Cisco Prime Home Integration Guide

November, 2012

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-28557-01 v5.0.1

Cisco Prime Home 5.0.1 Integration Guide

Table of Contents

Introduction **5**
Preface 5
Intended Audience *5*
In This Guide… *5*
Prime Home API Capabilities *5*
Device Management: *5*
Search: *5*
Create Control Panel Sessions: *5*
Getting Started 5
Language / Framework Requirements *5*
Searching the API *6*
Devices and Subscribers *6*
Applications & Drivers 6
Services 7

Device Lifecycle **8**
Management *9*

API REFERENCE **10**
REST 10
Anatomy of a REST Call *10*
Kinds of Resources *12*
Single Item *12*
Lists *12*
Processes *12*
Compared with SOAP *12*
JSON *12*
API Standards 13

Search and CAQL **16**
Overview 16
About Search 16
Language Specification 17

Web Services **23**
API Catalog 23
Full Text Search *23*
Provisioning & Management 23

Troubleshooting **29**
When Working with Documents 29
Revision Conflict *29*
Invalid Data *29*
JSON Formatting *30*

Subscriber DTO Schema **30**
Subscriber 31
Subscriber.Address.{} 31
Subscriber.Phone.{} 31

Examples **32**
Example Format 32

Use Case *32*
Method *32*
HTTP Interaction *32*
Utilities Used by the Examples *33*

Interacting with Search **34**
Use Case 35
Method *35*
HTTP Interaction *35*
Code 36
SearchService.scala *36*
SearchResult.scala *36*
SearchUseCase.scala *39*

Interacting with the Device Lifecycle **40**
Use Case 40
Method 40
HTTP Interaction 40
Creating the Subscriber *40*
Finding / Creating a Device *41*
Enabling Services *43*
Updating an Application *45*
Replacing & Deleting a Device *49*
Sample Code *53*
DeviceService.scala *55*
LifeCycleUseCases.scala *57*

Creating Control Panel sessions for Single Sign On **61**
Use Case 61
Method 61
HTTP Interaction (Web Services) 61
HTTP Interaction (Browser) 62
Sample Code 62
ControlPanelSessionService.scala *62*
AuthBridgeHttpRequestHandler.scala *64*

Deprecated **66**
Changing SSID with CURL 66
Working with Subscribers in Java 67
Introduction *67*
Main Example *67*
Interaction with the API *68*
Data Classes *70*
Support *75*

Appendix A: CAQL BNF **78**

Introduction

Preface

Intended Audience

This guide is intended to be used by anyone who wants to become familiar with integrating external systems with Prime Home. Most of this document specifically targets integration implementers.

In This Guide…

The device lifecycle
Northbound APIs for searching, provisioning, control panel session management
Applications
LDAP Configuration

Prime Home API Capabilities

Device Management:

Creating, editing, and deleting subscribers, devices, and associated data
Associating subscribers with managed devices
Configuring managed devices (i.e. port bridging, wireless settings, etc.)
Turning on/off specific services (i.e. IPTV, Content Filtering, etc.)
Querying the state of device operations

Search:

Find subscribers and devices given a number of fields to search
Extract bulk data such as statistics

Create Control Panel Sessions:

For subscriber single sign-on implementations

Getting Started

Language / Framework Requirements

You are free to choose whatever language or framework you are most comfortable with since the API is entirely web based and is language agnostic. This document will provide some suggestions and examples, but you are by no means limited to those languages.

Choose an HTTP client software to interact with the API. It should support the GET, POST, PUT, and DELETE HTTP methods. cURL, wget, apache java http client,

and Restlet are good choices. Select a tool to manipulate JSON formatted data structures. A straight text editor will do. TextMate and E Text Editor (http://www.e-texteditor.com), jEdit, and notepad++ are good choices. cURL (http://curl.haxx.se/docs/manpage.html) and a text editor will be used for the examples in this guide.

Searching the API

In Prime Home 3.0, CAQL (ClearAccess Query Language) is used with the search web service to find objects to manipulate with other web service endpoints.

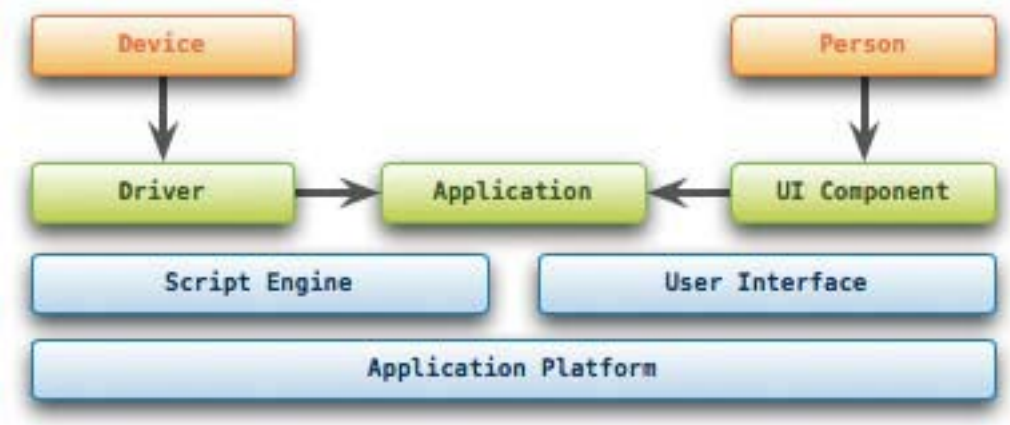
The properties returned for each item don't necessarily map directly to the search query. for example: querying 'subscriberCode' returns as 'code'.

Devices and Subscribers

Devices and subscribers are the primary objects managed by Prime Home. A device is managed by Prime Home. It identifies itself with an OUI and Serial Number. While many devices have a unique MAC address, the MAC is not used for identification purposes (though, some devices have a serial number that is the same as the MAC). It is critical to keep this in mind when integrating with Prime Home. When creating a device using the integration APIs, the OUI and Serial Number are the minimal amount of data required. A subscriber is a person (or account) who has devices. It is externally identified by a subscriber code. The format and contents of this code are determined by the integrator and may map to a phone number, billing account number, or some other unique value which has meaning in other systems.

Applications & Drivers

Figure 1



Prime Home is a platform for executing applications against devices in order to apply some configuration. An application is a cohesive set of device functionality and configuration. An example of this would be Time Blocking or Wireless. Applications some or all of:

- Name & ID

- Data schemas global to all devices in a system. (Typically called "Globals")
- Data schemas local to each individual device. (Typically called "Settings")
- Scripting engine APIs.

These applications are synchronized with a device during sessions with the CWMP server when they are marked as needing synchronization with that device. When this occurs, a translation layer called a "Driver" is used to implement the configuration of the application on the device. This translation layer is required because many devices need different kinds of configuration, or may not support the application at all.

More information about applications is available later in this guide.

Services

Services are the primary configuration element of Prime Home. They are used to implement business cases as logical groupings of user interfaces and device manageable configurations which can be enabled or disabled on devices. One example of this would be "Managed WiFi" where customer support personnel can enable or disable this functionality easily. Another example would be grouping time blocking and content filtering into a "parental controls" service, thus aggregating multiple sets of configuration into a single item.

Device Lifecycle

The device lifecycle consists of 4 phases: provisioning, activation, management, and termination.

Provisioning

This is the process of configuring a device for use by a subscriber. A device in this phase can be either a "future device" or a "managed device". A future device is a placeholder which contains configuration for a device which will by managed by the ACS at some point in the future.

During the provisioning phase:

- Subscribers and devices are associated with each other
- Initial service subscriptions are configured

Activation

Activation occurs the next time a device contacts the ACS after the provisioning phase.

This can come in multiple forms:

- The device has never contacted Prime Home, but a future device has been configured ahead of time.
- The device has previously contacted the ACS, but becomes associated with a new subscriber.

During this phase, services which are provisioned in an "enabled" state are enabled and initial settings are applied. Initial settings can include:

- PPP credentials

- LAN addresses
- Passwords
- Management server configuration
- Etc

Management

This is the day-to-day life of a managed device where:

- CSRs interact with applications in order to troubleshoot problems and self modify services
- Subscribers can change the configuration of their device by using the Control Panel
- The system can collect statistics and run bulk operations, such as firmware upgrades

Termination

Termination occurs when a device is no longer being used by its former subscriber. This can happen due to a number of conditions, such as: This phase happens for as long as the subscriber is using the managed device.

- Customer churn
- Device failure
- Device upgrade

Prime Home can handle this situation in a number of ways:

- The managed device can remain active in the system, but un-associated with the subscriber
- The managed device can simply be deleted from the system

A new device can replace an old device as an RMA.

API REFERENCE

REST

REST (REpresentational State Transfer) is a paradigm for interacting with a web service via the manipulation of data resources. In this paradigm, a URI represents a unique resource which can be manipulated using HTTP methods such as GET and POST. The state of the resource is transferred between the client and server as a representation of that resource which can be manipulated.

Anatomy of a REST Call

Request

```
POST /person/bjones HTTP/1.1
Host: people.mydomain.com
Accept: application/json, text/xml;q=9, */*;q=8
Content-Type: application/json

{
name: "Bob Jones",
age: 30
}
```

This request says: "Create a new resource at /person/bjones on people.mydomain.com using the provided JSON representation, and send the result back as JSON (or possibly XML, or maybe something else entirely)

An HTTP request is ALWAYS in the form:

```
[VERB] [resource uri] [http version]
[headers]
[\n]
[body]
```

This line indicates 3 things: the method, the URI of the resource, and the HTTP version. These things always appear like this as the first line of any HTTP request and are always followed by request headers.

