Alcatel-Lucent 5620

SERVICE AWARE MANAGER | 13.0 R7

APPLICATION API DEVELOPER GUIDE

3HE-10590-AAAA-TQZZA

Issue 1 | December 2015

Contents

About this document

Purpose

The *5620 SAM Application API Developer Guide* provides information to help OSS developers create business applications which interact with 5620 SAM applications in order to perform various network management functions.

Document support

Customer documentation and product support URLs:

Customer documentation welcome page

- https://infoproducts.alcatel-lucent.com/cgi-bin/doc_welc.pl

Technical support

- http://support.alcatel-lucent.com

How to comment

Documentation feedback

- documentation.feedback@alcatel-lucent.com

Part I: Getting started

Overview

Purpose

This volume provides an overview of OSS application development and Open Interfaces Professional Services.

Contents

1 Getting started

Overview

About this guide

This guide provides information to help OSS developers create business applications which interact with 5620 SAM applications in order to perform various network management functions.

Before you begin

Alcatel-Lucent recommends that you do the following before you start your OSS application development:

- review the requirements for licenses for the application with which you plan to integrate
- understand OSS interfaces in general and knowledge of the following technologies, which are beyond the scope of this guide:
  - HTTP(S) – *HyperText Transfer Protocol (Secure)*
  - REST architecture – *Representational State Transfer*
  - JSON – *Javascript Object Notation*
  - oAuth 2.0
  - applications

Open Interfaces Professional Services

The Alcatel-Lucent OIPS (Open Interfaces Professional Services) portfolio provides OSS developers support how to integrate with our suite of applications.

OSS integration initiatives include project review, design consultation, development support, and training for integration projects.

Key elements of each service offering include:

- Design consultation and high-level project reviews. Extensive software design consultation provides architectural review of a proposed design and guidance on whether the design methodology is suited to the proposed application given Alcatel-Lucent experience with the platform and interface.
- software development support and technical guidance
- sample code that provide examples to facilitate rapid OSS integration
- OSS product training that minimizes project down time and optimizes development efficiency

Alcatel-Lucent OIPS can customize the support content and contract length to meet the unique development requirements of service providers.

Part II: Applications and OSS

Overview

Purpose

This volume describes the applications, API, OSS communications, and OSS domains.

Contents

2 Applications overview

Overview

Purpose

This chapter describes the 5620 SAM applications and the application RESTful interfaces.

Contents

Applications

Overview

The 5620 SAM provides network management functions using browser-based applications. The applications are external to the 5620 SAM client GUI and do not require a local client installation. The use of the 5620 SAM applications is intended mainly for network fault monitoring and troubleshooting.

Each application publishes a set of URLs which point to resources, or web services, managed by them. Each domain application documents the URLs that are available to users. These URLs can be accessed through a browser by any authorized user, including OSSs which can use them to cross launch from their own application.

To view a dashboard of all the available applications:

http(s)://<*host*>

Where *host* is the hostname or IP address where the 5620 SAM is installed. HTTP(S) must be used when the 5620 SAM system is secure.

To view the published URLs of a given application:

http(s)://<*host*>/<*app_name*>/api-docs

Where *host* is the hostname or IP address which hosts the application, *app_name* is the name of the application. For example, Fault Management.

For more information on the 5620 SAM applications, see the *5620 SAM User Guide* and *5620 SAM System Administrator Guide*.

