</i>	Used in procedures for: Variables on the command line User input variables
< >	Angle brackets enclose parameter or variable values supplied by the user
[]	Square brackets enclose optional values
	Vertical bar indicates alternate selections — the bar means “or”
{ }	Braces enclose content that the user must specify, such as x or y or z
...	Ellipses indicate nonessential information omitted from the example

Where to get help

EMC support, product, and licensing information can be obtained on the EMC Online Support site as described next.

Note To open a service request through the EMC Online Support site, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or to answer any questions about your account.

Product information

For documentation, release notes, software updates, or for information about EMC products, licensing, and service, refer to EMC online support at: <https://support.EMC.com>.

Technical support

EMC offers a variety of support options.

Support by Product — EMC offers consolidated, product-specific information on the Web at: <https://support.EMC.com/products>. The Support by Product web pages offer quick links to Documentation, White Papers, Advisories (such as frequently used Knowledgebase articles), and Downloads, as well as more dynamic content, such as presentations, discussion, relevant Customer Support Forum entries, and a link to EMC Live Chat.

EMC Live Chat — Open a Chat or instant message session with an EMC Support Engineer.

eLicensing support

To activate your entitlements and obtain your Symmetrix license files, visit the Service Center on <https://support.EMC.com>, as directed on your License Authorization Code (LAC) letter emailed to you.

For help with missing or incorrect entitlements after activation (that is, expected functionality remains unavailable because it is not licensed), contact your EMC Account Representative or Authorized Reseller.

For help with any errors applying license files through Solutions Enabler, contact the EMC Customer Support Center.

If you are missing a LAC letter, or require further instructions on activating your licenses through the Online Support site, contact EMC's worldwide Licensing team at licensing@emc.com or call:

- ◆ North America, Latin America, APJK, Australia, New Zealand: SVC4EMC (800-782-4362) and follow the voice prompts.
- ◆ EMEA: +353 (0) 21 4879862 and follow the voice prompts.

CHAPTER 1

Welcome the Unisphere for VMAX REST API user's guide

This document describes the Unisphere for VMAX REST API architecture, schema's, and method examples used to retrieve Symmetrix VMAX diagnostic performance data.

Product overview

The EMC Unisphere for VMAX REST (Representational State Transfer) API allows for accessing diagnostic performance data from your VMAX hardware through robust APIs that communicate with Unisphere for VMAX. These API's 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 model

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

The main REST concepts are the following:

- ◆ 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 link – Link to another representation of the same resource
- ◆ Resource interface – Uniform interface for accessing the resource and manipulating it's state

The following table describes the standard resource methods.

Note The "methodType" - GET, PUT, POST or DELETE, is defined in the Methods.xsd files.

Table 1: REST resource methods

HTTP operation	Description	Method
PUT	Create a new resource	Create resource
GET	Retrieve state of resource	Get resource
POST	Modify resource	Update resource
DELETE	Delete resource	Delete resource

The resource representation is what is used to transfer its information to the client using the HTTP protocol.

The Unisphere for VMAX REST API supports both XML or JavaScript Object Notation (JSON) MIME types.

Supported metrics

The metrics supported with the REST API can be found in the performance.xsd schema. The schema can be downloaded by pointing a browser to:

```
https://{UNIVMAX_IP}:{UNIVMAX_PORT}/univmax/restapi
```

Where, UNIVMAX_IP is the IP address and UNIVMAX_PORT is the port, of the host running Unisphere for VMAX.

CHAPTER 2

REST API architecture

This chapter contains the following:

Architecture overview	16
Iterators	17

Architecture overview

The REST API for Unisphere for VMAX is built upon of two types of XML schemas:

- ◆ Method endpoint schemas — Specifies the method endpoints. The REST API contains the following method schemas:
 - commonMethods.xsd — Specifies the method endpoints associated with an iterator.
 - performanceMethods.xsd — Specifies the method endpoints for accessing storage array performance data.

See ["Sample method endpoint to get array metrics with iterator returned" on page 24](#)

- ◆ Data-type schemas — Defines the data types that are reported between the Unisphere for VMAX server and the client. The REST API contains the following data-type schemas:
 - common.xsd – Defines the iterator data-type. It also defines the array IDs (Symmetrix ID, director ID, disk ID etc.) for the various object types that are supported.
 - performance.xsd – Defines the data-types associated with the performance metrics.