Headers

`Host: people.mydomain.com`

This header is always required when using HTTP version 1.1 and indicates which host or virtual host to request the resource from.

`Accept: application/json, text/xml;q=9, /;q=8`

This header informs the server how the client would like the representation of the resource to be formatted. It indicates both which formats the client accepts and their order of preferences. In this example, the client primary wants JSON formatted data. If that is not available, then XML would be alright too. If neither of those are available, just send the data in any format.

See http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html for more details.

`Content-Type: application/json`
This header informs the server that the body of the request is formatted as JSON and should be interpreted as such.

Body

The headers and body are separated by a single empty line. A representation of the resource being sent to the server is contained within the body. Its format is dictated by the value of the `Content-Type` header.

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 34
{
name: "Bob Jones",
age: 30
}
```

The response says: "Your request was successful and here is your result in JSON format." An HTTP response is ALWAYS in the form:

```
[http version] [status number] [status text]
[headers]
[\n]
[body]
```

HTTP/1.1 200 OK

This line indicates that the response uses the HTTP 1.1 protocol, and that the request was successful.

Headers

`Content-Type: application/json`
The body of the response contains a JSON formatted representation of resource.

`Content-Length: 34`
There are 34 bytes in the body

Body

The headers and body are separated by a single empty line. A representation of the resource being sent from the server is contained within the body. Its format is dictated by the value of the Content-Type header.

Methods

Method	Purpose
GET	retrieves a representation of the resource
DELETE	removes a resource at the URI
POST	update a resource by appending to it, or by providing partial state which is filled in by an internal process
PUT	create or replace a resource at the URI

References

- http://en.wikipedia.org/wiki/Representational_State_Transfer

- http://stackoverflow.com/questions/630453/put-vs-post-in-rest
- http://www.aminus.org/blogs/index.php/2005/07/05/i_don_t_get_put_versus_post?blog=2
- http://jcalcote.wordpress.com/2008/10/16/put-or-post-the-rest-of-the-story/
- http://www.markbaker.ca/2002/08/HowContainersWork/
- Richardson, Ruby RESTful Web Services. O'Reilly 2007

Kinds of Resources

A resource generally comes in one of 3 flavors: a single item, a list of items, or a process. Depending on the kind of resource, the semantics of the various supported methods can be slightly different.

Single Item

Interacting with a single resource is the most common. In this case a single document is sent to and from the server. Typically, a resource can be retrieved, modified and sent back to the server to update the resource's state.

Lists

A list resource can act as a table of contents for sub-ordinate resources which can filtered, searched, sorted and paginated. New sub-ordinate resources can be created by appending (via POST) to a list resource.

Processes

Resources do not always have to be objects or state-centric. System processes can also be exposed as a resource and are typically interacted with using the POST method.

Compared with SOAP

- REST tends to focus on data, SOAP tends to focus on operations.
- SOAP is a much more strictly defined standard, REST is far less formal.
- REST services tend to be easier to access because any HTTP tool can interact with them. SOAP requires a richer toolset to make interaction easy.
- SOAP is designed to be transport-layer independent. REST is HTTP.

JSON

http://json.org/

JavaScript Object Notation is the data format primarily used by ClerVision's APIs. The format is easy to produce and consume and is largely schema-less. The Content-Type to use when sending or receiving JSON data is 'application/json'.

API Standards

All APIs found under the /api URI, will follow these standards.

URIs

- The root of all APIs is http://host/prime-home/api/
- Resources are versioned. the version immediately follows .../api. e.g.: .../api/v1
- Versions are determined by the structure of the representation
- If the resource name is plural, it should be treated as a list / table of contents. e.g.: .../api/v1/subscribers

Tables of Contents

A Table of Contents is used to find some sub-ordinate resource to work with. You can also create new items by posting to a Table of Contents.

GET

- request uri query ?filter=<string>; Provide this parameter as a search query to refine your results.
- request uri query ?first=<number>; Which item in the results to return first, beginning with 1.
- request uri query ?count=<number>; How many items from the results to return.

✎ The maximum count handled is 1000. If you request more, then the results will be limited to this maximum.

- response header Pagination-First: <number>; The actual first item returned.
- response header Pagination-Count: <number>; The actual number of items returned.
- response header Pagination-Total: <number>; The total number of items in the entire result set.

POST to append

DELETE not supported

Documents

Document services will behave as follows:

- GET will return the full representation of the resource
- DELETE will remove the resource from the system
- PUT accepts a new representation
- When a document is sent from the server to the client, it includes a 'revision' property. This value must be returned to the server when making changes. If the revision on the server is different at the time of the update, an error status will be returned to the user, and the state of the resource will be unchanged.

- Data retrieved from a GET can be sent directly with a PUT with no changes.

Most documents have some common fields

Field	Purpose
code	An immutable external identifier, used for integrations
Id	Internal identifier (typically a sequential number), used by Prime Home
revision	A value used for concurrency control of the document. it is required for updates to the document

Modifying a Document

To make changes to a document follow this procedure:

1. GET the resource
2. Make all of the necessary changes to that document within your application
3. PUT the document back to the same resource
4. If you are interested in the status of asynchronous side effects:
    a. The PUT response will contain headers which will indicate a URL to poll which will indicate status, if there were any asynchronous side-effects.
    b. Poll the provided status URI until it indicates complete.

Creating a New Document

To create a new document, such as a device, follow this procedure:

1. GET the template of the type of document you wish to create. ex: `http://host/prime-home/api/v1/templates/device`
2. Make all of the necessary changes to that template within your application to set the desired data
3. POST the document to the table of contents of that document type. ex: `http://host/prime-home/api/v1/devices`

Deleting a Document

1. DELETE the url of the document you wish to remove. ex: `http://host/prime-home/api/v1/subscribers/code:123456789`

WADL

Appending `?wadl` to any resource uri will return a generated HTML description of that resource including the supported methods, data types, references to sub-ordinate resources, and security parameters.

Status Codes

For detailed information: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html 400 (Bad Request) statuses are broken up into status codes for easy consumption. 400 statuses will be returned in the following format:

```
400 ERROR-CODE; Error Text
```

Status	Code	Meaning	Caused By
200		OK	The request was handled successfully
401		Unauthorized	Credentials have not been provided by the caller
403		Forbidden	Credentials were provided but the resource rejected access
404		Not Found	The resource at the URI does not exist
400		Bad Request	Caused by the client providing bad information to the server.
	SYNTAX-ERROR		The structure of the document provided is invalid
	VALIDATION-ERROR		Some value provided in the document is invalid
	CONCURRENCY-ERROR		Provided an out-of-date revision
	REFERENCED-ENTITY-NOT-FOUND		Providing data which references something that is invalid. e.g.: a service that does not exist
500	Internal Server Error		Internal error occurring on the server. Contact Cisco Support.

Dates

Dates conform to ISO8061 compliant format. Times are formatted in UTC time with descrete time zone notation.

Search and CAQL

Overview

ClearAccess Query Language (CAQL) is a domain-specific language (DSL) for specifying search queries. Search queries are executed against the Prime Home ACS to list subscribers, devices, and other ACS data that match the query criteria.

CAQL is used to support features throughout the product:

- Searching for devices, subscribers, and other data within the Portal web application
- Selecting devices for bulk operations
- Subscriber and device reports, including CSV export

> For subscriber and device reports the user does not have to enter CAQL directly. A user interface is provided that allows the user to create a report using drag-and-drop and by entering simple criteria.

About Search

CAQL is a language for specifying queries against a search engine. In order to understand how to write CAQL, it is useful to understand the basics of how the search engine works.

Data in the search engine is organized into documents. In Prime Home there are several document types, including "subscriber", "device", "user", and others. Each "subscriber" document in the search engine represents one subscriber in the Prime Home database. Each "device" document represents a device. Etc.

Within each document there are fields. The "subscriber" document contains fields which defined the various properties for the subscriber, for example "name", and "address". The "device" document contains fields which define the various properties, characteristics, and data collected from the device, for example "serialNumber", "informCount", and "downstreamAttenuation".

A complete list of available documents and fields is documented elsewhere. Within each field there are terms. A term is a simply a single word. For example, the subscriber John Doe has a "name" field containing the terms "john" and "doe". CAQL searches can be constructed to be as precise as desired.

For instance:

- Search for any documents with any field containing the term "john"
- Search for any documents with a "name" field containing the term "john"
- Search for "subscriber" documents with a "name" field containing the term "john"

Language Specification

Case Sensitivity

Keywords and field names are case sensitive. For instance, the "subscriber" document is all lowercase. The downstream attenuation field of the "device" document must be exactly "downstreamAttenuation".

Values are case insensitive. For instance, searching for the name "Jane", all the following are equivalent: "Jane", "JANE", "jane".

CAQL Tutorial

Terms

In its simplest form, a CAQL query can consist of a single term. This query searches for any documents that contain "juniper" in any field:

juniper

Often a search term should be restricted so that it only matches a certain field. The following query demonstrates searching for any documents with field "name" that contains "juniper":

```
name: juniper
```

Many fields are aliased so that they can be referred to using alternate names. For instance, the "subscriber" document has a "subscriberCode" field which may be referenced as "code". These aliases are provided for convenience and readability of CAQL. A complete list of fields and aliases is outside the scope of this document.

Specifying Document Type

By default, search terms are matched against all document types. Document types include "subscriber", "device", "user", etc. The search can be restricted to match only a specified document type using the "with" clause, which precedes the search terms. The following query searches for subscribers with any field containing "juniper".

