Application Program Interface

CorteX

RESTful Web Service
and
Resource-Oriented Architecture

Xiotech Corporation
9950 Federal Drive, Suite 100
Colorado Springs, CO 80921-3686

www.X-IO.com

Main: 719.388.5500
Fax: 719.388.5300
Customer Support: 1.866.472.6764

Publication Number: 160311-000 Rev E

November 2014

Table of Content

Introduction

This document describes the design of the X-IO CorteX Program using RESTful Web Services and Resource-Oriented Architecture.

Audience

This document is primarily intended for software developers using the CorteX interface.

Conventions

The terminology for this API follows industry standards where applicable, especially based on HTTP and storage-based terminology.

The following conventions are used in this manual.

Element	Convention
Button	**Arial, 9 pt, Bold**
Command Line Interface input/output	**`Courier New, 10 pt, Bold`**
Document title	*Italic*
Emphasize a word or phrase	important
Event notifications or other message	**Times New Roman, 10 pt, Bold**
Key	`COURIER NEW, 11 PT, CAPS`
Literal	***Times New Roman, 10 pt, Bold, Italic***
Web-Mgt menu item or option	**Arial, 10 pt, Bold**
Variable, represents text that must be entered	***`<Courier New, 10 pt, Bold, Italic>`***

Table 1: Document Conventions

Related Documents

The following references are essential to an understanding of this document:

- *RESTful Web Services* by Richardson & Ruby (ISBN: 978-0-596-52926-0)

Requirements

Strict adherence to RESTful methodologies (Resource-Oriented Architecture implementation) is required. A RESTful and Resource-Oriented design meets the following qualifications:

1. HTTP method matches the method information; only available HTTP Methods are used.
2. Scoping information must be in the Universal Resource Identifier (URI); the URI format must correctly exhibit both the name and address of the resource that the HTTP method is to execute on.

Overview

The CorteX interface provides the following:

1. Exhibits Resource-Oriented Architecture based on RESTful Web Services
    - Addressable
    - Stateless
    - Connected
    - Uniform Interface
2. Simplifies the overall storage management access to arrays
3. Increases management scalability to the level of the Internet
4. Provides the needed hook for Enterprise-level storage management
5. Provides a platform for Collective Management and Collective Intelligence

RESTful Web Services

The acronym **REST** stands for REpresentational State Transfer. REST is not an architecture design in and of itself, but it is a way to determine whether an architecture design is RESTful. Since the goal of designing a RESTful architecture is to make things simple in a well-defined way, care must be taken in the URI development and how the HTTP methods interact with the resource.

A RESTful Web Services design uses only available HTTP Methods and adheres to the following rules:

1. Use HTTP Method Requests to do actions, methods, or any other verb.
    - GET—Returns a payload of information about the requested resource.
    - POST—Creates a new resource on the target server system.
    - HEAD—Returns the header information about a request resource (no payload of info).
    - PUT—Modifies an existing resource.
    - DELETE—Remove an existing resource from the server system.
    - OPTIONS—Returns the **Allow: <*methods*>** header from the resource indicating which HTTP methods are allowed to operate on the requested resource.

    NOTE: Since the body of the response from an OPTIONS request is undefined, use the body to return the resource XML Schema Document as the payload.
2. Use HTTP Response Headers and values when returning status and/or data.
    - 1xx: META codes (rare and not typically needed).
    - 2xx: SUCCESS codes. Tells the client that the request executed successfully.
    - 3xx: REDIRECTION codes. Tells the client that the resource has moved to a new location.
    - 4xx: CLIENT-SIDE ERROR codes. Tells the client that something is wrong with the request from the client.
    - 5xx: SERVER-SIDE ERROR codes (typically Web server handles these automatically). Tells the client that the server cannot service the client request.

3. All URI strings must be noun-based. This means the URI must represent the address of something and must not have any action or method verb in the string.
4. Any resource must either be represented by its fully qualified URI or its ID as a query string input parameter to an HTTP method call. This means the RESTful client must not have to parse any piece of a particular resource's URI in order to enter it as an input parameter; instead, it must reuse the discovered URI of any resource that is to be used as a parameter in a method.
   - Examples:

   ```
   POST http://{resource}?param1={id1}&param2={id2}
   POST http://{resource}?param1={uri1}&param2={uri2}
   ```

Resource-Oriented Architecture

A resource is anything of interest that a client wishes to obtain from a Web service that may provide it. In order for resources to work on the Web, they must have a name and address. This is accomplished using the Universal Resource Identifier (URI). The URI contains both the name of the resource as well as the location of the resource (URN + URL). This allows the resource to become visible to the Web such that a client can access the representation of the resource.

As mentioned in the overview, the second rule of thumb in developing a RESTful Web Service is to make sure the Universal Resource Identifiers (URIs) uniquely describe or scope the resources. This aspect adheres to the first of the four tenants to ROA-based system design. In order to be considered an ROA-based Web Service, the service must be:

- Addressable
- Stateless
- Connected
- Uniform Interface

Addressable

An application or device is considered addressable if it exposes its data as a set of resources. In the case of X-IO, this means that each and every physical and logical resource has an address using URIs (if to be exposed). To scope this properly, the Array exposes its resources in a simple, intuitive addressing scheme based on both industry-standard (and the upcoming industry-standard-like Cloud Data Management Interface, or CDMI) and vendor unique addressing.

Stateless

To be stateless means that every HTTP request happens in isolation. This means that the server has everything it needs from the client in order to service the request and does not need to rely on any information from a previous client request. An example of how to violate this principle is to have a Web site that expects the user to move through pages in a certain order, like A, then B, then C, but the user chooses to redo B or backward in the history. The Web site then gets confused because the user selected resources out of an expected order. The goal for the CorteX design is to make any and all HTTP accesses are isolated and that all information for a given action or operation is atomic in order to adhere to the stateless tenant.

Connected

As a client surfs through a given Web service, the service should provide hypermedia or links to other resources that are near the current resource. In other words, a representation of a resource should provide links to other resources within the Web service. The server guides the client through the possible other resources that the Web service may serve up. Since the server-side is stateless, it is up to the client to maintain **application state** as it moves through the Web service resources. For this reason do not keep session contexts between clients and the Web service. Let the client handle what it should do next with the help of the Web service providing navigation of its site. The array has a first page resource URI containing all links to all other resources on the array. This first page can be thought of as the home page or site index so that the client can immediately determine where it wants to go next.

Uniform Interface

A uniform interface means using a common or industry-standard set of interfaces in working with a resource. In HTTP, there are four main methods that act on a resource:

- GET—Retrieves the representation of the resource
- POST—Creates a new instance of a resource
- PUT—Modifies an attribute of an existing resource
- DELETE—Deletes an existing resource

Any needed Array action should be available through these four methods. This makes the Array easy to manage and with using industry-standard methods and objects/attributes for storage devices a client developer can easily generate an application that can integrate with a Array.

Note. Overloading the HTTP POST method with verb parameters is a typical way of solving action commands on Web sites that do not support HTTP methods beyond GET and POST. This is not the case with the array Web server. The array Web server supports the six main HTTP methods, so the architecture should be designed to be fully RESTful. This surpasses the RPC-based design philosophy of SOAP-based Web services.

CorteX Universal Resource Identifier

This section discusses the URI, the template, and how it fits into the CorteX storage management model.

Universal Resource Identifier

Universal Resource Identifier (URI) is an industry-standard identifier for Web-based management. It refers to a unique address in the internet-space for a particular Web resource. A resource can be anything: a file, a Web page, a link, etc. In the case of an Array, a URI refers to a particular resource on the Array. The Array has the following available resources:

- Logical Storage Resources
  - Storage Volumes
  - Host Objects
  - Endpoints (HBA Ports)
  - Allocations (LUN Presentations)
  - Storage Pools
- Physical System Resources
  - Array (Array system-level resource)
  - Controllers (MRCs)
  - Media
  - Power Supplies
  - Batteries
- Logical System Resources
  - Network
  - Chronometer

  - Jobs (Operations)
  - Files
  - Subscriptions (for a client to receive RESTful Alerts)
  - Revision (Controller and Medium firmware images)
  - Performance
- Discovery
  - Query (non-authorized way to do Web crawl discovery of Arrays)

The URI must be noun-based according to the rules of REST Architecture. Any verbs (methods or actions) need to use one of the HTTP Methods called out in the HTTP specification. So the key to developing a proper URI is to think about the URI as being the address of a thing or noun. Dismiss any desires toward overloading the URI with verbs (like GET or POST http://noun?method=<action>). The Array Web server supports all 6 supported HTTP methods, so overloading POST is not necessary. Be creative in thinking about how to solve non-intuitive actions using the HTTP Methods. This helps in the design transition from typical RPC-like thinking into RESTful thinking.