Iterators

An iterator is an object that allows you to cycle through the elements in a collection and display each element. Unisphere for VMAX 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 ["Sample method endpoint to get array metrics with iterator returned" on page 24](#).

The Unisphere for VMAX 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 for VMAX imposes limits on the number of concurrent requests it will accept. See ["Sample iterator methods and typical use case" on page 26](#).

CHAPTER 3

REST API schemas

This chapter contains the following:

Downloading schemas	20
Transport protocol	21
Product version and compatibility	22

Downloading schemas

Clients can download the UNIVMAX REST API schemas, by pointing the browser to:

```
https://{UNIVMAX_IP}:{UNIVMAX_PORT}/univmax/restapi
```

Where, UNIVMAX_IP is the IP address and UNIVMAX_PORT is the port, of the host running Unisphere for VMAX.

Transport protocol

REST API uses the Hypertext Transfer Protocol Secure (HTTPS) version 1.1 as the transport for API requests. For 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.

Unisphere for VMAX 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 for VMAX application.

Backward compatibility

Since the REST API is a method-based API, the methods will 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.

REST API guarantees backwards compatibility for two major releases.

Client/Server compatibility

If the client REST 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.

CHAPTER 4

Method examples

This chapter contains the following:

- Sample method endpoint to get array metrics with iterator returned24
- Sample method to get array key with no iterator returned25
- Sample iterator methods and typical use case26

Sample method endpoint to get array metrics with iterator returned

This definition, demonstrating the method to retrieve array metrics, takes an `ArrayParamType` argument and returns an unbounded number of `ArrayResultType` results. Whenever a method endpoint is defined that returns an unbounded number of results, the actual result that is returned is an iterator of multiple instances of the return type along with the first page of results.

In this example, the result returned is an iterator of `ArrayResultType` instances. The iterator definition is contained in the `common.xsd` schema. The schema `performance.xsd` contains definitions for `ArrayParamType` and `ArrayResultType`.

A sample method endpoint to get array metrics using the REST API is defined as:

```
<!-- Get Array metrics -->
<xs:complexType name="ArrayMetrics">
  <xs:sequence>
    <xs:element name="param" type="perf:ArrayParamType"
      minOccurs="1" maxOccurs="1"/>
    <xs:element name="result" type="perf:ArrayResultType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="methodType" type="cmnm:POST"/>
  <xs:attribute name="url">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration
          value="restapi/performance/Array/metrics"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:element name="arrayMetrics" type="ArrayMetrics"/>
```

Sample method to get array key with no iterator returned

This definition demonstrates the method to retrieve array keys. This is a case where the size of the results is not unbounded and the method does not return an iterator. The request to get the array keys returns an instance of `ArrayKeyResultType`. The schema `performance.xsd` contains the definition for `ArrayKeyResultType`.

A sample method to get array keys is defined as:

```
<!-- Get Array keys -->
<xs:complexType name="ArrayKeys">
  <xs:sequence>
    <xs:element name="result" type="perf:ArrayKeyResultType"
      minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="methodType" type="cmnm:GET"/>
  <xs:attribute name="url">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="restapi/performance/Array/keys"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:element name="arrayKeys" type="ArrayKeys"/>
```

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 for VMAX server, the iterator expires, based on a pre-configured timeout. 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 method that requests array metrics and returns unbounded number of results.
2. The IteratorInfo method is called on an existing iterator, as shown in Get Iterator info example, and the client receives an iterator, with the first result page.

```
<!-- Get Iterator info -->
<xs:complexType name="IteratorInfo">
  <xs:sequence>
    <xs:element name="result" type="cmn:IteratorType"
      minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="methodType" type="GET"/>
  <xs:attribute name="url">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="restapi/common/Iterator/
          {iteratorId}"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:element name="iteratorInfo" type="IteratorInfo"/>
```

3. The client pages over the results until all results are retrieved, as shown in Get Iterator page example.

```
<!-- Get Iterator page -->
<xs:complexType name="IteratorPage">
  <xs:sequence>
    <xs:element name="result" type="cmn:ResultListType"
      minOccurs="1" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="methodType" type="GET"/>
  <xs:attribute name="url">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="restapi/common/Iterator/
          {iteratorId}/page?from={x}&to={y}"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:element name="iteratorPage" type="IteratorPage"/>
```