```
subscribers with juniper
```

Note that aliases are available for most document types. For example, the "subscriber" document type can also be referenced as "subscribers", and the "device" document type can be referenced as "devices". A complete list of document types and aliases can be viewed in the Prime Home application. Click on the Utilities Menu and select Search from the left sidebar.

The special document types "anything" and "everything" can also be used in order to write CAQL that explicitly searches all document types. For example, the following searches all documents for any field containing "juniper":

```
anything with juniper
```

Use of the "anything" and "everything" document types is equivalent to the same CAQL without the "with" clause; they are provided as a means to express more readable CAQL.

Wildcards

The following searches for devices with a serial number starting with "ABC":

```
devices
with sn:ABC*
```

Single-character wildcards are also supporting using "?". The following query searches for devices with a serial number starting with "ABC" followed by exactly four characters:

```
devices with sn:ABC????
```

Wildcards can also be used inside search terms. The following query searches for devices with a serial number starting with "ABC" and ending with "XYZ":

```
devices with sn:ABC*XYZ
```

Leading wildcard usage is also supported. For instance, the following query finds devices with serial numbers ending in "XYZ":

```
devices with sn:*XYZ
```

IP addresses also support wildcards. For example:

```
devices with wanip: 192.1.*.*
```

Note that inner and leading wildcard usage does not make optimal use of the search engine and requires more system resources than trailing wildcard usage. As a result these queries can fail if the wildcard matches too many terms. It is recommended that inner and leading wildcards be used for manually entered searches only (for instance, on the Find Devices page of the Portal web application), and they not be used in automated and reoccurring processes (for instance, bulk operations).

Combining Terms

Multiple terms can be combined in a single search. By default, terms are combined using AND logic. This query searches for devices with serial numbers starting with "1000" and OUIs starting with "10":

```
devices with sn:1000* oui:10*
```

In the above example, use of the "and" keyword is implied. The following query is semantically identical to the previous query (the only difference being that the "and" keyword is explicitly used):

```
devices with sn:1000* and oui:10*
```

Terms can be combined using OR logic using the "or" keyword. This query searches for devices with serial numbers starting with "1000" or "2000":

```
devices with sn:1000* or sn:2000*
```

In Clause

A single term can be checked for one of multiple choices using the "in" keyword. This query searches for subscribers with names "sam" or "juni":

```
subscribers with name in ( sam juni )
```

Note that this is identical to combining terms using the "or" keyword. The above query is identical to:

```
subscribers with name: sam or name: juni
```

Phrases

To search for a specific phrase, enclose the words in quotes. This query searches for the subscriber named "juniper jones":

```
subscribers with name: "juniper jones"
```

Ranged Searches

CAQL supports ranged searches. For instance, the following searches for devices with an inform count between 1 and 100:

```
devices with informCount: from 1 to 100
```

Open-ended ranges are also supported. For instance, the following searches for devices with an inform count of at least 100:

```
devices with informCount: from 100
```

The following searches for devices with an inform count no greater than 10:

```
devices with informCount: to 10
```

Precedence and Grouping

The "and" keyword binds more tightly than the "or" keyword. Consider the following query:

```
serialNumber:1000* and sam or juni
```

This will return documents that:

- Have a serial number that starts with "1000" and any field containing "sam"
- **or**, have any field containing "juni"

In other words, documents with fields containing "juni" will be returned even if they do not have a serial number that starts with 1000.

To alter the binding, parenthesis must be use:

```
serialNumber:1000* and (sam or juni)
```

This will return documents that:

- Have a serial number that starts with with "1000"
- **and**, contain any fields with "sam" or "juni"

Not

To invert a search term, the "not" keyword is used. The following query will return documents that do not contain "sam" in any field:

```
not sam
```

To restrict the query to return subscribers that do not contain "sam" in the "name" field:

```
subscribers with name: not sam
```

The "not" keyword can be used with the "in" keyword as well. The following query will return any subscribers that do not contain names "sam" or "juni":

```
subscribers with name: not in (sam juni)
```

Field Types

CAQL understands several types of data used in the ACS:

- Text
- Number
- Date
- IP address

CAQL interprets the data type of each term by the context of its usage, and then its format. Single-term queries consisting of a value to be matched against any document containing that term will only be interpreted as text or IP addresses. Queries restricted to search a specific field will have those values parsed according to their format as either text, numeric, date, or IP values. Data can be "quoted" to force parsing as a string for matching against numbers in text fields, e.g. serial numbers.

The format of each data type is explained in this section. Numbers are expressed with 1 or more decimal characters ("0" - "9").

Numbers can be expressed with or without decimal points. Use of numbers without decimal points has already been demonstrated. The following query

demonstrates a search for devices with a downstreamAttenuation between 1.0 and 1.5:

```
devices with downstreamAttenuation: from 1.0 to 1.5
```

Dates are expressed in the format "mm/dd/yyyy". The following query searches for devices with a first inform date later than February 14th, 2010:

```
devices with firstInform: from 2/14/2010
```

IP addresses are expressed in the format "xxx.xxx.xxx.xxx". Each "xxx" represents a single octet of the IP address. Each octet can be a number in the range 0-255, or a "*" to match any value. The following query searches for devices with a WAN IP of 1.2.3.0-255:

```
devices with wanip: 1.2.3.*
```

Any terms that do not match the syntax for numbers, dates, or IP addresses are assumed to be text. If a text field contains a value that appears to be a non-text type, then CAQL must be instructed to treat search terms as text.

For example, if a serial number has the value "12345" then the following search **will not** work as expected:

```
device with serialNumber: 12345
```

CAQL will treat "12345" as a number and will not match the text value of the device's serial number. To override this behavior the value must be enclosed in quotes:

```
device with serialNumber: "12345"
```

Showing Fields

CAQL can include an optional "show" clause to request a specific list of fields to include in the search output. Without a show clause, the query result will contain a default set of fields. The following returns the serial number and last inform date for every device with OUI AA1122:

```
devices with oui: AA1122 show serialNumber lastInform
```

The output fields can be aliased with using the "as" keyword, as in the following example:

```
devices with oui: AA1122 show serialNumber as "Serial Number"
lastInform as "Last Inform Date"
```

The "as" keyword is provided in order to generate more readable output from the CAQL query.

Sorting

Search results can be sorted using the optional "sort" clause. The following query finds devices with OUI AA1122, sorted by the last inform date:

```
devices with oui: AA1122 sort lastInform
```

By default, sort is in ascending order. The sort order can be made explicit using the the "asc" and "desc" modifiers. The following query finds devices with OUI AA1122, sorted by the last inform date, with the most recent informs at the top:

```
devices with oui: AA1122 sort lastInform desc
```

Multiple sort fields may be specified. The following query finds devices with OUI AA1122, sorted by the last inform date, with the most recent informs at the top, with any devices informing on the same date sorted by serial number:

```
devices with oui: AA1122 sort lastInform desc serialNumber
```

Zero-Term Queries

To match all documents of a given type, simply specify the document type. The following query finds all devices:

```
devices
```

The following query finds all devices, showing the serial number, sorting by the last inform date:

```
devices show serialNumber sort lastInform
```

Web Services

API Catalog

Unless otherwise specified, all web service endpoints use JSON as their transfer encoding.

Full Text Search

Endpoint: `.../prime-home/portal/query/execute`

Method	In	Out
POST	A CAQL String	A JSON array of search results

Endpoint: `.../prime-home/portal/query/execute/name.extension`

Method	In	Out
GET	`name:` The filename of the results, extension: the results file format. Supports ".csv" and ".json"	A file containing search results in extension format.

See the CAQL documentation in the previous chapter for more information.

Provisioning & Management

Devices

Data Types : `device`

```
element device {
    attribute deviceId {number}?
    attribute subscriberCode {string}?
    (
        (attribute oui {string}, attribute sn {string}) |
        (attribute provisioningCode {string}) |
        (attribute cpProvisioningId {int})
    )
    attribute manufacturer {string}? [out]
    attribute friendlyManufacturerName {string}? [out]
    attribute deviceTypeId {number}? [out]
    attribute friendlyName {string}? [out]
    attribute softwareVersion {string}? [out]
    attribute disposition {"FUTURE_DEVICE"|"MANAGED_DEVICE"}?
[out]
    attribute excludeFromBulkOperation {boolean}?
    attribute lastInform {date}?
    attribute firstInform {date}?
    attribute informCount {number}?
    element labels {(
        attribute name {string}
        attribute fgcolor {string} [out]
        attribute bgcolor {string} [out]
    )*}
    element applications {map({string}, {(
```

```
        element dto {attribute-tree}
    )})}
    element actionLog {(
        attribute queuedRunId {number}?
        attribute name {string}
        attribute actorName {string}
        attribute runOn {date}?
        attribute state {}
        element log {(
            attribute timestamp {date}
            attribute message {string}
            attribute status {
                "NEXT_SESSION"|"SCHEDULED"|"RUNNING"|"COMPLETED"|
                "INTERNAL_ERROR"|"SYNTAX_OR_EVAL_ERROR"|"ABORT"|

"NO_RESPONSE_FROM_DEVICE"|"DEVICE_FAULT"|"SUBSCRIBER_FAULT"|
                "DOWNLOAD_FAILURE"|CANCEL
            }
        )*}?
    )*}
    element queuedActions {(
        attribute revision {number}?
        attribute solicit {boolean}?
        element scripts {(
            attribute queuedRunId {number}? [out]
            attribute scriptCode {string}
            attribute status {
                "NEXT_SESSION"|"SCHEDULED"|"RUNNING"|"COMPLETED"|
                "INTERNAL_ERROR"|"SYNTAX_OR_EVAL_ERROR"|"ABORT"|