CorteX URI Template

The CorteX uses the following two URI patterns:

- SYSTEM-LEVEL RESOURCE URI
  http://<hostname or ip_addr>/storage/arrays/{id}/{resource}/{id}
  where the *arrays/{id}* is the Array ID, and the /{resource}/{id} is the unique identifier for this array's resource.
- STORAGE WEB-LEVEL RESOURCE URI
  http://<hostname or ip_addr>/storage/{resource}/{id}
  where the globally unique and available resource is directly accessible over the network independent of the "/arrays/{id}" path. This is used by higher tier management layers for doing collective management and collective intelligence regarding the Array's logical resources such as Storage Volumes, etc. See section 5.0 below for more detail.

Array URI Listing

Below is the URI listing for all resources on the Array (all URIs have the prefix *http://<ip_addr>/storage*).

SPECIFIC LEVEL URI FORMAT

- Array: /arrays/{id}
- Volumes: /arrays/{id}/volumes/{id}
- Host Objects: /arrays/{id}/hosts/{id}
- Endpoints: /arrays/{id}/endpoints/{id}
- Allocations: /arrays/{id}/allocations/{volumeId},{host_id}
- Storage Pools: /arrays/{id}/pools/{id}
- Controllers: /arrays/{id}/controllers/{id}
- Media: /arrays/{id}/media/{id}
- Power Supplies: /arrays/{id}/powersupplies/{id}
- Batteries: /arrays/{id}/batteries/{id}
- Network: /arrays/{id}/network
- Chronometer: /arrays/{id}/chronometer
- Jobs: /arrays/{id}/jobs/{id}
- Files: /arrays/{id}/files/{filename}
- Subscriptions: /arrays/{id}/subscriptions/{id}
- Revision: /arrays/{id}/revision

- Performance: /arrays/{id}/performance
- Query: http://<ip_addr>/query

STORAGE WEB LEVEL URI FORMAT

- Array: /arrays
- Volumes: /volumes/{id}
- Host Objects: /hosts/{id}
- Endpoints: /endpoints/{id}
- Allocations: /allocations/{volumeId},{host_id}
- Storage Pools: /pools/{id}

RESTful Resources

Because of the RESTful Web Services capability of the Array, all resources on the Array are directly addressable based on the Array URI template structure. For brevity in the following sections, the URI prefix has been removed from the Resource HTTP Method tables. The URI prefix is:

http://<ip_addr>/storage

(for all Array system-level resources except for the Array itself, which is just this URI)

Discovery

Query

The Query resource does not require authentication since it is meant for clients to discover Arrays based on using an IP Address and the /query resource entry for the Query URI. The /query returns some basic information about the Array in the response. The information includes the Array main URI (site index URI), Array Name, Array ID, Serial Number, Model, Chronometer information, and basic MRC information included FW Version level. Query is useful in system discovery processes that use search or Web-crawl technologies since it only requires an IP Address and /query:

GET http://*<ip_addr>*/query—Responds with some basic info about the Array

HEAD http://*<ip_addr>*/query—Responds with just the header; useful as a ping to discover Arrays without needing the basic info from the GET

The Query resource for the Array supports the following HTTP Methods:

• GET	Returns inquiry information of the Array
• HEAD	Returns the Location Header information about the Query but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for query

HTTP Method	Options	Description
URI	/query	
Method	GET	Get the Query information
Query String		
Response	200 404	OK and Query Info in response entity-body Not Found NOTE: If Not Found, then the device at the IP Address is most likely not a Array

Table 2: Get Query Information

HTTP Method	Options	Description
URI	/query	
Method	HEAD	Get the header information about Query
Query String		
Response	200 404	OK and the Location Header (URI of the Query) Not Found

Table 3: Get Header Info for Query

HTTP Method	Options	Description
URI	/query	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Not Found

Table 4: Get Options Info for Query

Logical Storage Resources

The logical storage resources are storage, hosts, HBA ports, LUN presentations, and storage pools.

Storage Volumes

Storage Volumes on the Array supports the following HTTP Methods.

• GET	Returns either the list of volumes or a particular volume if specified
• POST	Creates a new volume
• PUT	Modifies an attribute of an existing volume
• HEAD	Returns the Location Header information about the volume but not the attributes
• DELETE	Deletes a volume from the Storage Pool
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for volume

HTTP Method	Options	Description
URI	/volumes	
Method	GET	Returns the list of Storage Volume URIs and its information

HTTP Method	Options	Description

Query String	name={*name*}
	redundancy={**1** | **5**}
	writecache={Write-Back | Write-Through}
	pool={**1** | **2** | *poolGUID*}
	type={*type*}
	
	affinity=
	
	alloctype={*type*}
	IOPS min={*value*}
	
	IOPS max={*value*}
	IOPS burst={*value*}
		Returns all the information for the specified name
		Returns all the volumes with that redundancy setting
		Returns all the volumes with that write cache setting
		
		Returns all the volumes on that pool
		Returns only volumes of type: **`primary`**, **`snapshot`**, or **`mirror`** [not iSCSI]
		cadp (continuous adaptive data placement, default)
		hdd (volume pinned to HDD)
		flash (volume pinned to flash, if hybrid pool)
		Fully allocated or thin-provisioned.
		Quality of Service (QoS) IOPS minimum goal. A value of 0 means not set, default.
		QoS IOPs maximum limit. A value of 0 means not set, default.
		QoS IOPs burst limit. A value of 0 means not set, default.

Response	200 401	OK and URI List of volumes Unauthorized

Table 5: Get All Storage Volumes

HTTP Method	Options	Description
URI	/volumes/{*id*}	
Method	GET	Get the information about a specific volume

Query String	name={*name*}
	redundancy={**1** | **5**}
	writecache={Write-Back | Write-Through}
	pool={**1** | **2** | *poolGUID*}
	type={*type*}
	
	affinity=
	
	alloctype={*type*}
	IOPS min={*value*}
	
	IOPS max={*value*}
	IOPS burst={*value*}
		Returns all the information for the specified name
		Returns all the volumes with that redundancy setting
		Returns all the volumes with that write cache setting
		
		Returns all the volumes on that pool
		Returns only volumes of type: **`primary`**, **`snapshot`**, or **`mirror`** (not iSCSI)
		cadp (continuous adaptive data placement, default)
		hdd (volume pinned to HDD)
		flash (volume pinned to flash, if hybrid pool)
		Fully allocated or thin-provisioned.
		Quality of Service (QoS) IOPS minimum goal. A value of 0 means not set, default.
		QoS IOPs maximum limit. A value of 0 means not set, default.
		QoS IOPs burst limit. A value of 0 means not set, default.

Response	200 401 404	OK and Volume Info in response entity-body Unauthorized Not Found

Table 6: Get a specific Storage Volume information

HTTP Method	Options	Description
URI	/volumes	
Method	POST	Create a new volume

HTTP Method	Options	Description
Query String	size= name= writecache = pool= redundancy= comment= affinity= alloctype={*type*} IOPS min={*value*} IOPS max={*value*} IOPS burst={*value*}	Size of the volume in GB. [required] Name of the volume. Max 32 characters. [optional] Write Cache Policy. Write-Back is default. [optional] [Policy Values = {Write-Back, Write-Through}] Hosting Storage Pool. Global ID of the Pool. [optional] RAID Level for the volume. RAID 1 is default. [optional] [Redundancy Values = {1,5}] Comment field. Max 60 characters. [optional] cadp (continuous adaptive data placement, default) hdd (volume pinned to HDD) flash (volume pinned to flash, if hybrid pool) Fully allocated or thin-provisioned. Quality of Service (QoS) IOPS minimum goal. A value of 0 means not set, default. QoS IOPs maximum limit. A value of 0 means not set, default. QoS IOPs burst limit. A value of 0 means not set, default.
Response	201 400 401 409	Created and Location Header (URI of new volume) Bad Request (typically a faulty parameter) Unauthorized Conflict (typically volume name already in use)

Table 7: Create a Storage Volume

Note. Affinity thin provisioning and QoS are not supported in all firmware versions.