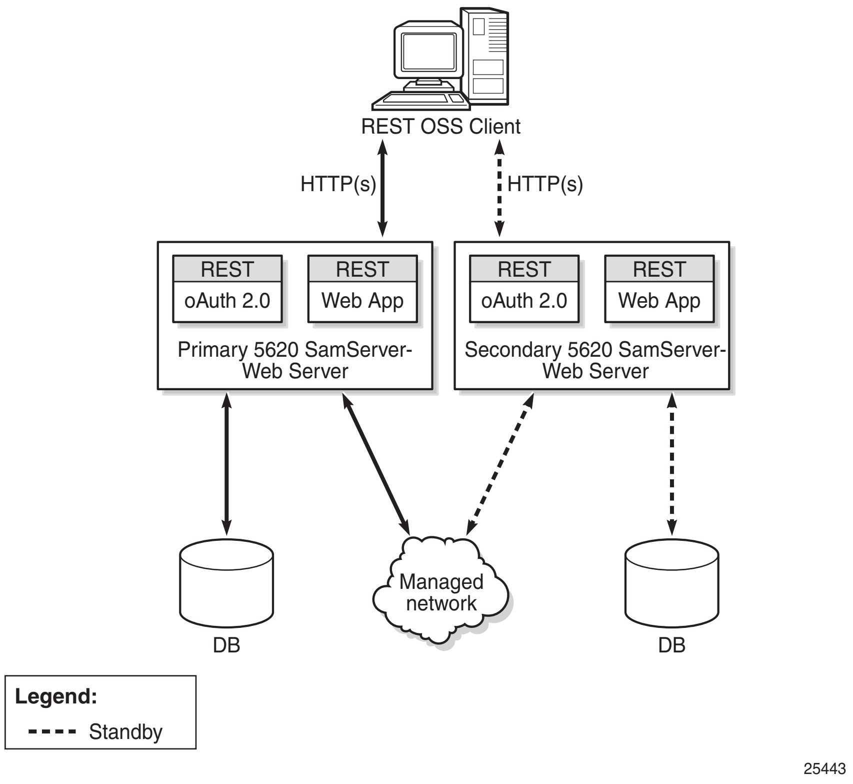
Applications API

Overview

In addition to the published URLs, each application also makes a set of REST north bound API available, when applicable, to allow OSSs to access the resources of the application.

The applications have a number of services, or resources, that OSSs can access through a REST API.

The following figure illustrates the major components during interactions between an OSS and a 5620 SAM application in general.



The REST OSS client is an application that can communicate using the HTTP(S) protocol with the 5620 SAM system. There is only one active 5620 SAM server at a time, which is the primary 5620 SAM server. The OSS client application must only communicate with the active 5620 SAM server. The OSS client sends a JSON formatted REST request using the standard HTTP(S) protocol to the appropriate application in the 5620 SAM system. The application processes the request and returns a JSON formatted response through the same HTTP protocol.

3 OSS communication with applications

Overview

Purpose

This chapter describes how an OSS communicates with 5620 SAM applications and the underlying network through the application REST interface.

Contents

OSS communication

Overview

The OSS application communicates with the applications using the HTTP(S) protocol. The requests and responses are formatted using JSON.

An OSS client must first determine the hostnames (or IP addresses) of the primary and secondary (for a redundant system) workstations which host the applications. The primary and secondary hostname (or IP addresses) are provided by the applications administrator.

The OSS client must always communicate with applications hosted on the same workstation as the active 5620 SAM server. The following steps are required to determine the primary application host in a redundant system:

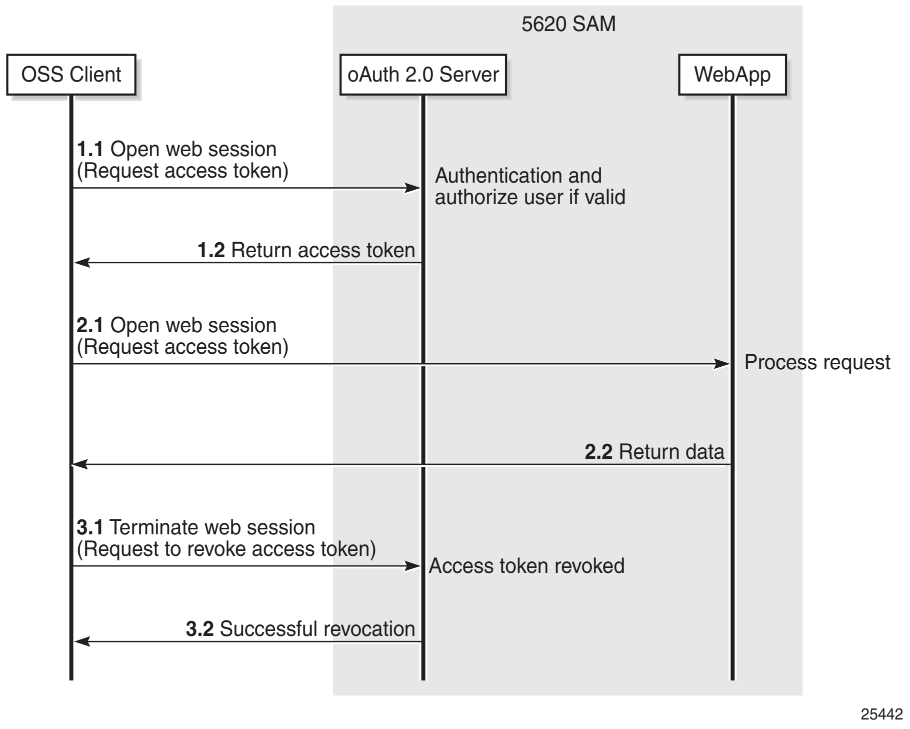
1. If the OSS application is integrated with 5620 SAM JMS, it receives a notification which indicates which is the active 5620 SAM server. For more information see the *5620 SAM XML OSS Developer Guide*. If JMS is not used, go to step 2.
2. The OSS client issues the HTTP(S) request to the primary hostname. If the request fails with error code 'HTTP/1.1 503 Service Unavailable', go to step 3.
3. The OSS client issues the HTTP(s) request to the second hostname. If the request fails again, return to step 1.