"NO_RESPONSE_FROM_DEVICE"|"DEVICE_FAULT"|"SUBSCRIBER_FAULT"|
                "DOWNLOAD_FAILURE"|"CANCEL"
            }?
            element parameters {(
                attribute name {string}
                attribute value {string}
            )*}
        )*}
        element applications {map({string}, {(
            attribute pendingSync {boolean}
            attribute dataOwner {"DEVICE"|"SERVER"}
            attribute needsInitialization {boolean}?
        )})}
        element services {map({string}, {(
            attribute canToggleStatus {boolean}
            attribute status {"ENABLED"|"DISABLED"}
        )})}
    )*}
}
```

Data Types : device-data

```
element device-data {
    attribute revision {number}?
    attribute deviceId {number}?
    attribute subscriberCode {string}?
    (
        (attribute oui {string}, attribute sn {string}) |
        (attribute provisioningCode {string}) |
        (attribute cpProvisioningId {int})
    )
    attribute manufacturer {string}? [out]
    attribute friendlyManufacturerName {string}? [out]
    attribute deviceTypeId {number}? [out]
    attribute deviceTypeImagePath {string}? [out]
```

```
    attribute friendlyName {string}? [out]
    attribute softwareVersion {string}? [out]
    attribute disposition {"FUTURE_DEVICE"|"MANAGED_DEVICE"}?
[out]
    attribute excludeFromBulkOperation {boolean}?
    attribute lastInform {date}?
    attribute firstInform {date}?
    attribute informCount {number}?
    element labels {(
        attribute name {string}
        attribute fgcolor {string} [out]
        attribute bgcolor {string} [out]
    )*}
    element applications {map({string}, {(
        element dto {attribute-tree}
    )})}
    element rmaFor {(
        attribute code {string}?
        attribute oui {string}
        attribute sn {string}
    )?}
}
```

Data Types : device-history

```
element device-history {
    element actionLog {(
        attribute queuedRunId {number}?
        attribute name {string}
        attribute actorName {string}
        attribute runOn {date}?
        attribute state {}
        element log {(
            attribute timestamp {date}
            attribute message {string}
            attribute status {
                "NEXT_SESSION"|"SCHEDULED"|"RUNNING"|"COMPLETED"|
                "INTERNAL_ERROR"|"SYNTAX_OR_EVAL_ERROR"|"ABORT"|
                "NO_RESPONSE_FROM_DEVICE"|"DEVICE_FAULT"|"SUBSCRI
BER_FAULT"|
                "DOWNLOAD_FAILURE"|CANCEL
            }
        )*}?
    )*}
    element applications {(
        attribute state {string}
        attribute messages {string*}
        attribute lastSyncTime {date}?
    )*}
}
```

Endpoint: … /prime-home/api/v1/templates/device

A template device, used in conjunction with ...prime-home/api/v1/devices

Method	In	Out
GET	-	(device)

Endpoint: … `/prime-home/api/v1/devices`
The TOC for devices, used to create new future devices.
See also: `.../prime-home/api/v1/templates/devices`.

The minimum required fields for creating a new future device are:

- `oui` and `sn`
- OR: `provisioningCode`
- OR: `cpProvisioningId`

Method	In	Out
POST	(device)	(device)

Note: For the following four endpoints which include `deviceId; deviceId` can be represented as:

a. A numberic ID (as seen in the URL of the Customer Support Page)
b. <oui>:<serial number>, for example: `/devices/001A2B:001A2B987654`
c. sn:<serial number>, if you are sure your serial numbers are all unique.
e.g. `/devices/sn:001A2B987654`

Endpoint: … `/prime-home/api/v1/devices/deviceId`
Represents a single device. This particular endpoint cannot be modified.

Method	In	Out
GET	-	(device)
DELETE	-	-

Endpoint: … `/prime-home/api/v1/devices/deviceId/data`
This document contains information about the device itself and the configuration of each application on that device.

Method	In	Out
GET	-	{device-data}
PUT	{device-data}	{device-data}

Endpoint: … `/prime-home/api/v1/devices/deviceId/actions`
This document contains the state of actions to be performed on the device, such as synchronizing applications or enabling services.

Method	In	Out
GET	-	{device-action}
PUT	{device-action}	{device-action}

Endpoint: … `/prime-home/api/v1/devices/deviceId/history`
This document contains recent history of scripts executed on the device and the last status of application synchronizations.

Method	In	Out
GET	-	device-history

Subscribers

Data Types : `subscriber-v2`

```
element subscriber-v2 {
    attribute subscriberId {long}?,
    code {string}
    element credentials {
        attribute login {string}
        attribute password {string} [in]
        attribute expires {long} [out]
    }
    element labels {(
        attribute name {string}
        attribute fgcolor {string} [out]
        attribute bgcolor {string} [out]
    )*},
    dto: {
        element dto {tree({string}, {string})}
    },
    element subscriptions {(
        attribute deviceId {long}
        attribute oui {string}? [out]
        attribute sn {string}? [out]
        attribute provisioningCode {string}? [out]
        attribute cpProvisioningId {string}? [out]
        attribute friendlyManufacturerName {string}? [out]
        attribute friendlyName {string}? [out]
        attribute disposition {"FUTURE_DEVICE"|"MANAGED_DEVICE"}? [out]
    )*}
}
```

Endpoint: … `/prime-home/api/v1/subscribers`

> **Warning:** This API has been deprecated. See .../prime-home/api/v2/subscribers and .../prime-home/portal/query/execute

Endpoint: … `/prime-home/api/v1/subscribers/subscriberId`
This document contains information about a single subscriber and is used to modify that subscriber's data. subscriberId can be in the form of the internal subscriber numeric id, or "code:subscriber code"

Method	In	Out
DELETE	-	-

> **Warning:** This API has been deprecated. See .../prime-home/api/v2/subscribers and .../prime-home/portal/query/execute

Endpoint: … `/prime-home/api/v2/templates/subscribers`
A template subscriber, used in conjunction with `.../prime-home/api/v2/subscribers`

Method	In	Out
GET	-	{subscriber-v2}

Endpoint: … `/prime-home/api/v2/subscribers`
The TOC used for creating new subscribers. See also `.../prime-home/api/v2/templates/subscriber`.

The minimum fields which must be set are:

- `subscriberCode`: The unique, external, identifier for that subscriber.

If you do not want the subscriber to have access to the control panel, then set credentials to an empty object.

```
subscriber.credentials = {};
```

Method	In	Out
POST	{subscriber-v2}	{subscriber-v2}

Endpoint: … `/prime-home/api/v2/subscribers/subscriberId`
This document contains information about a single subscriber and is used to modify that subscriber's data. `subscriberId` can be in the form of the internal subscriber numeric id, or "`code:subscriber code`"

Method	In	Out
GET	-	{subscriber-v2}
PUT		{subscriber-v2}

In order to delete a subscriber, you must call DELETE on `.../prime-home/api/v1/subscribers/`*`subscriberId`*

Troubleshooting

When Working with Documents

Revision Conflict

Problem: A 400 status caused by updating an out-of-date revision.

Example:

Response Status

```
HTTP/1.1 400 revision is out of date; provided=86183852, existing=86183853
```

Solution: Re-fetch the data, re-apply modifications to the fresh data, and re-submit.

Invalid Data

Problem: A 400 status caused by an invalid data type

Example:

String instead of an integer

```
{
"id":"1234"
}
```

Solution: Refer to the WADL documentation for the correct data types.

Problem: A 400 status caused by an illegal reference

Example:

Invalid service name

```
{
  "service": {
  "name":"a service name that does not exist", "status":"ENABLED"
  }
}
```

Solution: Ensure service names, labels and any other names which refer to something else are correct.

Problem: A 400 status caused by a data validation error.

Examples:

WIFI Key is Too Short for WPA

```
{
  "settings": {
```

```
    "Settings.WLANConfiguration.1.BeaconType":"WPA",
    "Settings.WLANConfiguration.1.Secret":"abcd",
    }
}
```

Supplying a Provisioning Code with a OUI/SN

```
{
   "deviceRef": {
   "sn":"01234567890ab", "oui":"012345",
   "provisioningCode":"pcode-test-1
   }
}
```

Solution: Check the HTTP Status Message following the 400 code for more detailed information regarding the exact validation error.

JSON Formatting

Problem: A 400 status caused by a syntax error in JSON formatted data.
Examples:

Un-closed braces

```
{
   a: 1
```

Un-closed string

```
{
   "a: 1
}
```

Trailing comma

```
{
   "a": 1,
}
```

Solution: Fix your JSON syntax. http://www.jsonlint.com/ can provide help.

Subscriber DTO Schema

Subscriber

Property	Type	R / W	Description
FullName	string	RW	
EmailAddress	string	RW	
AddressNumberOfEntries	unsignedInt	R	
PhoneNumberOfEntries	unsignedInt	R	

Subscriber.Address.{}

Property	Type	R / W	Description
Type	string	RW	
Street	string	RW	
City	string	RW	
State	string	RW	
PostalCode	string	RW	

Subscriber.Phone.{}

Property	Type	R / W	Description
Type	string	RW	
Number	string	RW	

Examples

The code in the following examples (except for the deprecated examples):

- Is written in scala
- Uses Spring-3.0 for wiring and configuration

Example Format

Each of the following examples are in the following format:

Use Case

The description of the use case being illustrated.

Method

Step by step description of the process.

HTTP Interaction

Request

```
METHOD /resource
{
"key":"value"
}
```

Response

```
{
"key":"changed value"
}
```

Sample Code

```
"SomeFile"
/**
  * Source code implementing some part of the use case.
  */
```

Utilities Used by the Examples

HttpConfiguration.scala