HTTP Method	Options	Description
URI	/volumes/{*id*}	s
Method	PUT	Modify an existing volume
Query String	size= name= writecache= comment= IOPS min={*value*} IOPS max={*value*} IOPS burst={*value*}	Size of the volume in GB. Name of the volume. Max 32 characters. Write Cache Policy. Write-Back is default. [Policy Values = {Write-Back, Write-Through}] Comment field. Max 60 characters. Quality of Service (QoS) IOPS minimum goal. A value of 0 means not set, default. QoS IOPs maximum limit. A value of 0 means not set, default. QoS IOPs burst limit. A value of 0 means not set, default.
Response	201 400 401 404 409	Created and Location Header (URI of volume) Bad Request (typically a faulty parameter) Unauthorized Not Found Conflict (typically volume name already in use)

Table 8: Modify Storage Volume

HTTP Method	Options	Description
URI	/volumes/{*id*}	
Method	DELETE	Delete an existing volume

HTTP Method	Options	Description
Query String		
Response	204 401 404 409	No Content (indicates the delete worked) Unauthorized Not Found Conflict (typically volume is still presented to a host)

Table 9: Delete Storage Volume

HTTP Method	Options	Description
URI	/volumes/{*id*}	
Method	HEAD	Get the header information about a volume
Query String		
Response	200 401 404	OK and the Location Header (URI of the volume) Unauthorized Not Found

Table 10: Get Header Info for Storage Volume

HTTP Method	Options	Description
URI	/volumes/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 11: Get Options Info for Storage Volume

Hosts

Hosts on the Array are really groupings of Endpoints (HBA Port WWNs). Hosts on the Array support the following HTTP Methods:

• GET	Returns either the list of hosts or a particular host if specified
• POST	Creates a new host
• PUT	Modifies an attribute of an existing host
• HEAD	Returns the Location Header information about the host but not the attributes
• DELETE	Deletes a host and returns the End-points to the unassigned list of End-points
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for host

HTTP Method	Options	Description
URI	/hosts	
Method	GET	Returns the list of Host Object URIs

Query String	name={*name*}
	os={*type*}
		Returns all information for specified name
		Returns all hosts of specified OS type

Response	200 401	OK and URI List of host objects Unauthorized

Table 12: Get All Host Objects

HTTP Method	Options	Description
URI	/hosts/{*id*}	
Method	GET	Get the information about a specific host object

Query String	name={*name*}
	os={*type*}
		Returns all information for specified name
		Returns all hosts of specified OS type

Response	200 401 404	OK and Host Object Info in response entity-body Unauthorized Not Found

Table 13: Get Specific Host Object information

HTTP Method	Options	Description
URI	/hosts	
Method	POST	Create a new host
Query String	wwn = OR endpoint= name= os = comment=	URI of the Endpoint to group with this host. [required] NOTE: To add more than 1 wwn to the list, simply add more **`wwn =<`*`endpoint_wwn`*`>`** to the query string. Name of the volume. Max 32 characters. [optional] Host OS for this Host Object. Windows is default. [optional] Comment field. Max 60 characters. [optional]
Response	201 400 401 409	Created and Location Header (URI of new host) Bad Request (typically a faulty parameter) Unauthorized Conflict (typically host object name already in use)

Table 14: Create Host Object

HTTP Method	Options	Description
URI	/hosts/{*id*}	
Method	PUT	Modify an existing host object
Query String	wwn = OR endpoint= name= os = comment=	Add a new Endpoint to the Host Object. Name of the volume. Max 32 characters. Change the Host OS setting. Comment field. Max 60 characters
Response	201 400 401 404 409	Created and Location Header (URI of host) Bad Request (typically a faulty parameter) Unauthorized Not Found Conflict (typically host name already in use)

Table 15: Modify Host Object

HTTP Method	Options	Description
URI	/hosts/{*id*}	
Method	DELETE	Delete an existing host object
Query String	wwn = OR endpoint=	When a WWN is included, the WWN is removed from the host object endpoint list. [optional]
Response	204 401 404 409	No Content (indicates the delete worked) Unauthorized Not Found Conflict (typically host is still presented to a volume)

Table 16: Delete Host Object

HTTP Method	Options	Description
URI	/hosts/{*id*}	
Method	HEAD	Get the header information about a host
Query String		
Response	200 401 404	OK and the Location Header (URI of the host) Unauthorized Not Found

Table 17: Get Header Info for Host Object

HTTP Method	Options	Description
URI	/hosts/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 18: Get Options Info for Host Object

End-points (HBA Ports)

End-points on the Array are the HBA Ports from the remote host systems that the Array detects on the fabric. End-points support the following HTTP Methods:

• GET	Returns either the list of host or a particular host if specified
• HEAD	Returns the Location Header information about the host but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for host

HTTP Method	Options	Description
URI	/endpoints	
Method	GET	Returns the list of endpoint URIs
Query String	hostname={*name*}	Returns all endpoints attached to specified host name
Response	200 401	OK and URI List of endpoints Unauthorized

Table 19: Get All Endpoints

HTTP Method	Options	Description
URI	/endpoints/{*id*}	
Method	GET	Get the information about a specific host object
Query String	hostname={*name*}	Returns all endpoints attached to specified host name
Response	200 401 404	OK and Host Object Info in response entity-body Unauthorized Not Found

Table 20: Get Specific Endpoint Information

HTTP Method	Options	Description
URI	/endpoints/{*id*}	
Method	HEAD	Get the header information about an endpoint
Query String		
Response	200 401 404	OK and the Location Header (URI of the endpoint) Unauthorized Not Found

Table 21: Get Header Info for Endpoint

HTTP Method	Options	Description
URI	/endpoints/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body
	401	Unauthorized
	404	Not Found

Table 22: Get Options Info for Endpoint

Allocations (LUN Presentations)

Allocations on the Array represent the Host-to-Volume mapping or path. Allocations support the following HTTP Methods:

• GET	Returns either the list of allocations or a particular allocation if specified
• POST	Creates a new allocation
• PUT	Modifies an attribute of an existing allocation. In this case, changing the LUN number
• HEAD	Returns the Location Header information about the allocation but not the attributes
• DELETE	Deletes an allocation; removes the Host-to-Volume mapping
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for allocation

HTTP Method	Options	Description
URI	/allocations	
Method	GET	Returns the list of Allocation URIs

Query String	volumename={*name*}
	hostname={*name*}
		Returns all allocations connected to that volume name
		Returns all allocations connected to that host name

Response	200 401	OK and URI List of allocations Unauthorized

Table 23: Get All Allocations

HTTP Method	Options	Description
URI	/allocations/{*id*}	
Method	GET	Get the information about a specific allocation
Query String	volumename={*name*}	Returns all information for the specified volume
Response	200 401 404	OK and Allocation Info in response entity-body Unauthorized Not Found

Table 24: Get Specific Allocation Information

HTTP Method	Options	Description
URI	/allocations	
Method	POST	Create a new allocation
Query String	volume= arrayendpoint = hostendpoint = lun = volumename={*name*} hostname={*name*}	Storage Volume Global ID or URI [required or use volumename] Array endpoint Global ID or URI [optional] Host Endpoint Global ID or URI [required or use hostname] LUN number for the allocation. Range 0-239 [optional] **NOTE:** If **lun** is not specified, the LUN is auto-assigned to the next available LUN number between the volume and host object. Volume Name instead of volume GUID or URI Host Name instead of hostendpoint GUID or URI
Response	201 400 401 409	Created and Location Header (URI of new allocation) Bad Request (typically a faulty parameter) Unauthorized Conflict (typically LUN already in use)

Table 25: Create Allocation

HTTP Method	Options	Description
URI	/allocations/{*id*}	
Method	PUT	Modify an existing allocation
Query String	lun =	Change the LUN number. Range 0-239.
Response	201 400 401 404 409	Created and Location Header (URI of volume) Bad Request (typically a faulty parameter) Unauthorized Not Found Conflict (typically LUN value already in use)

Table 26: Modify Specific Allocation

HTTP Method	Options	Description
URI	/allocations/{*id*}	
Method	DELETE	Delete an existing allocation
Query String		
Response	204 401 404	No Content (means the delete worked) Unauthorized Not Found

Table 27: Delete Specific Allocation

HTTP Method	Options	Description
URI	/allocations/{*id*}	
Method	HEAD	Get the header information about an allocation
Query String		
Response	200 401 404	OK and the Location Header (URI of the allocation) Unauthorized Not Found