Once the primary application host is established, the OSS application sends a JSON formatted HTTP(S) request to access application resources through the RESTful API provided by the application.

These resources are protected. OSS applications require a valid user to access them:

- User security for the applications is currently administered by the 5620 SAM user security system. See "5620 SAM user security" in the *5620 SAM System Administrator Guide* for more information.
- With a 5620 SAM authenticated user, OSS applications can open a web session (using oAuth 2.0 framework) which allows them to invoke the applications RESTful APIs.

The following figure shows how an OSS communicates with an application.


5620 SAM
OSS Client
oAuth 2.0 Server
WebApp
1.1 Open web session (Request access token)
Authentication and authorize user if valid
1.2 Return access token
2.1 Open web session (Request access token)
Process request
2.2 Return data
3.1 Terminate web session (Request to revoke access token)
Access token revoked
3.2 Successful revocation
25442

Workflow to access an application resource

Overview

The following workflow describes how to access a web resource through its RESTful API. See Chapter 6, “Application REST API examples”.

Process

1 Open a web session – uses oAuth 2.0 standard mechanism

The web service owner (i.e. the 5620 SAM system administrator) provides user credentials (username and password) to OSS application (client). To obtain an access token:

1. The OSS client issues a request to obtain an access token (or a bearer token) from the authorization server using the user credentials:

   An HTTP(S) POST request is sent to http(s)://host/auth/api/v1 .

   The request header must contain “Content-Type” field with value “application/json” and “Authorization” field with value “*Basic <base 64 encoding of username:password>*”.

   The request body must contain “*grant_type*” field with value ”*client_credentials*”.
2. The authorization server authenticates the OSS client user credentials. If the credentials are valid, the authorization server issues the bearer token in the response. The OSS client application has an active session as long as the bearer token is valid.

2 Access the protected resource using your active web session.

To access an application resource:

1. Obtain the URL of the REST API.
2. Build your JSON-formatted request using the schema of the REST API; the request header must contain the field “Authorization” with value “Bearer <*access token received in workflow step 1 of the opened web session*>”.
3. Issue the HTTP(S) request (e.g. GET, POST, etc). When required, use URL double encoding/decoding to handle special characters (not allowed in URL).
4. Data is returned to be processed by the OSS application.

3 Terminate the web session.

The web session must be terminated when it is no longer needed by the OSS application because the token does not expire.

To terminate a web session:

1. Send an HTTP(S) request to terminate the web session by revoking the bearer token:

   An HTTP(S) POST request is sent to http(s)://host/auth/api/v1

   The request header must contain "*Authorization*" field with value "*Bearer <access token>*"

   The request body must contain "*revoke_type*" field with value "*token*" and "*username*" field with value set to the username of the user who obtained the token earlier
2. The access token is rendered invalid and can no longer be used.

Part III: OSS domains

Overview

Purpose

This volume contains information that a developer can use to help with the development of applications for OSS domains.

Contents

4 Network and service assurance

Overview

Purpose

This chapter contains information that a developer can use to help with the development of applications for the following OSS domains:

- Fault Management

Contents

Fault Management

Fault Management API

The Fault Management application provides a means of investigating faults in a network. A user can obtain information about faults, such as their causes and impacts. The Fault Management Application provides a list of published URLs that can be launched from OSS applications and a set of RESTful APIs designed for Fault Management OSS applications.