```
package com.clearaccess.example
import org.springframework.beans.factory.annotation.Value
import org.springframework.context.annotation.{ImportResource,
Configuration}

@Configuration
@ImportResource(Array("classpath:spring/httpConfiguration.xml"))
class HttpConfiguration {
    @Value("${http.username}") var username: String = _
      @Value("${http.password}") var password: String = _
      @Value("${http.host}") var host: String = _
}
```

JSONHelpers.scala

```
    package com.clearaccess.example.api
    import org.json.{JSONArray, JSONObject}
    import collection.mutable.ListBuffer

    /**
      * This mixin gives implicit conversions for JSON arrays and
    object into rich
      * wrappers allowing more complex common operations on those
    arrays and
      * objects.
     */
    trait JSONHelpers {
     /**
      * A rich wrapper around a JSON object
      * @param jsonObject the wrapped JSON object
      */
    class RichJSONObject(jsonObject: JSONObject) {
      def getOrUpdateJSONObject(key: String, default: JSONObject):
    JSONObject = {
            if (jsonObject.has(key))
            jsonObject.getJSONObject(key)
            else {
             jsonObject.put(key, default)
             default
           }
      }

      /**
        * Allows you to set an object property deep within a JSON
      object tree. For
        * example given: { x: { y: {} } }, you can call:
      deepPut(List("x", "y", "z"), "1") to
        * yield { x: { y: { z: 1 } } }
        *
       * @param path a list of keys forming the path of the property
      to set
       * @return the resulting JSON object
       */
    def deepPut(path: List[String], value: String): JSONObject = {
    def getObject(path: List[String], parent: JSONObject): JSONObject
    = if (path.length > 1) {
        val child =
    toRichJSONObject(parent).getOrUpdateJSONObject(path.head, new
    JSONObject)
```

```
    getObject(path.tail, child)
    } else {
        parent
    }
    getObject(path, jsonObject).put(path.last, value)
    jsonObject }
    /**
      * Allows you to set an object property deep within a JSON
  object tree.
      * For example given: { x: { y: {} } }, you can call:
  deepPut("x.y.z", "1") to
      * yield { x: { y: { z: 1 } } }
      * @param path a '.' separated path of to the property to
  set
      * @return the resulting JSON object
      */
    def deepPut(path: String, value: String): JSONObject =
  deepPut(path.split('.').toList, value) }
    /**
      * A rich wrapper around a JSON array
      * @param jsonObject the wrapped JSON object
      */
    class RichJSONArray(jsonArray: JSONArray) {
       /**
         * Converts a JSON array to a list of json objects.
  useful for
         * arrays in the form: [{}, {}, ... {}]
         *
         * @return a list of JSON objects
         */
    def toJSONObjectList(): List[JSONObject] = {
        var buffer = new ListBuffer[JSONObject]
        for (ix <- 0 until jsonArray.length) buffer +=
jsonArray.getJSONObject(ix)
        buffer.toList
        }
    }

    /**
      * An implicit conversion to the wrapped JSON object
      */
     implicit def toRichJSONObject(jsonObject: JSONObject) = new
RichJSONObject(jsonObject)

    /**
      * An implicit conversion to the wrapped JSON array
      */
    implicit def toRichJSONArray(jsonArray: JSONArray) = new
    RichJSONArray(jsonArray)
    }
```

Interacting with Search

Use Case

Get all serial numbers for all devices with a given OUI.

Method

This example executes a CAQL query against the search service.

HTTP Interaction

Request

```
POST /prime-home/portal/query/execute
device with oui: 001A2B
```

Response


```
{
  "docId": "subscription#null#84",
  "fields": {
     "deviceId": "84",
     "disposition": "MANAGED_DEVICE",
     "lastInform": "2010-11-11T23:15:32.016Z",
     "manufacturer": "ClearAccess",
     "model": "AG10W-NA2",
     "oui": "001A2B",
     "serialNumber": "001A2B0AB3C2",
     "wanType": "ADSL2+",
     "wanip": ["010.001.001.050"]
  },
  "id": "null#84",
  "type": "subscription"
},
{
  "docId": "subscription#null#77",
  "fields": {
    "deviceId": "77",
    "disposition": "FUTURE_DEVICE",
    "oui": "001A2B",
    "serialNumber": "ASDFASDFASDF"
  },
  "id": "null#77",
  "type": "subscription"
},
{
  "docId": "subscription#8#78",
  "fields": {
    "deviceId": "78",
    "disposition": "MANAGED_DEVICE",
    "emailAddress": "00001@isp.net",
    "fullName": "kender elford",
    "lastInform": "2010-12-02T01:43:44.172Z",
    "manufacturer": "ClearAccess",
    "model": "AG10W-NA2",
    "oui": "001A2B",
    "personLabel": ["active"],
    "serialNumber": "001A2B3C09B7",
    "subscriberCode": "00001",
    "subscriberId": "8",
    "wanType": "ADSL2+",
    "wanip": ["010.001.001.016"]
},
  "id": "8#78",
  "type": "subscription"
}
```

Code

SearchService.scala

```
package com.clearaccess.example.api

import org.springframework.beans.factory.annotation.Autowired
import org.springframework.web.client.RestTemplate
import com.clearaccess.example.HttpConfiguration
import org.json.JSONArray
import org.springframework.stereotype.Component
@Component
class SearchService @Autowired()(
  val restTemplate: RestTemplate,
  val httpConfiguration: HttpConfiguration
  ) {
  /**
    * executes a CAQL query and returns results
    *
    * services:
    * POST /prime-home/portal/query/execute
    *
    * @param caql the ClearAccess Query Language string to
execute
    * @return the search result json array wrapped in a
SearchResults object
    */
  def query(caql: String): SearchResults = {
     val queryUrl =
"http://%s/prime-
home/portal/query/execute".format(httpConfiguration.host)
  new SearchResults(restTemplate.postForObject(queryUrl, caql,
classOf[JSONArray]))
  }
}
```

SearchResult.scala

```
package com.clearaccess.example.api

import org.json.{JSONObject, JSONArray}

class SearchResult(val json: JSONObject) extends JSONHelpers {
  /**
    * @return the underlying json object
    */
  def getJSON(): JSONObject = json
  /**
```

```
     * gets a value from the 'fields' object
     *
     * @param field the field name
     * @return the string value of the field specified by the key,
or null
     */
  def getString(key: String): String = {
      json.getOrUpdateJSONObject("fields", new
JSONObject).optString(key)
  }
  /**
     * gets a value from the 'fields' object
     *
     * @param field the field name
     * @return the long value of the field specified by the key,
  or null
     */
  def getLong(key: String): Long = {
      json.getOrUpdateJSONObject("fields", new
JSONObject).optLong(key)
  }
}

class SearchResults(val json: JSONArray) extends JSONHelpers with
Traversable[SearchResult] {
  /**
     * @return the underlying json array
     */
  def getJSON(): JSONArray = json
   /**
   * @return the number of items in the result set. this does not
account for pagination!
   */
    def resultCount(): Int = json.length

    /**
   * searches the result set for results matching the given key
and value
   *
   * @param key the field to query
   * @param value the value of the field to check
   *
   * @return a collection of matching SearchResults
   */
  def findResultByFieldValue(key: String, value: String):
java.util.Collection[SearchResult] = {
       val found = new java.util.LinkedList[SearchResult]
       for (i <- 0 until json.length) {
         val result = new SearchResult(json.getJSONObject(i))
         if (result.getString(key) == value)
             found.add(result)
        }
        return found
}
/**
   * @return the first result, wrapped in a SearchResult. handy
for unique queries
   */
def first(): SearchResult = {
  new SearchResult(json.getJSONObject(0))
}

def foreach[U](f: (SearchResult) => U): Unit = {
  val results = json.toJSONObjectList().map(j => new
SearchResult(j))
```

```
      for (result <- results) f(result)
    }
}
```

SearchUseCase.scala

```
package com.clearaccess.example

import api.SearchService
import org.scalatest.matchers.MustMatchers
import org.scalatest.{FeatureSpec, GivenWhenThen}
import
org.springframework.context.annotation.AnnotationConfigApplicationC
ontext

class SearchUseCases extends FeatureSpec with GivenWhenThen with
MustMatchers {
   val applicationContext = new
AnnotationConfigApplicationContext("com.clearaccess")
   val searchService =
applicationContext.getBean(classOf[SearchService])
   feature("the integrator searches for devices") {
      scenario("retrieving all device serial numbers") {
        given("a CAQL query with an OUI")
        val caql = "devices with oui: 001A2B"
        when("the search API is invoked")
        val results = searchService.query(caql)
        then("print out all of the serial numbers")
        for (result <- results) {
            val serialNumber = result.getString("serialNumber")
            println(serialNumber)
         }
      }
   }
}
```

Interacting with the Device Lifecycle

Use Case

This example uses the search, subscriber, and device APIs to illustrate managing a device through its entire lifecycle.

Method

1. Create a new subscriber
2. Associate a device with that subscriber
3. Change the state of a service to 'ENABLED'
4. Change some application data with synchronization
5. Perform an RMA on a 'broken' device

HTTP Interaction

Creating the Subscriber

Request

```
GET /prime-home/api/v2/templates/subscriber
```

Response

```
{
  "code": "",
  "credentials": {
     "login": "",
     "password": ""
  },
  "dto": {},
  "labels": [],
  "subscriptions": []
}
```

Request

```
POST /prime-home/api/v2/subscribers

{
  "code": "life-cycle-use-case-subscriber",
  "credentials": {},
  "dto": {},
  "labels": [],
  "subscriptions": []
}
```

Response

