ActiveEfficiency Web API

The ActiveEfficiency Web API layer is designed to be simple, scalable, and to allow consumption of the services using any programming or scripting language. The architecture is based on REST and is accessible via the HTTP protocol. These notes will be referenced from any document that mentions the RESTful architecture. Use this API if you are developing a script for use with the AppClarity device tagging feature or to provide site and subnet information for Nomad's single-site download feature.

REST stands for Representational State Transfer. It refers to an architecture used to create stateless services on the web that are typically accessed using the HTTP protocol. Software developed using this architecture are considered to be RESTful. RESTful architectures generally don't use the technologies associated with traditional web services, such as the SOAP protocol or WSDL. They are limited to the standard GET, POST, PUT, and DELETE HTTP methods. This results in a simple, well defined web API that can be accessed by any client program that can transmit network messages using HTTP, regardless of programming language.

A concept central to RESTful architectures is the idea of resources. A RESTful resource can be any object or action that can be addressable using a URI (Uniform Resource Identifier). URLs, as used on the Web, are a type of URI, used to identify a page or web resource. The basic function of the HTTP methods used in REST are:

HTTP method	Description
GET	Returns information concerning the resource in the body of the HTTP response
PUT	Updates the resource if it already exists based on information passed in the body of the HTTP request. An HTTP response code of 404 will be returned if the resource does not exist.
POST	Creates a new instance of a resource based on information passed in the body of the HTTP request
DELETE	Removes the resource

What are resources?

Within a RESTful architecture, resources are the objects or actions that are addressable via a Uniform Resource Identifier (URI). Each resource will have a specific URI associated with. The resource URI, the HTTP method, possible query parameters and the HTTP request body sent to the ActiveEfficiency web API determine the resource to be accessed and what action that resource needs to take. Resources can often be addressed individually as well as within groups. For example, you can request to view a list of devices, or you can request to view the details of a single devices.

RESTful API HTTP response codes

The 1E ActiveEfficiency REST APIs use the HTTP protocol for sending and retrieving data. Client code utilizing the REST APIs make an HTTP request to the ActiveEfficiency web service and process the HTTP response accordingly. Included with the HTTP response data is the HTTP response code. The response code gives the client some indication as to the success of the HTTP request, and can provide information on how the client should handle the included response data. Some of the most common response codes are:

- 200 - The request succeeded.
- 201 - The resource was created.
- 202 - The request has been accepted for processing.
- 204 - The request has been processed and no content was returned.
- 304 - The resource was not modified. (Returned for conditional GET when resource is not modified.)
- 400 - The request was not valid. The request may be incorrect, or the data in the request is not in the correct format. Check the message details for more information concerning the bad request, correct the request data, and try the invocation again.
- 401 - The request requires user authentication. Note that if the request already included valid credentials, then the 401 response indicates that authorization has been refused by ActiveEfficiency for those credentials. Check to make sure the authorization credentials provided are correct, and that the credentials have the correct authority for the action being requested. The security subsystem has not been developed at the time of writing this page, but whatever we come up with, it will support this response code.
- 404 - The 1E ActiveEfficiency web service has not found anything matching the resource specified in the request URI. (Check to make sure the URI is well-formed and correct.)
- 409 - A conflict occurred with the specified resource in the URI.
- 500 - The 1E ActiveEfficiency web service encountered an unexpected condition which prevented it from processing the request.
- 503 - The 1E ActiveEfficiency web service needed services which were not available to process the request.

For a complete listing of the HTTP response codes and their meanings, see http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

Resources supported by the ActiveEfficiency Discovery Web API

The following resource representations are supported by the ActiveEfficiency Web API for third party use:

Devices

This is the primary resource that represents the hardware being discovered eg a desktop, server, laptop, phone or even mobile phone. The following HTTP URI's and methods are supported for this resource:

URI	Method	Description

`http://<server>/ActiveEfficiency/discovery/devices`	GET	Retrieves a list of devices.
`http://<server>/ActiveEfficiency/discovery/devices/{device id}`	GET	Retrieves the single device by the id provided in the url.
`http://<server>/ActiveEfficiency/discovery/devices`	PUT	Adds or updates a single device.

DELETE is not supported.

A device resource has the following properties:

Name	Type	Description
Id	unique identifier	A unique identifier that is assigned to the resource by the server. Needed to uniquely identify a device resource.
Identities	IList<DeviceIdentityResource>	A list of identities that the device is known by in other systems. See here to go to the device identity properties information.
Fqdn	string	The FQDN of the device. The FQDN for a device is mandatory.
DomainName	string	The netbios name of the domain (or the domain suffix for non Windows devices)
NetbiosName	string	The netbios name of the device (or host name for non Windows devices).
Type	int	0: Unknown 1: Desktop 2: Laptop 3: Server
CreatedBy	string	The name of the client creating the resource. Currently not mandatory, but will change.
CreatedTimestamp	DateTime	The time in UTC when the web service created the reource
ModifiedTimestamp	DateTime	The last time in UTC when the resource was modified.

A device identity resource has the following properties:

Name	Type	Description
Source	string	Name of the identity. Source is a required property.
Identity	string	Value of the identity. This is always represented as a string even if the external source represents it as another type (integer, GUID etc.) Identity is a required property.
AssignedUtc	DateTime	The date/time that the external system assigned this identity to the device.