4. If the iterator has not expired, the client deletes the iterator as shown in Delete Iterator example. this releases server resources.

```
<!-- Delete Iterator -->
<xs:complexType name="IteratorDelete">
  <xs:attribute name="methodType" type="DELETE"/>
  <xs:attribute name="url">
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="restapi/common/Iterator/
          {iteratorId}"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<xs:element name="iteratorDelete" type="IteratorDelete"/>
```


CHAPTER 5

Setup tasks

This chapter contains the following:

Setting up user authentication	30
---	-----------

Setting up user authentication

Prior to using the Unisphere for VMAX REST API, user authorization must be assigned for each Symmetrix VMAX array the user is permitted to access.

User assignment for the arrays is performed as follows:

1. Log into Unisphere for VMAX.
2. At the bottom of the Common Tasks panel click Administration.
3. Click Users and Roles to open the Users and Roles list view.
4. Create a user login profile for each Symmetrix array to be accessed by the user, and assign them to any of the following roles:
 - ◆ Users that are allowed access the Symmetrix performance metrics:
 - Administrator
 - Storage Administrator
 - Performance Monitor
 - Monitor
 - ◆ Users that are not allowed access the Symmetrix performance metrics:
 - Auditor
 - Security Administrator
 - None

Note These user roles are valid for the currently available REST resource methods (GET, POST, and DELETE) for the Unisphere interface. When PUT methods become available only Admin and Storage Admin roles will be able to initiate PUT methods.

CHAPTER 6

Sample REST API client

This chapter contains the following:

Downloading sample REST API client	32
Client authentication	33
Sample client request and response examples in XML format	34
Sample client request and response examples in JSON format	36

Downloading sample REST API client

The RESTApi.zip file contains all the jar files needed to build the sample client. The RESTApi.zip can be downloaded from <http://support.EMC.com>.

To build the client:

```
javac -classpath lib\httpclient-4.2.1.jar;lib\httpcore-4.2.1.jar  
RESTApiTest.java
```

To run the client:

```
java -classpath .;lib\httpclient-4.2.1.jar;lib\httpcore-  
4.2.1.jar;lib\commons-logging-1.1.1.jar;lib\commons-codec-1.6.jar  
RESTApiTest <fully_qualified_hostname_or_ip_address> <username>  
<password> <startDate> <endDate> <symmID>
```

Example

```
java -classpath .;lib\httpclient-4.2.1.jar;lib\httpcore-  
4.2.1.jar;lib\commons-logging-1.1.1.jar;lib\commons-codec-1.6.jar  
RESTApiTest 127.0.0.1 smc smc 1360894200000 1360894200000  
000195600233
```

Note The *startDate* and *endDate* are time expressed as the number of milliseconds using the "epoch" as the reference point from which time is measured (January 1, 1970, 00:00:00 GMT). This allows the date and time of events to be specified as an integer value rather than a calendar representation. Epoch converter tools are available on the internet.

Client authentication

REST API uses HTTP Basic Access Authentication to authenticate API clients, described in RFC 2617. HTTP Basic Access Authentication is simple, since 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 Symmetrix VMAX arrays within Unisphere for VMAX. 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.

Sample client request and response examples in XML format

The following examples in XML format are shown with the input (request) into the sample client and the output (response) from the sample client. The [Array metrics example code](#) is also available for copying and pasting into the client.