```
{
  "code": "life-cycle-use-case-subscriber",
  "credentials": {},
  "dto": {},
  "id":3,
  "labels": [],
```

```
    "revision": "BBDC7558BAA568CAD7BFD072DF37F828",
    "subscriptions": []
}
```

Finding / Creating a Device

Request

```
POST /prime-home/portal/query/execute

device with oui: 001A2B999999
```

Response

```
[]
```

Request

```
GET /prime-home/api/v1/templates/device
```

Response

```
{
    "actionLog": [],
    "applications": {
        (...)
    },
    "labels": [],
    "queuedActions": {
        "applications": {
        (...)
    },
        "scripts": [],
        "services": {
            (...)
        }
    }
}
```

Request

```
POST /prime-home/api/v1/devices

{
    "sn": "001A2B999999", oui: "001A2B",
    "subscriberCode": "life-cycle-use-case-subscriber",
    "actionLog": [],
    "applications": {
        (...)
    },
    "labels": [],
    "queuedActions": {
    "applications": {
        (...)
    },
    "scripts": [],
    "services": {
        (...)
    }
}
```

Response

```
{
```

```
  "actionLog": [],
  "applications": {
     (...)
  },
  "deviceId":1231564,
  "disposition":"FUTURE_DEVICE",
  "labels": [],
  "oui":"001A2B",
  "queuedActions": {
     "applications": {
        (...)
  },
  "scripts": [],
  "services": {
      (...)
  }
  "sn":"001A2B999999",
  "subscriberCode":"life-cycle-use-case-subscriber"
}
```

Request

```
POST /prime-home/api/v1/devices

{
  "sn": "001A2B999999", oui: "001A2B",
  "subscriberCode": "life-cycle-use-case-subscriber",
  "actionLog": [],
  "applications": {
     (...)
  },
  "labels": [],
  "queuedActions": {
     "applications": {
        (...)
     },
     "scripts": [],
     "services": {
        (...)
     }
  }
}
```

Response

```
{
  "actionLog": [],
  "applications": {
      (...)
  },
  "deviceId":1231564,
  "disposition":"FUTURE_DEVICE",
  "labels": [],
  "oui":"001A2B",
  "queuedActions": {
     "applications": {
        (...)
     },
     "scripts": [],
     "services": {
        (...)
     }
     "sn":"001A2B999999",
     "subscriberCode":"life-cycle-use-case-subscriber"
```

```
}
```

Enabling Services

Request

```
POST /prime-home/portal/query/execute

device with oui: 001A2B999999
```

Response

```
{
  "docId": "subscription#8#78",
     "fields": {
     "deviceId": "78",
     "disposition": "MANAGED_DEVICE",
     "lastInform": "2010-12-06T19:19:59.486Z",
     "manufacturer": "ClearAccess",
     "model": "AG10W-NA2",
     "oui": "001A2B",
     "personLabel": ["kender"],
     "serialNumber": "001A2B999999",
     "subscriberCode": "life-cycle-use-case-subscriber",
     "subscriberId": "8",
     "wanType": "ADSL2+",
     "wanip": ["010.001.001.016"]
  },
  "id": "8#78",
  "type": "subscription"
}
```

Request

```
GET /prime-home/api/v1/devices/78/actions
```

Response

```
{
  "aplications": {
      (...)
  },
  "revision": -1537035015,
  "scripts": [],
  "services": {
     "CF": {
        "canToggleStatus": true,
        "status": "DISABLED"
     },
     (...)
  },
  "solicitableStatus": "OK"
}
```

Request

```
PUT /prime-home/api/v1/devices/78/actions

{
  "applications": {
      (...)
  },
  "revision": -1537035015,
  "scripts": [],
  "services": {
     "CF": {
```

```
            "canToggleStatus": true,
            "status": "ENABLED"
         },
         (...)
      },
      "solicit": true",
      "solicitableStatus": "OK"
   }
```

Response

```
   {
      "applications": {
          (...)
      },
      "revision": -1537035015,
      "scripts": [],
      "services": {
         "CF": {
            "canToggleStatus": true,
            "status": "ENABLING"
         },
         (...)
      },
      "solicitableStatus": "OK"
   }
```

Updating an Application

Request

```
POST /prime-home/portal/query/execute

device with oui: 001A2B999999
```

Response


```
{
  "docId": "subscription#8#78",
     "fields": {
     "deviceId": "78",
     "disposition": "MANAGED_DEVICE",
     "lastInform": "2010-12-06T19:19:59.486Z",
     "manufacturer": "ClearAccess",
     "model": "AG10W-NA2",
     "oui": "001A2B",
     "personLabel": ["kender"],
     "serialNumber": "001A2B999999",
     "subscriberCode": "life-cycle-use-case-subscriber",
     "subscriberId": "8",
     "wanType": "ADSL2+",
     "wanip": ["010.001.001.016"]
  },
  "id": "8#78",
  "type": "subscription"
  }
```


Request

```
GET /prime-home/api/v1/devices/78/data
```

Response


```
{
  "applications": {
     "WIFI": {"dto": {"Settings": {"WLANConfigurations": {"1": {
        "BeaconAdvertisementEnabled": "0",
        "BeaconType": "WPA2",
        "Channel": "11",
        "Enable": "0",
        "SSID": "ClearAccess",
        "Secret": "G846DkDRDj64byr"
     }}}}}
  },
  "deviceId": 78,
  "deviceTypeId": 3,
  "deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
  "disposition": "MANAGED_DEVICE",
  "excludeFromBulkOperation": false,
  "firstInform": "2010-11-04T19:56:22.323Z",
  "friendlyManufacturerName": "ClearAccess",
  "friendlyName": "AG10W-NA2",
  "informCount": 12438,
  "labels": [],
  "lastInform": "2010-12-06T19:19:59.486Z",
  "manufacturer": "ClearAccess",
  "oui": "001A2B",
  "revision": -15500043644,
  "sn": "001A2B999999",
  "softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
  "subscriberCode": "kender"
}
```

Request

```
PUT /prime-home/api/v1/devices/78/data

{
  "applications": {
    "WIFI": {"dto": {"Settings": {"WLANConfigurations": {"1": {
       "BeaconAdvertisementEnabled": "0",
       "BeaconType": "WPA2",
       "Channel": "11",
       "Enable": "0",
       "SSID": "life-cycle-use-case-ssid",
       "Secret": "G846DkDRDj64byr"
    }}}}},
    (...)
  },
  "deviceId": 78,
  "deviceTypeId": 3,
  "deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
  "disposition": "MANAGED_DEVICE",
  "excludeFromBulkOperation": false,
  "firstInform": "2010-11-04T19:56:22.323Z",
  "friendlyManufacturerName": "ClearAccess",
  "friendlyName": "AG10W-NA2",
  "informCount": 12438,
  "labels": [],
  "lastInform": "2010-12-06T19:19:59.486Z",
  "manufacturer": "ClearAccess",
  "oui": "001A2B",
  "revision": -15500043644,
  "sn": "001A2B999999",
  "softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
  "subscriberCode": "kender"
}
```

Response

```
{
  "applications": {
    "WIFI": {"dto": {"Settings": {"WLANConfigurations": {"1": {
       "BeaconAdvertisementEnabled": "0",
       "BeaconType": "WPA2",
       "Channel": "11",
       "Enable": "0",
       "SSID": "life-cycle-use-case-ssid",
       "Secret": "G846DkDRDj64byr"
    }}}}},
    (...)
  },
  "deviceId": 78,
  "deviceTypeId": 3,
  "deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
  "disposition": "MANAGED_DEVICE",
  "excludeFromBulkOperation": false,
  "firstInform": "2010-11-04T19:56:22.323Z",
  "friendlyManufacturerName": "ClearAccess",
  "friendlyName": "AG10W-NA2",
  "informCount": 12438,
  "labels": [],
  "lastInform": "2010-12-06T19:19:59.486Z",
  "manufacturer": "ClearAccess",
  "oui": "001A2B",
  "revision": -15500043644,
  "sn": "001A2B999999",
  "softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
  "subscriberCode": "kender"
```

```
    }
```

Request

```
    GET /prime-home/api/v1/devices/78/actions
```

Response

```
    {
      "applications": {
         "WIFI": {
            "dataOwner": "SERVER",
            "needsInitialization": false,
            "pendingSync": false
         },
         (...)
      },
      "revision": -1537035015,
      "scripts": [],
      "services": {
         "CF": {
            "canToggleStatus": true,
            "status": "DISABLED"
         },
         (...)
      },
      "solicitableStatus": "OK"
    }
```

Request

```
    PUT /prime-home/api/v1/devices/78/actions

    {
      "applications": {
         "WIFI": {
            "dataOwner": "SERVER",
            "needsInitialization": false,
            "pendingSync": true
         },
         (...)
      },
      "revision": -1537035015,
      "scripts": [],
      "services": {
         "CF": {
            "canToggleStatus": true,
            "status": "DISABLED"
         },
         (...)
      },
      "solicitableStatus": "OK"
    }
```

Response

```
    {
      "applications": {
         "WIFI": {
            "dataOwner": "SERVER",
            "needsInitialization": false,
```

```
            "pendingSync": true
         },
         (...)
      },
      "revision": -1537035015,
      "scripts": [],
      "services": {
         "CF": {
            "canToggleStatus": true,
            "status": "DISABLED"
         },
         (...)
      },
      "solicitableStatus": "OK"
   }
```

Replacing & Deleting a Device

Request

```
POST /prime-home/portal/query/execute
device with oui: 001A2B999998
```

Response