Device Tags

A device tag represents a name-value pair that defines a certain attribute of the device. These attributes may extend the identity of the device or may be used in defining query based collections of devices. The following HTTP URI's and methods are supported for this resource:

Uri	Method	Description

`http://<server>/ActiveEfficiency/discovery/devices/{device id}/tags`	GET	Retrieves a list of tags for the device specified
`http://<server>/ActiveEfficiency/discovery/devices/{device id}/tags/{category}`	GET	Retrieves a list of tags for the device and category specified
`http://<server>/ActiveEfficiency/discovery/devices/{device id}/tags/{category}/{name}`	GET	Retrieves a list of tags for the device, category and name specified
`http://<server>/ActiveEfficiency/discovery/devices/{device id}/tags/{category}/{name}/{index}`	GET	Retrieves a single tag for the device, category, name and index specified
`http://<server>/ActiveEfficiency/discovery/devices/{device id}/tags`	POST	Adds a single device tag to the list of device tags, returning the created resource. nDevice tags are identified by category, name and index - they do not have an explicit id. If tag with the same category, name and index already exists, the value and the modified date get updated.

DELETE is not supported.

The device tag resource has the following public properties:

Name	Type	Description
Category	string	The category (namespace) for the tag. The category and name uniquely identify the resource. Category is mandatory. It must not be null, empty or only whitespace.
Name	string	The tag name. Name is mandatory. It must not be null, empty or only whitespace.
StringValue	string	The value represented as a string.
Type	int	Must be 0 (string), or 1 (date/time). If date/time, value must be in universal sortable format (http://msdn.microsoft.com/en-us/library/az4se3k1.aspx)
CreatedBy	string	The name of the client creating the resource. Currently not mandatory, but will change.
CreatedTimestamp	string	The UTC time when the web service created the reource
ModifiedTimestamp	string	The last UTC time when the resource was modified.

Location

These location resources are used by Nomad to identify subnets contained within the same site. These are pre-populated by some out-of-band method (e.g. PowerShell script). They are not directly consumed by Nomad (the web service determines which Content Delivery resources to return based on the contents of the location section of its database).

URI	Method	Description
`http://<server>/ActiveEfficiency/locations/`	POST	Adds a single location resource (returns its unique identifier).
`http://<server>/ActiveEfficiency/locations/{id}/`	GET	Returns a specific location resource using its unique identifier.
`http://<server>/ActiveEfficiency/locations/`	GET	Returns a listing of all location resources.
`http://<server>/ActiveEfficiency/locations/{id}`	DELETE	Removes a single location record using its unique identifier.
`http://<server>/ActiveEfficiency/locations/`	DELETE	Removes all location records.

Location resources have the following public properties:

Name	Type	Description	Auto-generated	Mandatory
Id	GUID	Unique identifier for this location.	Yes	Yes
Site	string	Name of Site that this Location belongs to. Arbitrary and used to find related Subnets.		Yes
Subnet	string	Subnet (for example, "10.0.10.0/24")		Yes

Pre-cached jobs

These content distribution job resources are used to manage pre-cached jobs and retrieve download notifications:

URI	Method	Description
`http://<server>/ActiveEfficiency/ContentDistributionJobs/`	POST	Creates, view and delete pre-cached jobs.
`http://<server>/ActiveEfficiency/ContentDownloadNotifications/{id}/`	GET	Retrieves download notifications

Content distribution job resources have the following public properties:

Name	Type	Mandatory	Description
Id	unique identifier	No	Identifier for the resource.
DeviceFqdns	Array of strings	Yes	List of device FQDNs
ContentItems	Array of content items	Yes	List of contents. Must include the ID and version of the content.
UserData	XML	No	Additional details about job.

Content download notifications have the following public properties:

Name	Type	Mandatory	Description
ContentId	string	Yes	Identifier for the content.
Version	string	Yes	Version of the content.
Hash	string	No	SHA-256 hash to verify integrity of files inside content

Coping with large amounts of data returned by the web service

ODATA is an open standard that allows clients to consume data feeds in an open format and was designed for this purpose. The ActiveEfficiency web service supports ODATA so that the clients can ask for pages of data. Assuming that the web service contains 1000 devices, the following are examples of how ODATA helps us:

Without ODATA

Without ODATA, the following url returns 1000 devices in one go; no good if the client limits the buffer size:

```
http://<server>/ActiveEfficiency/devices [GET] ;
```

returns 1000 devices in one go

With ODATA

ODATA supports the keyword **top** and **skip** which allows you to implement the following loop:

```
http://<server>/ActiveEfficiency/devices?$top=100&$skip=0;
```

returns the first 100 devices

```
http://<server>/ActiveEfficiency/devices?$top=100&$skip=100;
```

returns the next 100 devices

```
http://<server>/ActiveEfficiency/devices?$top=100&$skip=200;
```

returns the next 100 devices and so on…

Example scripts

ActiveEfficiency currently provides three example scripts, one for use with AppClarity and the others for use with Nomad:

1. Script to fetch data from Active Directory and store them in the ActiveEfficiency database as device tags via the ActiveEfficiency Web API. This script can be found here.
2. Script to fetch site and subnet data from AD and store in ActiveEfficiency database via the ActiveEfficiency Web API. This script can be found here
3. Script to add custom site and subnet data to the ActiveEfficiency database via the ActiveEfficiency Web API. This script can be found here.