Table 28: Get Header Info for Specific Allocation

HTTP Method	Options	Description
URI	/allocations/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 29: Get Options Info for Specific Allocation

Storage Pools

Storage Pools on the Array are the logical representation of the physical Mediums. Storage Pools support the following HTTP Methods:

• GET	Returns either the list of pools or a particular pool if specified
• HEAD	Returns the Location Header information about the pool but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for pool
• PUT	Modify an attribute of an existing pool

HTTP Method	Options	Description
URI	/pools	
Method	GET	Returns the list of pool URIs
Query String		
Response	200 401	OK and URI List of pools Unauthorized

Table 30: Get All Storage Pools

HTTP Method	Options	Description
URI	/pools/{*id*}	
Method	GET	Get the information about a specific pool
Query String		
Response	200 401 404	OK and Storage Pool Info in response entity-body Unauthorized Not Found

Table 31: Get Specific Storage Pool Info

HTTP Method	Options	Description
URI	/pools/{*id*}	
Method	HEAD	Get the header information about a storage pool
Query String		
Response	200 401 404	OK and the Location Header (URI of the storage pool) Unauthorized Not Found

Table 32: Get Header Info for Specific Storage Pool

HTTP Method	Options	Description
URI	/pools/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 33: Get Options Info for Specific Storage Pool

HTTP Method	Options	Description
URI	/pools/{id}	
Method	PUT	Modify an existing pool
Query String	flashquota=	Size of the quota in GB

Table 34: Put Option Info for Specific Storage Pool

Mirrors

Mirrors are virtual objects that can span multiple Arrays. They provide three basic functions for data management:

1. LUN Migration—Mirrors can be set up to move data from one Array to another, delete the original LUN. The new LUN assumes the identity of the original LUN.
2. LUN Copy—Mirrors can be set up to synchronize the data from one LUN to another, then break to keep both LUNs in play but with different identities.
3. Resilient LUN—A long-lasting Mirror provides continuous data synchronization between two mirrored LUNs that can be accessed from either side at any time.

Note. This method is not applicable to iSCSI systems.

Mirrors on the Array support the following HTTP Methods:

• GET	Returns either the list of mirrors or a particular mirror if specified
• POST	Creates a new mirror
• PUT	Modifies an attribute of an existing mirror
• HEAD	Returns the Location Header information about the mirror but not the attributes
• DELETE	Breaks a mirror
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for mirror

HTTP Method	Options	Description
URI	/mirrors	
Method	GET	Returns the list of Mirror URIs
Query String		
Response	200 401	OK and URI List of mirrors Unauthorized

Table 35: Get All Mirrors

HTTP Method	Options	Description
URI	/mirrors/{*id*}	
Method	GET	Get the information about a specific mirror
Query String		
Response	200 401 404	OK and Mirror Info in response entity-body Unauthorized Not Found

Table 36: Get Specific Mirror Info

HTTP Method	Options	Description
URI	/mirrors	
Method	POST	Create a new mirror
Query String	name=	Name of the mirror. Max 32 characters. [optional]
	comment=	Comment field. Max 60 characters. [optional]
	hvid =	Sets or clears the **sameHVID** bit. Tells the system whether or not to keep the original identity or take on the new one.
	member=	Form of **{IseId / VolumeGuid}** (must have at least two entries of **member ={ }** in **POST** body. The first member entry is assumed to be the **master member** or **data source** volume. [required]
	type= basic	Specifies the mirror type as **`basic`** (default) or, optionally, one of the following: • **`ha`** • **`copy`** • **`migrate`**
Response	202	Accepted and Location Header (URI of new mirror)
	400	Bad Request (typically a faulty parameter)
	401	Unauthorized
	409	Conflict (typically mirror name already in use)

Table 37: Create Mirror

HTTP Method	Options	Description
URI	/mirrors/{*id*}	
Method	PUT	Modify an existing mirror
Query String	name=	Name of the volume. Max 32 characters.
	comment=	Comment field. Max 60 characters.
Response	201	Created and Location Header (URI of mirror)
	400	Bad Request (typically a faulty parameter)
	401	Unauthorized
	404	Not Found
	409	Conflict (typically mirror name already in use)

Table 38: Modify Mirror

HTTP Method	Options	Description
URI	/mirrors/{*id*}	
Method	DELETE	Delete an existing volume
Query String		
Response	204	No Content (indicates the delete worked)
	401	Unauthorized
	404	Not Found
	409	Conflict (typically mirror is still presented to a host)

Table 39: Delete Mirror

HTTP Method	Options	Description
URI	/mirrors/{*id*}	
Method	HEAD	Get the header information about a mirror
Query String		
Response	200 401 404	OK and the Location Header (URI of the mirror) Unauthorized Not Found

Table 40: Get Header Info for Mirror

HTTP Method	Options	Description
URI	/mirrors/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 41: Get Options Info for Mirror

Physical System Resources

The physical system resources are array, controllers, media, power supplies, and batteries.

Array

The Array-level methods and attributes return information about the Array and do certain operations. The Array supports the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays

• GET	Returns information about the Array
• PUT	Modifies an attribute of the Array or executes a certain operation
• HEAD	Returns the Location Header information about the Array but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for the array

HTTP Method	Options	Description
URI	Base URI http://<*ip_addr*>/storage/arrays/{*id*}	
Method	GET	Returns the current information and attributes of the array
Query String	discoveredarrays =yes	Return any discovered array URI and ID information
Response	200 401	OK and Response Body Unauthorized

Table 42: Get Array Information

Note. Optionally, executing the string

`http://<ip_addr>/storage/arrays/{id}/discoveredarrays`

returns the discovered arrays list portion only, reducing the data collection time.

HTTP Method	Options	Description
URI	*< BaseURI >*/{*id*}	
Method	POST	Create an on-demand alert, general update, or telemetry push

Query String	alert=yes
	generalupdate =yes
	telemetry=yes
	brickinfo =yes
	uds =yes
	smart=yes
	ismart =yes
	drivelogs =yes
	cel =yes
	eft =yes
	memcrashdump =yes
	crashdump =yes
		The system sends a Test Alert to the subscribed locations
		The system sends a General Update to the subscribed locations
		Sends all telemetry information (listed below) to the subscribed locations unless filtered as follows. With **telemetry=yes, send**:
		Brick information
		UDS information
		SMART information
		iSMART information
		Drive logs information
		CEL data
		EFT data
		Memory crash dump
		Crash dump

Response	201 400 401 404	Created and Location Header (URI of Array) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 43: On-Demand Alert, General Update, Telemetry--Array to Remote

HTTP Method	Options	Description
URI	*< BaseURI >*/{*id*}	
Method	PUT	Modify an attribute or execute an operation

Query String	led= <enabled | disabled>
	name=
	address=
	location=
	contactname =
	contactphone =
	contactemail =
	initialize=true
	reformat=true
	restart=true
	shutdown=true
	qosmode=<*enabled | disabled*>
		Toggles the Activity LED on the system display module
		Name of the Array. Max 32 characters.
		Address field. Max 60 characters.
		Location field. Max 60 characters.
		Contact Name field. Max 60 characters.
		Contact Phone field. Max 16 characters.
		Contact E-mail field. Max 60 characters.
		Initializes the Array
		Reformats the Array to Factory Defaults
		Restarts the Array (cold boot)
		Shutdown the Array (complete power down)
		NOTE: If Wake-On-LAN is enabled, a WOL Magic Packet powers up the Array (see **/network**)
		Toggles the global Quality of Service (QoS) mode. Default is enabled. Disabled will override volume settings. The QoS feature is not in all firmware versions.