```
[]
```

Request

```
GET /prime-home/api/v1/templates/device
```

Response

```
{
  "actionLog": [],
  "applications": {
     (...)
  },
  "labels": [],
  "queuedActions": {
     "applications": {
     (...)
  },
  "scripts": [],
  "services": {
     (...)
  }
 }
}
```

Request

```
POST /prime-home/api/v1/devices

{
  "sn": "001A2B999998", oui: "001A2B",
  "actionLog": [],
  "applications": {
      (...)
  },
  "labels": [],
  "queuedActions": {
     "applications": {
        (...)
     },
     "scripts": [],
     "services": {
        (...)
     }
  }
}
```

Response

```
{
  "actionLog": [],
  "applications": {
      (...)
  },
  "deviceId":79,
  "disposition":"FUTURE_DEVICE",
  "labels": [],
  "oui":"001A2B",
  "queuedActions": {
     "applications": {
     (...)
  },
  "scripts": [],
  "services": {
      (...)
  }
  "sn":"001A2B999998",
 }
}
```

Request

GET /prime-home/api/v1/devices/79/data

Response


```
{
  "applications": {
     (...)
  },
  "deviceId": 79,
  "deviceTypeId": 3,
  "deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
  "disposition": "MANAGED_DEVICE",
  "excludeFromBulkOperation": false,
  "firstInform": "2010-11-04T19:56:22.323Z",
  "friendlyManufacturerName": "ClearAccess",
  "friendlyName": "AG10W-NA2",
  "informCount": 12438,
  "labels": [],
  "lastInform": "2010-12-06T19:19:59.486Z",
  "manufacturer": "ClearAccess",
  "oui": "001A2B",
  "revision": -15500043644,
  "sn": "001A2B999998",
  "softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
}
```


Request


```
PUT /prime-home/api/v1/devices/79/data

{
  "applications": {
     (...)
},
"deviceId": 79,
"deviceTypeId": 3,
"deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
"disposition": "MANAGED_DEVICE",
"excludeFromBulkOperation": false,
"firstInform": "2010-11-04T19:56:22.323Z",
"friendlyManufacturerName": "ClearAccess",
"friendlyName": "AG10W-NA2",
"informCount": 12438,
"labels": [],
"lastInform": "2010-12-06T19:19:59.486Z",
"manufacturer": "ClearAccess",
"oui": "001A2B",
"revision": -15500043644,
"rmaFor": {
  "code": "BROKEN"
  "oui": "001A2B",
  "sn": "001A2B999999"
}
"sn": "001A2B999998",
"softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
}
```


Response


```
{
  "applications": {
```

```
        (...)
      },
      "deviceId": 79,
      "deviceTypeId": 3,
      "deviceTypeImagePath": "/acs-portal/images/devices/ag10w.png",
      "disposition": "MANAGED_DEVICE",
      "excludeFromBulkOperation": false,
      "firstInform": "2010-11-04T19:56:22.323Z",
      "friendlyManufacturerName": "ClearAccess",
      "friendlyName": "AG10W-NA2",
      "informCount": 12438,
      "labels": [],
      "lastInform": "2010-12-06T19:19:59.486Z",
      "manufacturer": "ClearAccess",
      "oui": "001A2B",
      "revision": -15500043644,
      "rmaFor": {
         "code": "BROKEN"
         "oui": "001A2B",
         "sn": "001A2B999999"
      }
      "sn": "001A2B999998",
      "softwareVersion": "2.3.0.13_4.02L.03.wp1.A2pB025c1.d21j2",
      "subscriberCode": "life-cycle-use-case-subscriber"
   }
```


Request

```
DELETE /prime-home/api/v1/devices/78
```

Response

Sample Code

DeviceActions.scala

```
package com.clearaccess.example.api

import org.json.JSONObject

class DeviceActions(val json: JSONObject) extends JSONHelpers {
  /**
    * @return the underlying json object
    */
  def getJSON(): JSONObject = json
    /**
    * sets the state of a service to 'ENABLED' or 'DISABLED'
    * @param serviceCode the service code of the service. this
    service must be present
    * @param isEnabled determines whether to set the status to
    'ENABLED' or 'DISABLED'
    * @return self, for chaining
    */
  def setServiceEnabled(serviceCode: String, isEnabled: Boolean):
DeviceActions = {
     val status = if (isEnabled) "ENABLED" else "DISABLED"
     json.deepPut(List("services", serviceCode, "status"),
     status)
     this
  }

  /**
   * marks an application state as requiring a synchronization
   *
   * @param applicationCode the code of the application to be
   marked
   * @param pendingSync whether or not the application needs to
   be
   synced
   *
   * @return self, for chaining
   */
def setApplicationPendingSync(applicationCode: String,

pendingSync: Boolean): DeviceActions = {
  json
     .getJSONObject("applications")
     .getJSONObject(applicationCode)
     .put("pendingSync", pendingSync)
  this
}
/**
 * set to make the system request a connection from the device
 *
 * @param applyImmediately whether the changes should be applied
 immediately to the device
 *
 * @return self, for chaining
 */
def setApplyImmediately(applyImmediately: Boolean): DeviceActions
= {
  json.put("solicit", applyImmediately)
  this
}
}
```

DeviceData.scala

```
package com.clearaccess.example.api

import org.json.JSONObject

class DeviceData(val json: JSONObject) extends JSONHelpers {
   /**
    * @return the underlying json object
    */
   def getJSON() = json
   /**
    * @return the device id as a long value
    */
   def getDeviceId(): Long = json.optLong("deviceId")

   /**
    * sets the oui and serial number of the device
    *
    * @param oui device oui
    * @param sn device serial number
    * @return self, for chaining
    */
def setOuiSn(oui: String, sn: String): DeviceData = {
      json.put("oui", oui).put("sn", sn)
      this
}

/**
  * associates this device to an existing subscriber referenced by
  * the given subscriber code
  *
  * @param subscriberCode the code of the subscriber who is
  associated with this device
  * @return self, for chaining
  */
def setSubscriberCode(subscriberCode: String): DeviceData = {
  json.put("subscriberCode", subscriberCode)
   this
}

/**
 * Marks this device as being replaced with the device referenced  by
 the parameters.
 *
 * @param sn replacement device sn
 * @param oui replacement device oui
 * @code code integrator specific rma code
 */
def setRMA(sn: String, oui: String, code: String): DeviceData = {
   val rmaFor = new JSONObject
   rmaFor
   .put("sn", sn)
   .put("oui", oui)
   .put("code", code)
   json.put("rmaFor", rmaFor)
   this
}

/**
 * Marks this device as being replaced with the device referenced
 by the parameters.
 *
 * @param sn replacement device sn
 * @param oui replacement device oui
```

```
 */
def setRMA(sn: String, oui: String): DeviceData = {
   val rmaFor = new JSONObject
   rmaFor
      .put("sn", sn)
      .put("oui", oui)
   json.put("rmaFor", rmaFor)
   this
}

/**
 *
 */
def setApplicationDTO(applicationCode: String, name: String, value:
String): DeviceData = {
      json
         .getJSONObject("applications")
         .getJSONObject(applicationCode)
         .getJSONObject("dto")
         .deepPut(name, value)
      this
   }
}
```

DeviceService.scala

```
package com.clearaccess.example.api

import org.springframework.beans.factory.annotation.Autowired
import com.clearaccess.spring.rest.RestTemplate
import com.clearaccess.example.HttpConfiguration
import org.springframework.stereotype.Component
import org.json.JSONObject

@Component
class DeviceService @Autowired()(
   val restTemplate: RestTemplate,
   val httpConfiguration: HttpConfiguration
   ) extends JSONHelpers {
   private def devicesUrl: String =

"http://%s/prime-home/api/v1/devices".format(httpConfiguration.host)

   private def deviceTemplateUrl: String =

"http://%s/prime-
home/api/v1/templates/device".format(httpConfiguration.host)

   private def deviceDataUrl(deviceId: Long): String =

"http://%s/prime-
home/api/v1/devices/%d/data".format(httpConfiguration.host, deviceId)

   private def deviceUrl(deviceId: Long): String =

"http://%s/prime-home/api/v1/devices/%d".format(httpConfiguration.
host, deviceId)

   private def deviceActionsUrl(deviceId: Long): String =

"http://%s/prime-home/api/v1/devices/%d/actions".format(httpConfig
uration.host, deviceId)
```

```
   /**
    * creates a new device with a serial number and oui associated to
    a subscriber
    *
    * services:
    * GET /prime-home/api/v1/templates/device
    * POST /prime-home/api/v1/devices
    *
    * @param sn device serial number
    * @param oui device oui
    * @param subscriberCode the unique identifier of the device's
    subscriber
    * @return the device's json object data, wrapped in a DeviceData
    object
    */
    def createDevice(sn: String, oui: String, subscriberCode:String):
DeviceData = {
      val deviceData = new
DeviceData(restTemplate.getForObject(deviceTemplateUrl,classOf[JSONOb
ject]))
      .setOuiSn(oui, sn)
      .setSubscriberCode(subscriberCode)
   val deviceId = new
DeviceData(restTemplate.postForObject(devicesUrl, deviceData.json,
classOf[JSONObject])).getDeviceId
      getDeviceData(deviceId)
   }
   /**
    * deletes a device by it's unique numeric id
    *
    * services:
    * DELETE /prime-home/api/v1/devices/{deviceId}
    *
    * @param deviceId the device's unique id
    */
   def deleteDevice(deviceId: Long): Unit = {
      restTemplate.delete(deviceUrl(deviceId))
   }