Information model documentation

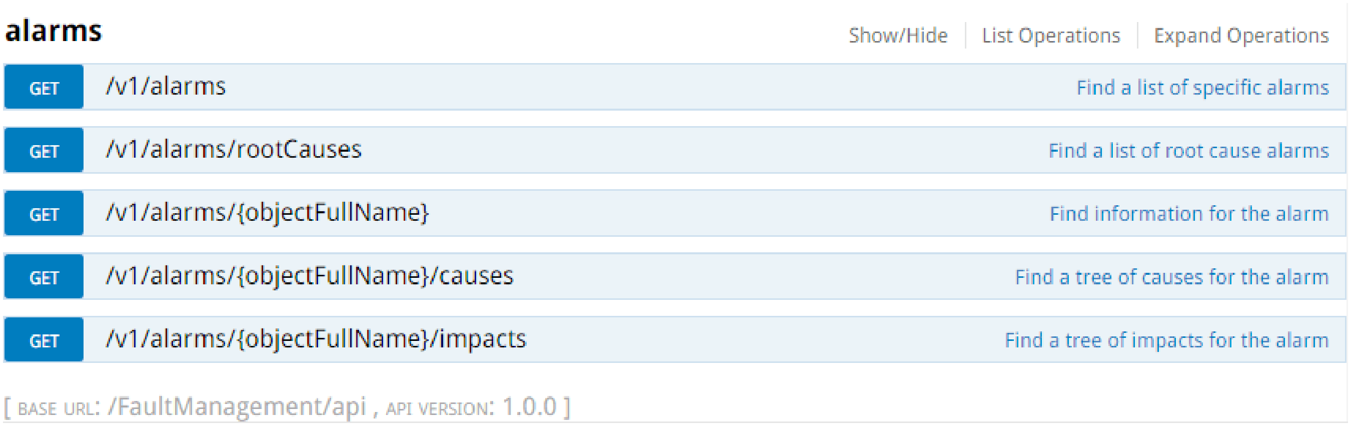
The Fault Management application provides REST interfaces to expose some of its resources, the information model of these REST APIs can be accessed as follows:

https://<*host*>/FaultManagement/api-docs

Where *host* is the hostname or IP address which hosts the application.

Fault management application example on host 135.121.156.20:

http://135.121.156.20/FaultManagement/api-docs/ .