HTTP Method	Options	Description
Response	201 400 401 404	Created and Location Header (URI of Array) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 44: Modify Specific Array

HTTP Method	Options	Description
URI	< *BaseURI* >/{*id*}	
Method	HEAD	Get the header information about the Array
Query String		
Response	200 401 404	OK and the Location Header (URI of Array) Unauthorized Not Found

Table 45: Get Header Info for Specific Array

HTTP Method	Options	Description
URI	< *BaseURI* >/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 46: Get the Options Info for Array

Controllers (MRC)

There are two Managed Reliability Controllers on the Array. Controllers on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/controllers

• GET	Returns either the list of Controllers or a particular Controller if specified
• POST	Adds a Controller to the Array
• PUT	Modifies an attribute of Controller
• HEAD	Returns the Location Header information about the Controller but not the attributes
• DELETE	Removes a Controller from the Array
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for controller

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/controllers	
Method	GET	Returns the list of Controller URIs
Query String		
Response	200 401	OK and URI List of Controllers Unauthorized

Table 47: Get All Controllers

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	GET	Get the information about a specific Controller
Query String		
Response	200 401 404	OK and Controller Info in response entity-body Unauthorized Not Found

Table 48: Get Specific Controller Info

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	POST	Add a Controller to the system
Query String		
Response	202 400 401 404	Accepted and Location Header URI to check status Bad Request Unauthorized Not Found

Table 49: Add Controller

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	PUT	Modify an existing Controller

Query String	led=<*enabled*> | <*disabled*>
	
	fcportspeed ={auto | <*speed* > }
		Toggles the Activity LED on the Controller
		
		Sets ISE Fibre Channel port speed, **auto** indicates to auto-negotiate with the fabric speed and ***<speed >*** specifies in Gbps as: ISE [**1**, **2**, or **4**] and ISE-2 [**2**, **4**, or **8**].

Response	200 400 401 404	Created and Location Header (URI of Controller) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 50: Modify Controller

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	DELETE	Delete an existing Controller from the system
Query String	force=true	Forces the remove independent of status [optional]
Response	202 401 404 409	Accepted and Location Header URI to check status Unauthorized Not Found Conflict (typically system is busy)

Table 51: Remove Specific Controller

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	HEAD	Get the header information about a Controller
Query String		
Response	200 401 404	OK and the Location Header (URI of the Controller) Unauthorized Not Found

Table 52: Get Header Info for Specific Controller

HTTP Method	Options	Description
URI	/controllers/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 53: Get Options Info for Specific Controller

Media

The Media are the sealed-drive containers. The Media are the physical representation of the logical Storage Pool. Media on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/media

• GET	Returns either the list of Mediums or a particular Medium if specified
• POST	Adds a Medium to the Array
• PUT	Modifies an attribute of Medium
• HEAD	Returns the Location Header information about the Medium but not the attributes
• DELETE	Removes a Medium from the Array
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for medium

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/media	
Method	GET	Returns the list of Media URIs
Query String		
Response	200 401	OK and URI List of Media Unauthorized

Table 54: Get All Media

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	GET	Get the information about a specific Medium
Query String		
Response	200 401 404	OK and Medium Info in response entity-body Unauthorized Not Found

Table 55: Get Specific Medium Info

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	POST	Add a Medium to the system
Query String		
Response	202 400 401 404	Accepted and Location Header URI to check status Bad Request Unauthorized Not Found

Table 56: Add Medium

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	PUT	Modify an existing Medium
Query String	led=<enabled | disabled>	Toggles the Activity LED on the Medium
Response	200 400 401 404	Created and Location Header (URI of Medium) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 57: Modify Specific Medium

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	DELETE	Delete an existing Medium from the system
Query String	erase=true	Executes a secure erase of the Medium before removing from the system [optional]
Response	202 401 404 409	Accepted and Location Header URI to check status Unauthorized Not Found Conflict (typically system is busy)

Table 58: Remove Specific Medium

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	HEAD	Get the header information about a Medium
Query String		
Response	200 401 404	OK and the Location Header (URI of the Medium) Unauthorized Not Found

Table 59: Get Header Info for Specific Medium

HTTP Method	Options	Description
URI	/media/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the Allow list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 60: Get Options Info for Specific Medium

Power Supplies

Power Supplies on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/powersupplies

• GET	Returns either the list of Power Supplies or a particular Power Supply if specified
• PUT	Modifies an attribute of Power Supply
• HEAD	Returns the Location Header information about the Power Supply but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for power supply

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/powersupplies	
Method	GET	Returns the list of Power Supply URIs
Query String		
Response	200 401	OK and URI List of Power Supplies Unauthorized

Table 61: Get All Power Supplies

HTTP Method	Options	Description
URI	/ powersupplies /{*id*}	
Method	GET	Get the information about a specific Power Supply
Query String		
Response	200 401 404	OK and Power Supply Info in response entity-body Unauthorized Not Found

Table 62: Get Specific Power Supply Info

HTTP Method	Options	Description
URI	/ powersupplies /{*id*}	
Method	PUT	Modify an existing Power Supply
Query String	led=<enabled | disabled>	Toggles the Activity LED on the Power Supply
Response	200 400 401 404	Created and Location Header (URI of Power Supply) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 63: Modify Specific Power Supply

HTTP Method	Options	Description
URI	/ powersupplies /{*id*}	
Method	HEAD	Get the header information about a Power Supply
Query String		
Response	200 401 404	OK and the Location Header (URI of the Power Supply) Unauthorized Not Found

Table 64: Get Header Info for Specific Power Supply

HTTP Method	Options	Description
URI	/ powersupplies /{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 65: Get Options Info for Specific Power Supply

Batteries

Batteries on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/batteries

- GET Returns either the list of Batteries or a particular Battery if specified
- PUT Modifies an attribute of Battery
- HEAD Returns the Location Header information about the Battery but not the attributes
- OPTIONS Returns the allowable HTTP Methods plus the XML Schema Document for battery

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/batteries	
Method	GET	Returns the list of Battery URIs
Query String		
Response	200 401	OK and URI List of Batteries Unauthorized

Table 66: Get All Battery Info

HTTP Method	Options	Description
URI	/batteries/{*id*}	
Method	GET	Get the information about a specific Battery
Query String		
Response	200 401 404	OK and Battery Info in response entity-body Unauthorized Not Found

Table 67: Get Specific Battery Info

HTTP Method	Options	Description
URI	/batteries/{*id*}	
Method	PUT	Modify an existing battery

Query String	led=<*enabled* | *disabled*>
	upsmode =<*enabled* | *disabled*>
		Toggles the Activity LED on the Battery
		Indicates UPS power protection active. This is not applicable to ISE-2.

Response	200 400 401 404	Created and Location Header (URI of Battery) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 68: Modify Specific Battery

HTTP Method	Options	Description
URI	/batteries/{*id*}	
Method	HEAD	Get the header information about a Battery
Query String		
Response	200 401 404	OK and the Location Header (URI of the Battery) Unauthorized Not Found

Table 69: Get Header Info for Specific Battery

HTTP Method	Options	Description
URI	/batteries/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 70: Get Options Info for Specific Battery

Logical System Resources

The logical system resources are network, chronometer, jobs, files, subscriptions, revision, and performance.

Network

The Network resource represents the networking status and capabilities of the Array. It includes the two IP Address network information as well as the DHCP and Wake-On-LAN capabilities.

The Ethernet Network on the Array supports the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{id}/network

• GET	Returns list of network ports or a particular port if specified
• PUT	Modifies a Network attribute
• HEAD	Returns the Location Header information about the Network but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for network

HTTP Method	Options	Description
URI	http://<*ip_addr*>/ storage/arrays/{*id*}/ network	
Method	GET	Get the system network information
Query String		
Response	200 401 404	OK and Network Info in response to entity-body Unauthorized Not Found

Table 71: Get Network information

HTTP Method	Options	Description
URI	/network/ports/ {*id*}?< *query_string* >={*value*}	
Method	PUT	Modify an existing network attribute for Port 1 or Port 2
Query String	Ipaddress Ipmask Gateway dhcp =<*enabled* | *disabled*> wakeonlan =<*enabled* | *disabled*>	Change the IP Address of Network Port <*x*> Change the IP Mask of Network Port <*x*> Change the IP Gateway of Network Port <*x*> Setup the DHCP capability Setup the Wake-On-LAN capability
Response	200 400 401 404	Created and Location Header (URI of Network) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 72: Modify Specific Network Attribute

HTTP Method	Options	Description
URI	/network	
Method	HEAD	Get the header information about Network
Query String		
Response	200 401 404	OK and the Location Header (URI of Network) Unauthorized Not Found

Table 73: Get Header Info for Specific Network

HTTP Method	Options	Description
URI	/network	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 74: Get Options Info for Specific Network

Chronometer

The Chronometer resource represents the Array system clock. It contains the date, time, time zone, and NTP information and settings.