Sample client request for array metrics

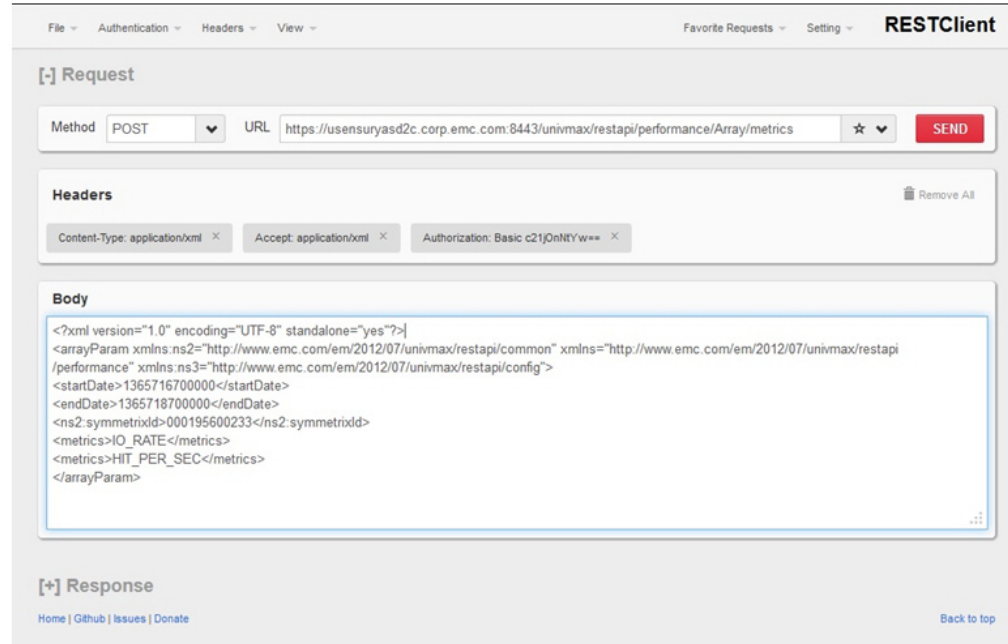


Figure 1: Client request to get array metrics, IO_RATE and HIT_PER_SEC

Array metrics example code

Note Change the values for **symmetrixId**, **startDate**, and **endDate** as appropriate for the environments. Also, the **Authorization** header value represents Basic Authentication information in Base64 encoded format. Refer to RFC 2617 for details of Basic Access Authentication.

Method — POST

URL — `https://IP:PORT/univmax/restapi/performance/Array/metrics`

Headers

- ◆ Content Type: `application/xml`
- ◆ Accept: `application/xml`
- ◆ Authorization: `Basic c21jOnNtYw==`

Body

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<arrayParam
xmlns:ns2="http://www.emc.com/em/2012/07/univmax/restapi/common"
xmlns="http://www.emc.com/em/2012/07/univmax/restapi/performance"
xmlns:ns3="http://www.emc.com/em/2012/07/univmax/restapi/config">