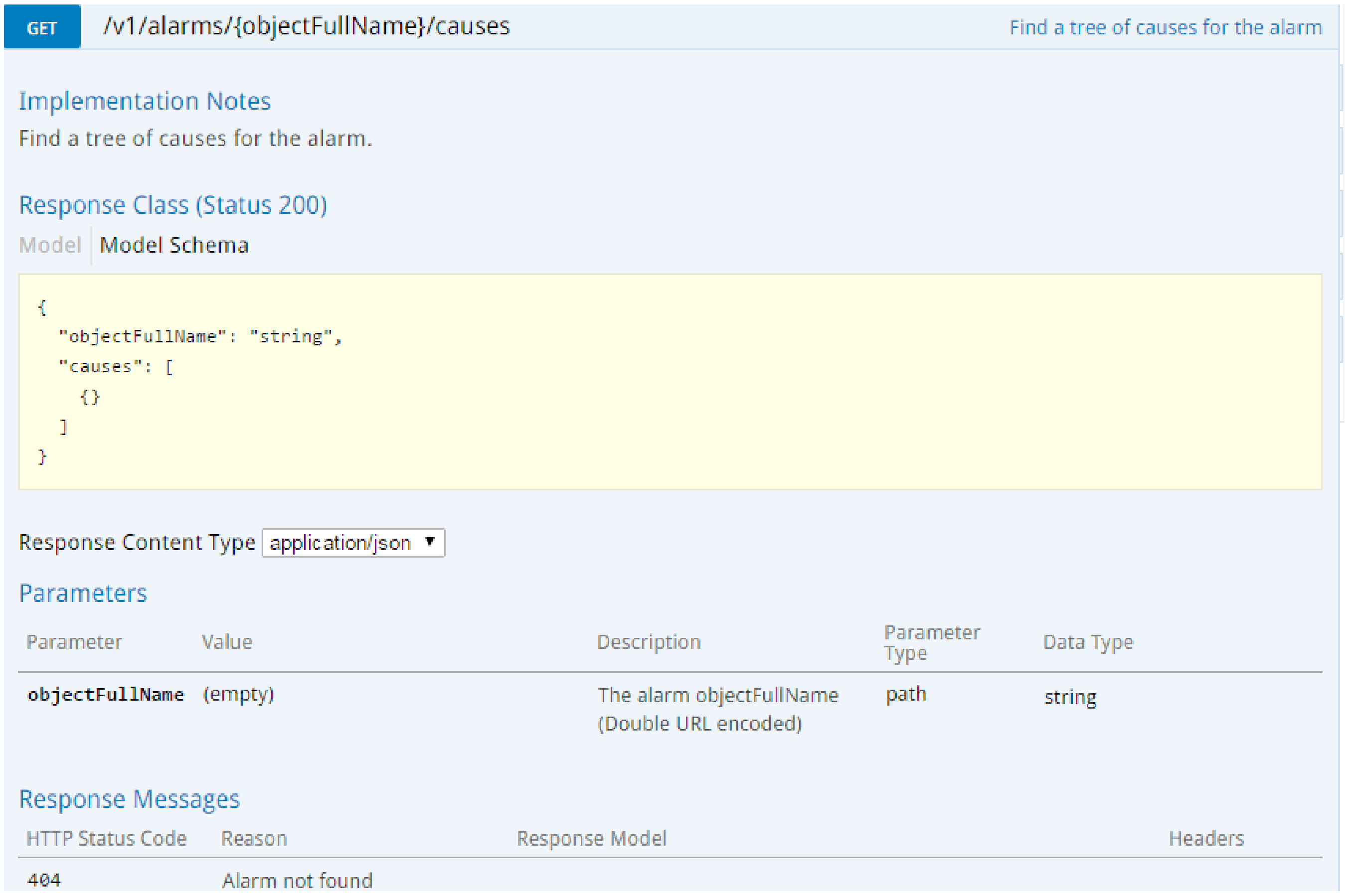
alarms

GET	/v1/alarms	Find a list of specific alarms
GET	/v1/alarms/rootCauses	Find a list of root cause alarms
GET	/v1/alarms/{objectFullName}	Find information for the alarm
GET	/v1/alarms/{objectFullName}/causes	Find a tree of causes for the alarm
GET	/v1/alarms/{objectFullName}/impacts	Find a tree of impacts for the alarm

[BASE URL: /FaultManagement/api , API VERSION: 1.0.0]

To obtain the full URL of the REST API, append the name of API to the base URL shown at the bottom. For example, the URL to retrieve the alarms is http://135.121.156.20/FaultManagement/api/v1/alarms.

To view the schema of a particular REST API, click on the interface name. For example, if you click on '/v1/alarms/{objectFullName}/causes, the following information about the specific alarm is shown:



GET /v1/alarms/{objectFullName}/causes Find a tree of causes for the alarm

Implementation Notes

Find a tree of causes for the alarm.

Response Class (Status 200)

Model | Model Schema

```
{
  "objectFullName": "string",
  "causes": [
    {}
  ]
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
objectFullName	(empty)	The alarm objectFullName (Double URL encoded)	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	Alarm not found		

OSS use case: Root Cause analysis and impact of an alarm

1. OSS application receives a critical alarm through the 5620 SAM JMS notification system. The alarm objectFullName is “faultManager:svc-mgr@service-5@35.121.20.55@circuit-21-37|alarm-221-30-166”.
2. To determine what other alarms have been caused by this alarm, the OSS issues the following request (the objectFullName of the alarm is URL encoded):

```
GET /FaultManagement/api/v1/alarms/faultManager%3Asvc-
mgr%40service-5%4035.121.20.55%40circuit-21-

37%7Calarm-221-30-166/causes HTTP/1.1

Host: 135.121.158.77

Authorization: Bearer
YWRtaW4tNDMxNmJkYTktZjQxMC00YmRiLWExYzItMTFhY2E0MjA4NGFj

Cache-Control: no-cache
```

The response is a JSON formatted tree of alarm causes:

```
{ "objectFullName": "faultManager:svc-mgr@service-
5@35.121.20.55@circuit-21-37|alarm-221-30-166",

 "causes":

  [{"objectFullName": "faultManager:serviceTunnel@from-
35.121.20.55-id-21|alarm-30-12-23",