The Chronometer on the Array supports the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/chronometer

- GET Returns the Chronometer (Date, Time, and Time Zone) information
- PUT Modifies an attribute of the Chronometer system
- HEAD Returns the Location Header information about the Chronometer but not the attributes
- OPTIONS Returns the allowable HTTP Methods plus the XML Schema Document for chronometer

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/chronometer	
Method	GET	Get the system chronometer information
Query String		
Response	200 401 404	OK and Chronometer Info in response entity-body Unauthorized Not Found

Table 75: Get Chronometer Info

HTTP Method	Options	Description
URI	/chronometer?<*query_string*>={*value*}	
Method	PUT	Modify an existing network attribute
Query String	date=<*ss-mmm-yyyy*>	Change the date. Format is 01-Jan-2010
	time=<*hh:mm:ss*>	Change the time. Format is 23:00:00
	timezone =<*tz_value*>	Change the time zone. There is a list of entries to choose from.
	dst =<*enabled* | *disabled*>	Set up the automatic daylight saving setting
	ntpmode={automatic | static | disabled}	Set the NTP Mode
	ntpserver=<*ntp server ip*> | <*name*>	Set the NTP Server location
Response	200 400 401 404	Created and Location Header (URI of Chronometer) Bad Request (typically a faulty parameter) Unauthorized Not Found

Table 76: Modify Chronometer Attribute

HTTP Method	Options	Description
URI	/chronometer	
Method	HEAD	Get the header information about the Chronometer
Query String		
Response	200 401 404	OK and the Location Header (URI of Chronometer) Unauthorized Not Found

Table 77: Get Header Info for Specific Chronometer

HTTP Method	Options	Description
URI	/chronometer	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 78: Get Options Info for Specific Chronometer

Jobs

The Array supports just a few jobs or operations that can be long-lived and monitored. These include the following operations:

1. Add and Remove Controller
2. Add and Remove Medium
3. Update FW (Array and disk drive-level firmware)

Jobs on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/chronometer

• GET	Returns the list of current jobs or a particular job information
• DELETE	Acknowledges a completed job and removes it from the list
• HEAD	Returns the Location Header information about the job but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for jobs

HTTP Method	Options	Description
URI	http://<*ip_addr*>/ storage/arrays/{*id*}/ jobs	
Method	GET	Returns the list of Job URIs
Query String		
Response	200 401	OK and URI List of Jobs Unauthorized

Table 79: Get All Jobs

HTTP Method	Options	Description
URI	/jobs/{*id*}	
Method	GET	Get the specific job information
Query String		
Response	200 401 404	OK and Job Info in response entity-body Unauthorized Not Found

Table 80: Get Info for Specific Job

HTTP Method	Options	Description
URI	/job/{*id*}	
Method	DELETE	Acknowledge a specific job as complete and remove it from the / jobs list
Query String		
Response	204 401 404	No Content Unauthorized Not Found

Table 81: Delete (Acknowledge) Specific Job

HTTP Method	Options	Description
URI	/jobs/{I}	
Method	HEAD	Get the header information about the Job
Query String		
Response	200 401 404	OK and the Location Header (URI of Job) Unauthorized Not Found

Table 82: Get Header Info for Specific Job

HTTP Method	Options	Description
URI	/job/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 83: Get Options Info for Specific Job

Files

The Array supports retrieving the following files from the system:

1. Syslog (per Controller)
2. Event Log (file system-based log that is currently available via the CLI and Web GUI; not CEL; can be deleted)
3. Management Log (can be deleted)
4. Upgrade Log
5. Modules Log
6. SNMP MIB
7. Diagnostic Report (per Controller; similar to the old XioWebService; this is a gzip/tarball file)

   Includes:
   - /var/log/*
   - /proc/*

Files on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/files

• GET	Returns the list of available Files
• DELETE	Some files are allowed to be deleted (some logs, etc.)
• HEAD	Returns the Location Header information about the file but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for files

HTTP Method	Options	Description
URI	http://<*ip_addr*>/ storage/arrays/{*id*}/ files	
Method	GET	Returns the list of File URIs
Query String		
Response	200 401	OK and URI List of Files Unauthorized

Table 84: Get All Files

HTTP Method	Options	Description
URI	/files/{*name*}	
Method	GET	Get the specific file information
Query String		
Response	200 401 404	OK and File Info in response entity-body Unauthorized Not Found

Table 85: Get Info for Specific File

HTTP Method	Options	Description
URI	/file/{*filename*}	
Method	DELETE	Delete a log file (/event and /mgmt only) NOTE: On the next event or mgmt entry, the file automatically is created by the system
Query String		
Response	204 401 404	No Content Unauthorized Not Found

Table 86: Delete Specific File

HTTP Method	Options	Description
URI	/file/{*filename*}	
Method	HEAD	Get the header information about the File
Query String		
Response	200 401 404	OK and the Location Header (URI of File) Unauthorized Not Found

Table 87: Get Header Info for Specific File

HTTP Method	Options	Description
URI	/file/{*filename*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 88: Get Options Info for Specific File

Subscriptions

Subscriptions are what a RESTful client can set up for receiving RESTful Alerts from the Array. These subscriptions on the Array support the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/subscriptions

• GET	Returns the list of current subscription information
• POST	Create a new subscription
• DELETE	Delete a subscription
• HEAD	Returns the Location Header information about a subscription but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for **subscriptions**

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/subscriptions	
Method	GET	Returns the list of Subscriptions URIs
Query String		
Response	200 401	OK and URI List of Subscriptions Unauthorized

Table 89: Get All Subscriptions

HTTP Method	Options	Description
URI	/subscriptions/{*id*}	
Method	GET	Get the information about a specific Subscription
Query String		
Response	200 401 404	OK and Subscription Info in response entity-body Unauthorized Not Found

Table 90: Get Specific Subscription Info

HTTP Method	Options	Description
URI	/subscriptions	
Method	POST	Add a Subscription to the system
Query String	id= setting=<*enabled* | *disabled*> interval=<*interval*> type={alert | malert | gupdate | telemetry} ssl={enabled | disabled} starthour= startminute= proxy=*enabled* | *disabled*> proxyaddress= proxyusername= proxypassword=	IP Address or DNS name of the destination listener Sets ID to be enabled or disabled upon creation [default is enabled] Specifies the subscription type interval [gupdate = 2 to 1440; telemetry = 45 to 1440] Specifies the type of subscription to create Specifies whether or not to connect with SSL Specifies hour to send gupdate or telemetry [0 to 23; default=-1] Specifies minute to send gupdate or telemetry [0 to 59; default=-1 Enable or disable proxy services Specify the proxy address Specify the proxy user ID Specify the proxy user password
Response	201 400 401 404	Created and Location Header URI for this Subscription Bad Request Unauthorized Not Found

Table 91: Add Subscription

HTTP Method	Options	Description
URI	/subscriptions/{*id*}	
Method	PUT	Modify a Subscription in the system
Query String	setting=<enabled | disabled> interval= type={alert | gupdate | telemetry} starthour= startminute= proxy=*enabled* | *disabled*> proxyaddress= proxyusername= proxypassword=	Toggles the subscription to be either enabled or disabled Specifies the new interval [2 to 1440 for gupdate; 45 to 1440 for telemetry] Specifies the type of subscription to modify Specifies the new start hour Specifies the new start minute Enable or disable proxy services Specify the proxy address Specify the proxy user ID Specify the proxy user password
Response	201 400 401 404	Created and Location Header URI for this Subscription Bad Request Unauthorized Not Found

Table 92: Modify Specific Subscription

HTTP Method	Options	Description
URI	/subscriptions/{*id*}	
Method	DELETE	Delete an existing Subscription from the system
Query String		
Response	204 401 404	Not Content Unauthorized Not Found

Table 93: Remove Specific Subscription

HTTP Method	Options	Description
URI	/subscriptions/{*id*}	
Method	HEAD	Get the header information about a Subscription
Query String		
Response	200 401 404	OK and the Location Header (URI of the Subscription) Unauthorized Not Found

Table 94: Get Header Info for Specific Subscription

HTTP Method	Options	Description
URI	/subscriptions/{*id*}	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 95: Get Options Info for Specific Subscription

Revision

This resource returns the current status/state of both the Array and disk drive-level FW update. It is also the entry point for doing FW updates (using POST). The response header would point the client to a /jobs resource to monitor progress of an update.