<startDate>1365716700000</startDate>

<endDate>1365716700000</endDate>

<ns2:symmetrixId>000195600233</ns2:symmetrixId>

<metrics>IO_RATE</metrics>

<metrics>HIT_PER_SEC</metrics>

</arrayParam>
```

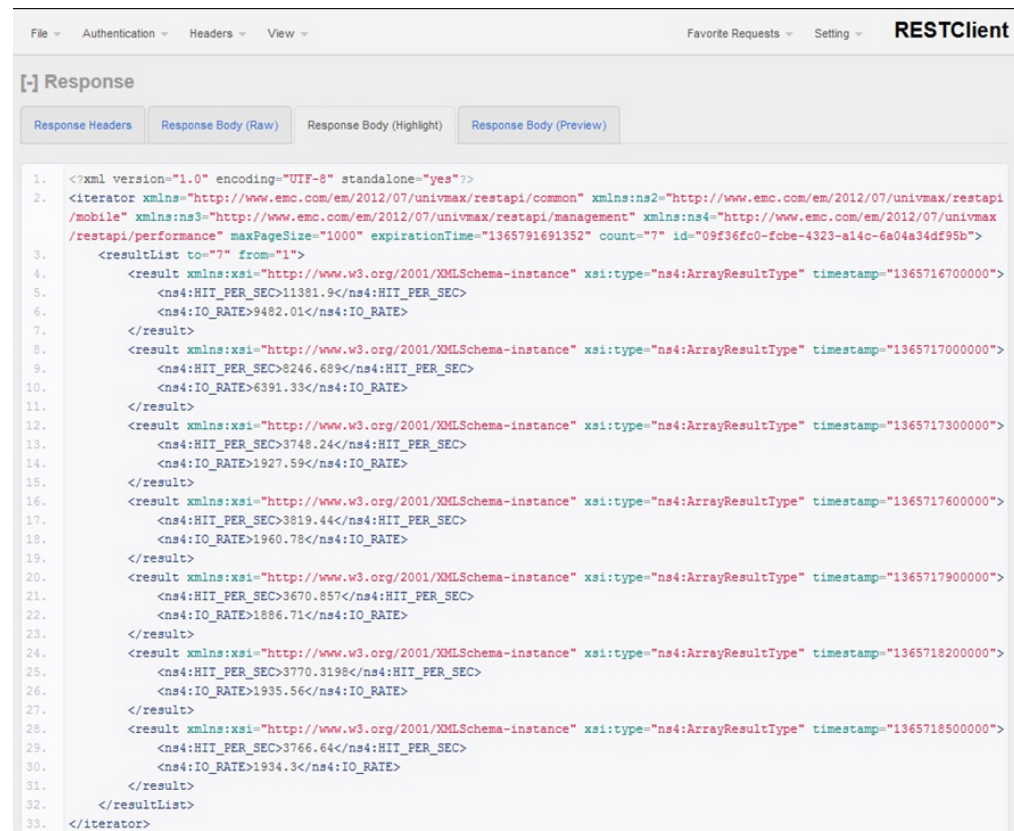


Figure 2: Client response to get array metrics, IO_RATE and HIT_PER_SEC

Sample client request and response examples in JSON format

The following examples in JSON format are shown with the input (request) into the sample client and the output (response) from the sample client. The [FE director metrics example code](#) is also available for copying and pasting into the client.

Sample client request for FE director metrics

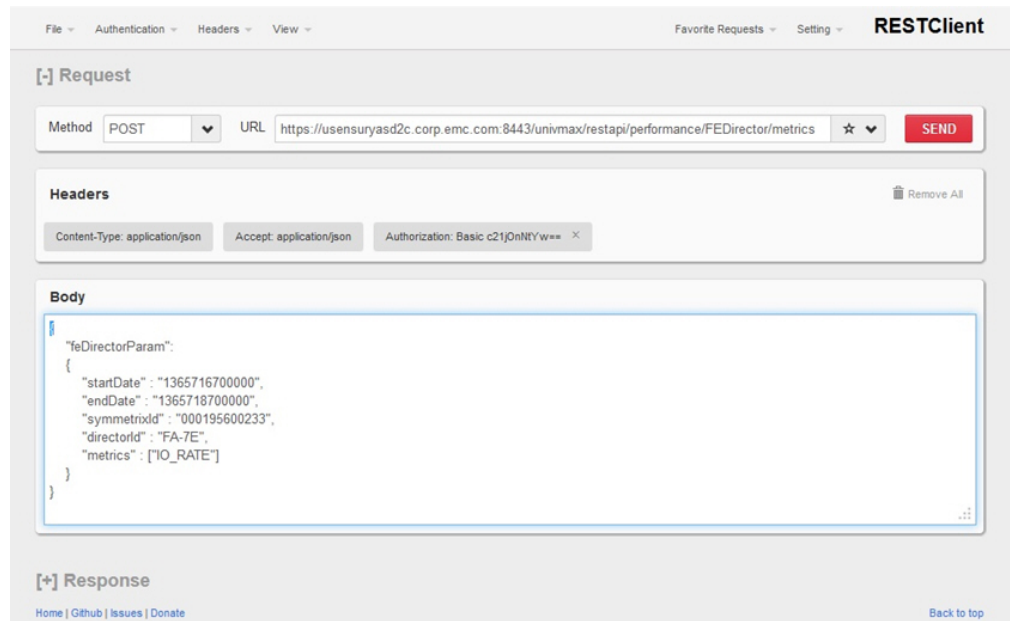


Figure 3: Client request to get array metric IO_RATE

FE director metrics example code

Note Change the values for **symmetrixId**, **startDate**, **endDate** and **directorId** as appropriate for the environments. Also, the **Authorization** header value represents Basic Authentication information in Base64 encoded format. Refer to RFC 2617 for details of Basic Access Authentication.

Method — POST

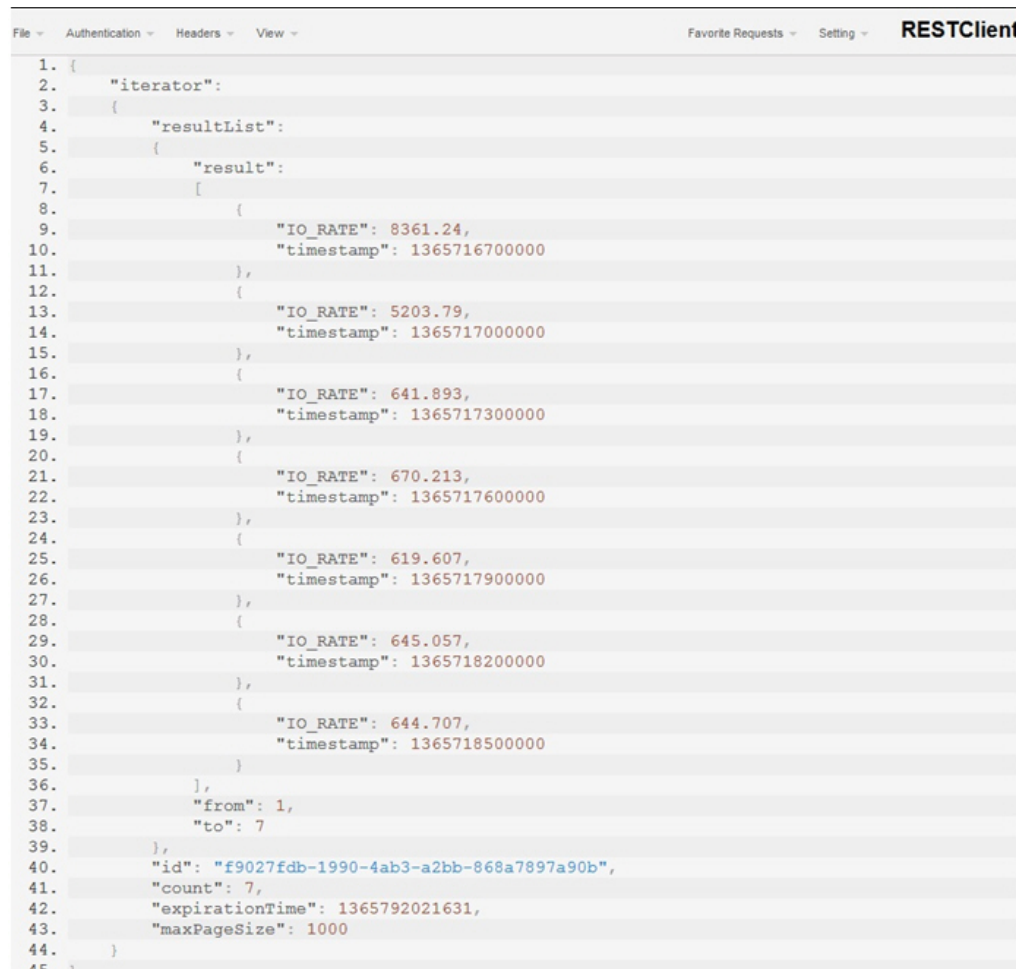
URL — `https://IP:PORT/univmax/restapi/performance/FEDirector/metrics`

Headers

- ◆ Content-Type: application/json
- ◆ Accept: application/json
- ◆ Authorization: Basic c2ljOnNtYw==

Body