   "causes": []

   }

   ]

 }
```

The response shows a tree of alarms caused by the current alarm.

3. To determine the impact this alarm has on the 5620 SAM network objects, the OSS issues the following request:

```
GET /FaultManagement/api/v1/alarms/ faultManager%3Asvc-
mgr%40service-5%4035.121.20.55%40circuit-21-

37%7Calarm-221-30-166/impacts HTTP/1.1

Host: 135.121.158.77

Authorization: Bearer
YWRtaW4tNDMxNmJkYTktZjQxMC00YmRiLWExYzItMTFhY2E0MjA4NGFj

Cache-Control: no-cache
```

The response is JSON formatted tree of impacted alarms:

```
{ "objectFullName": "faultManager:svc-mgr@service-
5@35.121.20.55@circuit-21-37|alarm-221-30-166", "impacts": [

 {

  "objectFullName": "faultManager:svc-mgr@service-
5@35.121.20.55|alarm-97-16-83","impacts": []

 }

 ]

}
```

The response is a tree of impacted alarms.

Fault Management cross-launch

The Fault Management application publishes a list of URLs which allows an OSS application to perform various functions on 5620 SAM alarms. The full list of public URLs can be viewed at:

http(s)://<*host*>/FaultManagement/api-docs/

Where *host* is the hostname or IP address which hosts the Fault Management application (which is typically the workstation which hosts the 5620 SAM system).

Below are examples of the published URLs:

<*host*>/FaultManagement/

<*host*>/FaultManagement/?view=<*viewName*>

Where *viewName* is one of: unhealthyNE, alarmList, neInspector, topProblems.

<*host*>/FaultManagement/?view=alarmListImpacts&objectFullName=<*alarm_objectFullName*>

Where *alarm_objectFullName* is a double URL-encoded alarm objectFullName.

Use case: cross-launch Fault Management from an OSS application

1. OSS application receives a critical alarm (vpls.site down) through the 5620 JMS notification system. The alarm objectFullName is "faultManager:svc-mgr@service-9@35.121.20.55|alarm-97-16-83"
2. OSS wants to determine the impact of this alarm by cross-launching the Fault Management application. It issues the following request (Notice that the objectFullName of the alarm is URL encoded):

   ```
   http://135.121.158.77/FaultManagement/?view=alarmListImpacts&objectFullName=faultManager%3Asvc-
   mgr%40service-9%4035.121.20.55%7Calarm-97-16-83
   ```
3. The Fault Management application is launched in a web browser and opened to the Alarm Impacts window.

   You can quickly assess the impact of the alarm including the affected objects.

Part IV: Appendices

Overview

Purpose

The following appendices contain additional reference information about the 5620 SAM application API.

Contents

5 Error handling

Overview

Purpose

This appendix describes error handling.

Contents

Error handling

HTTP(S) Error codes

The HTTP(S) protocol is used between the OSS application and the 5620 SAM applications; all standard HTTP(S) error codes must be handled accordingly.

API Specific Errors

Each REST interface documents error codes specific to the interface; see interface documentation on how to handle these errors.

6 Application REST API examples

Overview

Purpose

This appendix provides application API examples.

Contents

Examples

OSS user authorization and authentication

Example request:

POST /auth/api/v1 HTTP/1.1

Host: mairead

Authorization: Basic

eHZ6MWV2RlM0d0VFUFRHRUZQSEJvZzpMOHFxOVBaeVJnNmllS0dFS2hab2xHQzB2SldM

==

Content-Type: application/json

Content-Length: 29

Accept-Encoding: gzip

{"grant_type":"client_credentials"}

Example response:

HTTP/1.1 200 OK

Status: 200 OK

```
Content-Type: application/json; charset=utf-8

...

Content-Encoding: gzip

Content-Length: 140

{"token_type":"bearer",
"access_token":"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%2F"}
```

Access an application resource

Example request:

```
GET /FaultManagement/api/v1/alarms HTTP/1.1

Host: mairead

Authorization: Bearer
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%2F

Accept-Encoding: gzip
```

Terminate a web session

Example request:

```
POST /auth/api/v1 HTTP/1.1

Host: mairead

Authorization: Bearer
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA%2F

Content-Length: 29

Accept-Encoding: gzip

{"revoke_type":"token","username":"my_username" }
```