Revision on the Array supports the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/revision

• GET	Returns the current Array and disk drive-level FW Update information
• POST	Uploads the firmware image and Updates the ISE or disk drive-level FW system
• HEAD	Returns the Location Header information about the Revision but not the attributes
• OPTIONS	Returns the allowable HTTP Methods plus the XML Schema Document for revision

HTTP Method	Options	Description
URI	http://<*ip_addr*>/storage/arrays/{*id*}/revision	
Method	GET	Get the information about the system Revision
Query String		
Response	200 401 404	OK and Revision Info in response entity-body Unauthorized Not Found

Table 96: Get Revision

HTTP Method	Options	Description
URI	/revision	
Method	POST	Upload and Update either the Array or disk drive-level FW
Query String	file=<*filename*>	The FW image file to upload and use in the update process
Response	202 400 401 404	Accepted and Location Header URI to track progress Bad Request Unauthorized Not Found

Table 97: Start Revision Update

HTTP Method	Options	Description
URI	/revision	
Method	HEAD	Get the header information about Revision
Query String		
Response	200 401 404	OK and the Location Header (URI of the Revision) Unauthorized Not Found

Table 98: Get Header Info for Revision

HTTP Method	Options	Description
URI	/revision	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body
	401	Unauthorized
	404	Not Found

Table 99: Get Options Info for Revision

Performance

This resource returns an instantaneous snapshot of the current performance values for several Array resources at once. The performance attributes of a given resource is as follows:

1. Total I/Os per second
2. Read I/Os per second
3. Write I/Os per second
4. Total Kilobytes per second
5. Read Kilobytes per second
6. Write Kilobytes per second
7. Average Read Latency in milliseconds
8. Average Write Latency in milliseconds
9. Current Queue Depth
10. Read Percentage
11. Average Transfer Size in bytes

These Array resources support the Performance resource:

a. Array (Grand Totals)
b. Controllers
c. Storage Volumes
d. Endpoints (Host Port WWNs)
e. Media per back-end drive (NOTE: This data may not be available to end-users)

Performance on the Array supports the following HTTP Methods:

Base URI = http://<*ip_addr*>/storage/arrays/{*id*}/performance

- GET Returns the current Performance information
- HEAD Returns the Location Header information about the Performance resource but not the attributes
- OPTIONS Returns the allowable HTTP Methods plus the XML Schema Document for performance

HTTP Method	Options	Description
URI	http://<*ip_addr*>/ storage/arrays/{*id*}/ performance	
Method	GET	Get the information about the system Performance
Query String		
Response	200 401 404	OK and Revision Info in response to entity-body Unauthorized Not Found

Table 100: Get Performance Data

HTTP Method	Options	Description
URI	/performance	
Method	HEAD	Get the header information about Performance
Query String		
Response	200 401 404	OK and the Location Header (URI of the Performance) Unauthorized Not Found

Table 101: Get Header Info for Performance

HTTP Method	Options	Description
URI	/performance	
Method	OPTIONS	Get the allowable HTTP Methods list for this resource
Query String		
Response	200 401 404	OK and the **Allow** list of methods in the response header plus the XSD in the entity-body Unauthorized Not Found

Table 102: Get the Options Info for Performance

HTTP Method	Options	Description
URI	/performance	
Method	GET	Get information about the system performance
Query String	scale=usec	Returns Latency Counter Information in microseconds instead of the default of milliseconds
Response	201 401 404	Get successful and Location Header (URI of Performance) Unauthorized Not Found

Table 103: Get Performance Data

HTTP Method	Options	Description
URI	/performance	
Method	PUT	Reset all the Maximum Value Counters to zero
Query String	reset=true	Resets read & write latency max and queue depth max counters to zero (**0**) NOTE: This does not include the counters Total I/Os and Total KBs since boot.
Response	201 400 401	Reset successful and location header (URI of Performance) Bad request (typically a faulty parameter) Unauthorized

Table 104: Reset Performance Maximum Counters

I/O Networks (iSCSI ISE only)

This resource returns I/O network lists, location header information, and HTTP methods and modifies I/O Networks. I/O Networks on the Array support the following HTTP Methods:

Base URI = http://<ip_addr>/storage/arrays/{id}/ionetworks

• GET	Returns either a list of I/O networks or a particular I/O network when specified
• PUT	Modifies an I/O network attribute
• HEAD	Returns an I/O network ‘s Location Header information but not its attributes
• OPTIONS	Returns the allowable HTTP Methods

HTTP Method	Options	Description
URI	http://<ip_addr>/storage/arrays/{id}/ionetworks	
Method	GET	Returns the list of I/O network URIs
Query String		
Response	200 401	OK and list of I/O network URIs Unauthorized

Table 105: Get All I/O Networks

HTTP Method	Options	Description
URI	/ionetworks/{a | b}	
Method	GET	Get information on a specific I/O network Supported parameter values: **/a, /b**, **/A**, **/B**, **/1**, or **/2**
Query String		
Response	200 401 404	OK and list of I/O network URIs Unauthorized Not Found

Table 106: Get Specific I/O Network Data

HTTP Method	Options	Description
URI	/ionetworks/{a | b}	
Method	PUT	Modify an existing I/O network
Query String	nameserver={iSNSaddr}	Modify the iSNS name server; use with no specific URI specified since this is a global setting
	chapin={enabled | disabled}	iSCSI incoming CHAP setting (for single direction CHAP)
	OR	
	chapout={enabled | disabled}	iSCSI outgoing CHAP setting (for dual direction CHAP)
	username={name}	Username for either IN or OUT setting (1 to 255 characters)
	password={word}	Password for either IN or OUT setting (must be 12 to 16 characters)
	dhcp={enabled | disabled}	Modify the iSCSI network DHCP setting
	mtu={setting}	Modify the maximum transfer unit. Possible settings: {1500 | standard | Standard} or {9000 | jumbo | Jumbo}
	ipaddress1={ipaddr}	Modify IP Address 1 setting
	ipaddress2={ipaddr}	Modify IP Address 2 setting
	ipmask={ipmask}	Modify IP Mask setting
	gateway={ipaddr}	Modify Gateway setting
Response	201	Successful modification and Location Header (URI of I/O Network)
	400	Erroneous request (typically a faulty parameter)
	401	Unauthorized
	404	Not Found

Table 107: Modify I/O Network

HTTP Method	Options	Description
URI	/ionetworks/{id}	
Method	OPTIONS	Return list of supported HTTP Methods for this resource Should return: **GET**, **PUT**, **HEAD**, and **OPTIONS**
Query String		
Response	200 401 404	OK and supported methods for URI specified Unauthorized Not Found

Table 108: Get I/O Network Options Data

I/O Ports (iSCSI ISE only)

This resource returns I/O port information for the specified I/O Networks. I/O Networks on the Array support the following HTTP Methods:

Base URI = http://<ip_addr>/storage/arrays/{id}/ioports

- GET Returns either a list of I/O ports or a particular I/O port when specified
- PUT Modifies an I/O network attribute
- HEAD Returns an I/O port ‘s Location Header information but not its attributes
- OPTIONS Returns the allowable HTTP Methods

HTTP Method	Options	Description
URI	http://<ip_addr>/ storage/arrays/{id}/ ports	
Method	GET	Returns the list of I/O port URIs
Query String		
Response	200 401	OK and list of I/O port URIs Unauthorized

Table 109: Get All I/O Ports

HTTP Method	Options	Description
URI	/ioports/{1 - 4}	
Method	GET	Retrieve port data for a specific I/O port Supported parameter values: **1 2**, **3**, and **4**
Query String		
Response	200 401 404	OK and I/O port data for specified port number Unauthorized Not Found

Table 110: Get Specific I/O Port Data

HTTP Method	Options	Description
URI	/ioports/{1 - 4}	
Method	HEAD	Retrieve Location Header data for a specific I/O port Supported parameter values: **1 2**, **3**, and **4**
Query String		
Response	200 401 404	OK and I/O port data for specified port number Unauthorized Not Found

Table 111: Get Specific I/O Port Header Data

HTTP Method	Options	Description
URI	/ioports/{1 - 4}	
Method	OPTIONS	Retrieve supported methods for a specific I/O port Supported parameter values: **1 2**, **3**, and **4**
Query String		
Response	200 401 404	OK and supported I/O port methods for specified port number Unauthorized Not Found

Table 112: Get Specific I/O Port Options List

SNMP

This resource returns and modifies current SNMP configuration for the specified ISE. The SNMP method supports the following:

Base URI = http://<ip_addr>/storage/arrays/{id}/snmp

• GET	Returns current SNMP configuration
• POST	Adds an SNMP attribute to the configuration
• PUT	Modifies an SNMP attribute
• DELETE	Removes an SNMP attribute from the configuration
• HEAD	Returns the SNMP Location Header information but not its attributes
• OPTIONS	Returns the allowable HTTP Methods