```
{
  "feDirectorParam":
  {
    "startDate" : "1357247700000",
    "endDate" : "1357280400000",
    "symmetrixId" : "000195600233",
    "directorId" : "FA-7E",
    "metrics" : ["IO_RATE"]
  }
}
```

Sample client response for FE director metrics


```
1. {
2.   "iterator":
3.   {
4.     "resultList":
5.     {
6.       "result":
7.       [
8.         {
9.           "IO_RATE": 8361.24,
10.          "timestamp": 1365716700000
11.        },
12.        {
13.          "IO_RATE": 5203.79,
14.          "timestamp": 1365717000000
15.        },
16.        {
17.          "IO_RATE": 641.893,
18.          "timestamp": 1365717300000
19.        },
20.        {
21.          "IO_RATE": 670.213,
22.          "timestamp": 1365717600000
23.        },
24.        {
25.          "IO_RATE": 619.607,
26.          "timestamp": 1365717900000
27.        },
28.        {
29.          "IO_RATE": 645.057,
30.          "timestamp": 1365718200000
31.        },
32.        {
33.          "IO_RATE": 644.707,
34.          "timestamp": 1365718500000
35.        }
36.      ],
37.      "from": 1,
38.      "to": 7
39.    },
40.    "id": "f9027fdb-1990-4ab3-a2bb-868a7897a90b",
41.    "count": 7,
42.    "expirationTime": 1365792021631,
43.    "maxPageSize": 1000
44.  }
45. }
```

Figure 4: Client response to get I/O rate