HTTP Method	Options	Description
URI	http://<ip_addr>/storage/arrays/{id}/snmp	
Method	GET	Returns the current SNMP attribute list
Query String		
Response	200 401	OK and list of SNMP attributes Unauthorized

Table 113: Get SNMP Configuration

HTTP Method	Options	Description
URI	/snmp/	
Method	POST	Add a new Client and/or Trap Subscription

Query String	client={ip_address}
	trap={trap_id}
		Add SNMP Trap Destination Client IP Address
		Add SNMP Trap Subscription

Response	201 400 401 409	Successful and Location Header (URI for SNMP) Bad request Unauthorized Conflict, client IP or Trap Subscription already exists

Table 114: Create an SNMP Configuration Attribute

HTTP Method	Options	Description
URI	/snmp	
Method	PUT	Modify an SNMP configuration attribute

Query String	community={string}
	organization={string}
	contact={string}
	description={string}
		Modify the SNMP Community String
		Modify the Organization String
		Modify the Contact String
		Modify the Description String

Response	201 400 401	Successful and Location Header (URI for SNMP) Bad request (typically an erroneous parameter) Unauthorized

Table 115: Modify an SNMP Configuration Attribute

HTTP Method	Options	Description
URI	/snmp/	
Method	DELETE	Delete a Client and/or Trap Subscription

Query String	client={ip_address}
	trap={trap_id}
		Delete SNMP Trap Destination Client IP Address
		Delete SNMP Trap Subscription

Response	201 400 401 409	Successful delete, no content Bad request Unauthorized Conflict, client IP or Trap Subscription already exists

Table 116: Delete an SNMP Client or Trap Subscription Attribute

HTTP Method	Options	Description
URI	/snmp	
Method	HEAD	Returns the SNMP header data
Query String		
Response	200 401 404	OK and SNMP Location Header Unauthorized Not found

Table 117: Get SNMP Header Data

HTTP Method	Options	Description
URI	/snmp	
Method	OPTIONS	Retrieves supported HTTP Methods list for this resource Should return: GET, POST, PUT, DELETE, HEAD, OPTIONS
Query String		
Response	200 401 404	OK and list of supported SNMP methods Unauthorized Not found

Table 118: Get SNMP Options Data

Using Conditional GET, PUT

The Emprise 5000 ISE firmware (consult the *Emprise 5000 Release Notes* for versions) supports Web client usage of the HTTP conditional **GET** and **PUT** methods. This includes an entity tag (ETag) returned in most **GET** and **PUT** request response headers. The ETag is used to determine if certain resources have **changed state** since the last time the Web client requested information regarding that resource or list of resources.

CorteX Conditional GET

Use the conditional CorteX **GET** to determine whether a given resource or list of resources have **changed state** since the last time the Web client requested information regarding this given resource or list of resources. The Web client receives an ETaq as part of the response header than can be used to pass the **`If-Not-Modified`** conditional to the Web service to determine whether the specific resource has changed state between HTTP **GET** methods.

For example the following Web client request:

```
GET /storage/volumes
```

Receives a response like:

```
HTTP/1.0 200 OK

Etag: "7acd9b16a47ed1297c4ff34d79fd8c9b"

Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"?><volumes self="http://........
```

The Web client then captures and saves the ETag from the response for use in subsequent requests. For example, the following **GET** request.

```
GET /storage/volumes

If-Not-Modified: "7acd9b16a47ed1297c4ff34d79fd8c9b"
```

Response from CorteX:

```
HTTP/1.0 304 Not Modified

Etag: "7acd9b16a47ed1297c4ff34d79fd8c9b"

Content-length: 0
```

The Web client recognizes the **`304 Not Modified`** response because the resource or list of resources is not changed from the last retrieved information. Using the conditional **GET** significantly reduces network traffic by eliminating unnecessary duplicate data transfers. This is a good means for Web clients to execute frequent status polls.

Benefits

Because the potential network traffic reduction with the conditional **GET** method, the **`returned=optional`** query string parameter is removed from the CorteX API on the participating resources and all responses are fully verbose. GET responses are now either fully verbose or fully silent when the Web client uses the ETag with **`If-Not-Modified`** conditional request header.

CorteX Conditional PUT

Use of the conditional **PUT** method including the conditional parameter **`If-Match:`** followed by the entity tag (ETag) conditionally executes the command. The ETag is returned in most GET method response headers. When used with the HTTP PUT method, the Web client can specify to execute the command if the resource has not changed since the last receipt of a GET response for a given resource. This is called **optimistic concurrency** as opposed to **pessimistic concurrency** where an explicit lock is used to handle multiple Web clients.

For example the following Web client request:

```
GET /storage/volumes
```

Receives a response like:

```
HTTP/1.0 200 OK

Etag: "7acd9b16a47ed1297c4ff34d79fd8c9b"

Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"?><volumes self="http://........
```

The Web client then captures and saves the ETag from the response for use in subsequent requests. For example, the following **PUT** request.

```
PUT /storage/volume/{id}?size=44

If-Match: "7acd9b16a47ed1297c4ff34d79fd8c9b"
```

Condition–Unchanged

Response from CorteX if the specified resource is unchanged:

```
HTTP/1.0 201 Created

Location: http://10.20.54.32/storage/volumes/{id}

Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"?><response value="2">A volume was
modified.</response>
```

Condition–Changed

The Web client requests the following conditionally.

```
PUT /storage/volume/{id}?size=44

If-Match: "7acd9b16a47ed1297c4ff34d79fd8c9b"
```

Response from CorteX if the specified resource is changed (by some other Web client), the ETag is expired, and the change request is discarded as below:

```
HTTP/1.0 412 Precondition Failed

Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"?><response value="412">Precondition
Failed: Unexpected change occurred since last GET</response>
```

CorteX RESTful Alerts

The Array is capable of sending RESTful alerts (as a RESTful Client) to a RESTful Array Alert Listening system (service). For example, the XENITH architecture includes a RESTful Alert Listener.

To receive Array RESTful alerts, a Client uses the HTTP POST to the /subscription resource and input its own IP Address or a separate proxy listening system IP Address in the URI. Once the Array receives the request, the Array adds the client address to its **alert list**, and whenever an alert is to be sent all subscribed systems receive the alert.

The format of the alert is: **https://< ip_addr_of_listener>/storage/arrays/{id}/{alert_payload}**

Where **/arrays/{id}** is the identity of the sending Array and **{alert_payload}** contains the alert information.

The listener is responsible to store or act upon the reception of alerts. Typically, the alert information is in either the event or management log file resources located in **/files** on the Array.

CorteX RESTful General Updates

The Array is capable of sending RESTful General Updates (as a RESTful Client) to a RESTful Array General Update Listening system (service). For example, the XENITH architecture includes a RESTful General Update Listener.

To receive Array RESTful alerts, a Client uses the HTTP POST to the /subscription resource and input its own IP Address or a separate proxy listening system IP Address in the URI. Once the Array receives the request, the Array adds the client address to its **general update list**, and whenever a general update is to be sent all subscribed systems receive the general update.

The format of the alert is **http://< ip_addr_of_listener>/storage/arrays/{id}/{generalupdate_payload}**

where **/arrays/{id}** is the identity of the sending Array and **{generalupdate _payload}** contains the general update information.

The listener is responsible to store or act upon the reception of general update. Typically, the general update information is a roll-up of all the configuration and status information regarding the Array.

CorteX RESTful Telemetry

The Array is capable of sending RESTful Telemetry (as a RESTful Client) to a RESTful Array Telemetry Listening system (service). For example, the XENITH architecture includes a RESTful Telemetry Listener.

To receive Array RESTful telemetry, a Client uses the HTTP POST to the /subscription resource and input its own IP Address or a separate proxy listening system IP Address in the URI. Once the Array receives the request, the Array adds the client address to its **telemeter** so that whenever a telemetry is to be sent, all subscribed systems receive the telemetry.

The format of the alert is: **http://< ip_addr_of_listener>/storage/arrays/{id}/{telemetry_payload}**

Where /arrays/{id} is the identity of the sending Array and {telemetry _payload} contains the general update information.

The listener is responsible to store or act upon the reception of telemetry. Typically, the telemetry information is a roll up of all the telemetry information regarding the Array.

X-IO Technologies
Xiotech Corporation
9950 Federal Drive, Suite 100
Colorado Springs, CO 80921

www.X-IO.com