   /**
    * gets the device's manipulable data.
    *
    * services:
    * GET /prime-home/api/v1/devices/{deviceId}/data
    *
    * @param deviceId the device's unique id
    * @return the device's data as a json object, wrapped in a
    DeviceData object
    */
   def getDeviceData(deviceId: Long): DeviceData = {
      new
DeviceData(restTemplate.getForObject(deviceDataUrl(deviceId),
classOf[JSONObject]))
   }

   /**
    * updates the device's manipulable data.
    *
    * services:
    * PUT /prime-home/api/v1/devices/{deviceId}/data
    *
    * @param deviceId the device's unique id
    * @param deviceData the device's data to send to the server
    * @return the device's updated data as a json object, wrapped in
    a DeviceData object
```

```
    */
   def putDeviceData(deviceId: Long, deviceData: DeviceData):
DeviceData = {
      new
DeviceData(restTemplate.putForJSONObject(deviceDataUrl(deviceId),
deviceData.json))
   }

   /**
    * gets the device's actions state.
    *
    * services
    * GET /prime-home/api/v1/devices/{deviceId}/actions
    *
    * @param deviceId the device's unique id
    * @return the device's actions data as a json object, wrapped in
    a DeviceActions object
    */
   def getDeviceActions(deviceId: Long): DeviceActions = {
      new
DeviceActions(restTemplate.getForObject(deviceActionsUrl(deviceId),
classOf[JSONObject]))
   }

   /**
    * updates the device's actions state.
    *
    * services
    *   PUT /prime-home/api/v1/devices/{deviceId}/actions
    *
    * @param deviceId the device's unique id
    * @param deviceActions the device actions to send to the server
    * @return the device's updated actions data as a json object,
    wrapped in a DeviceActions object
    */
   def putDeviceActions(deviceId: Long, deviceActions:DeviceActions):
   DeviceActions = {
      new
DeviceActions(restTemplate.putForJSONObject(deviceActionsUrl(device
Id), deviceActions.json))
   }
}
```

LifeCycleUseCases.scala

```
package com.clearaccess.example

import api.{SearchService, DeviceService, SubscriberService}
import
org.springframework.context.annotation.AnnotationConfigApplicationC
ontext
import org.scalatest.matchers.MustMatchers
import org.scalatest.{FeatureSpec, GivenWhenThen}
class LifeCycleUseCases extends FeatureSpec with GivenWhenThen with
MustMatchers {
   val applicationContext = new
AnnotationConfigApplicationContext("com.clearaccess")
   val subscriberService =
applicationContext.getBean(classOf[SubscriberService])
   val deviceService =
applicationContext.getBean(classOf[DeviceService])
   val searchService =
applicationContext.getBean(classOf[SearchService])
```

```
   feature("Using the APIs to interact with the device lifecycle") {
      given("a subscriber code, device oui / serial number")
      val subscriberCode = "life-cycle-use-case-subscriber"
      val deviceOui = "001A2B"
      val deviceSerialNumber = "001A2B999999"

      scenario("Setup: Create a new subscriber for the device") {
         val subscriber =
      subscriberService.createSubscriber(subscriberCode)
subscriber.setEmailAddress("%s@isp.com".format(subscriberCode))
         subscriberService.updateSubscriber(subscriberCode,subscriber
         )
      }

      scenario("Setup: Create a new future device, associated with
the subscriber") {
         val caql = "device with oui: %s and serialNumber:%s".format
         (deviceOui, deviceSerialNumber)
         val deviceSearchResults = searchService.query(caql)
       if (deviceSearchResults.resultCount == 0) {
         given("that the device does not pre-exist in the system")
         then("create the device with a serial number, oui, and
         subscriber code")
         deviceService.createDevice(deviceSerialNumber,
         deviceOui, subscriberCode)
       } else {
         given("that the device is already in the system")
         val deviceId = deviceSearchResults.first.getLong("deviceId")
         val deviceData = deviceService.getDeviceData(deviceId)
         then("set that device's subscriber code on that device")
         deviceData.setSubscriberCode(subscriberCode)
         and("PUT the device data back to the server")
         deviceService.putDeviceData(deviceId, deviceData)
       }
   }

   scenario("Activation: The device checks into the ACS") {
      given("Device Initial Contact")
      then("The device becomes managed")
   }
   scenario("Management: Change services") {
      given("an existing device")
      val caql = "device with oui: %s and
      serialNumber:%s".format(deviceOui, deviceSerialNumber)
      val deviceSearchResults = searchService.query(caql)
      val deviceId = deviceSearchResults.first.getLong("deviceId")
      val deviceActions = deviceService.getDeviceActions(deviceId)

      and("a service code")
      val serviceCode = "lifeCycleService"

      then("set the service to 'ENABLED'")
      deviceActions.setServiceEnabled(serviceCode, true)

      and("PUT the device actions back to the server")
      deviceService.putDeviceActions(deviceId, deviceActions)
   }

   scenario("Management: Change settings") {
      given("an existing device")
      val caql = "device with oui: %s and
      serialNumber:%s".format(deviceOui, deviceSerialNumber)
      val deviceSearchResults = searchService.query(caql)
      val deviceId = deviceSearchResults.first.getLong("deviceId")
      val deviceData = deviceService.getDeviceData(deviceId)
```

```
    and("a new SSID for the wireless interface")
    val ssid = "life-cycle-use-case-ssid"

    then("set the data in the WIFI application's dto")
    deviceData.setApplicationDTO("WIFI","Settings.WLANConfiguration
    s.1.SSID", ssid)
    and("PUT the device data back to the server")
    deviceService.putDeviceData(deviceId, deviceData)

    then("get the device actions")
    val deviceActions = deviceService.getDeviceActions(deviceId)

    and("mark 'WIFI' as needing to be synced")
    deviceActions.setApplicationPendingSync("WIFI", true)

    and("PUT the device actions back to the server")
    deviceService.putDeviceActions(deviceId, deviceActions)
 }
 scenario("Termination: Replace a broken device") {
    given("an existing device")
    val brokenDeviceOui = deviceOui
    val brokenDeviceSerialNumber = deviceSerialNumber
    and("an oui/serial number for a device to replace with")
    val replacementDeviceOui = "001A2B"
    val replacementDeviceSerialNumber = "001A2B99998"

    then("set the 'rmaFor' oui / serial number in the
 replacement device to the original device")
    val (replacementDeviceId, replacementDeviceData) = {
    val caql = "device with oui: %s and serialNumber:
 %s".format(replacementDeviceOui, replacementDeviceSerialNumber)
    val deviceSearchResults = searchService.query(caql)
    if (deviceSearchResults.resultCount == 0) {
       given("that the device does not pre-exist in the system")
       then("create the device with a serial number, oui, and
       subscriber code")
       val replacementDevice =deviceService.createDevice
    (replacementDeviceOui,replacementDeviceSerialNumber,
    subscriberCode)
       (replacementDevice.getDeviceId, replacementDevice)
     } else {
       given("that the device is already in the system")
       val deviceId = deviceSearchResults.first.getLong("deviceId")
       (deviceId, deviceService.getDeviceData(deviceId))
     }
  }
  replacementDeviceData.setRMA(brokenDeviceSerialNumber,
brokenDeviceOui, "BROKEN")

  and("PUT the replacement device data back to the server")
  deviceService.putDeviceData(replacementDeviceId,replacementDevic
  eData)

  and("find the old device id by oui / serial number")
  val brokenDeviceId = {
     val caql = "device with oui: %s and serialNumber:
  %s".format(brokenDeviceOui, brokenDeviceSerialNumber)
     val deviceSearchResults = searchService.query(caql)
     deviceSearchResults.first.getLong("deviceId")
  }

  and("DELETE the old device")
  deviceService.deleteDevice(brokenDeviceId)
  }
```

```
    }
}
```

Creating Control Panel sessions for Single Sign On

Use Case

Create a 'bridge' between an external system and the ACS to provide proxied authentication to the
Control Panel, implementing a single sign on solution for the subscriber.

Method

1. Determine the mac address of a device to use for the subscriber / device for the control panel
   a. Can come from a pre-determined mapping of arbitrary guids to macs
   b. Can come from the mac parameter of the request
2. Split the mac address into oui and serial number
3. Create a control panel session by passing the oui / serial number into the session service with an authenticated HTTP connection
4. Form a url out of: the oui, serial number, and session id
5. Redirect the browser

HTTP Interaction (Web Services)

Request

```
POST /prime-home/api/v1/sessions/controlPanel

{
   "device": {
      "sn": "001A2B999999",
      "oui": "001A2B"
   }
}
```

Response

```
{
  "cpSessionId": "527BBB943DAAE5838D11298EAC8442DABD3BC2A0",
  "expires": "2010-12-06T22:55:35.860Z",
  "subscriberCode": "testSubscriber",
  "username": "testSubscriber1"
}
```

HTTP Interaction (Browser)

1. `GET /acs-portal/control-panel/login?cpSessionId=527BBB943DAAE5838D11298EAC8442DABD3BC2A0&device=001A2B:001A2B999999`, confirms the session and configures the browser's cookies, forwarding them to the control panel UI, if all is good.
2. `GET /acs-portal/controlpanel?cpSessionId=527BBB943DAAE5838D11298EAC8442DABD3BC2A0&device=001A2B:001A2B999999,` the actual control panel UI.

Sample Code

ControlPanelSessionService.scala

```
package com.clearaccess.example.api

import org.springframework.beans.factory.annotation.Autowired
import com.clearaccess.spring.rest.RestTemplate
import com.clearaccess.example.HttpConfiguration
import org.json.JSONObject
import org.springframework.stereotype.Component

@Component
class ControlPanelSessionService @Autowired()(
   val restTemplate: RestTemplate,
   val httpConfiguration: HttpConfiguration
   )   {
   private lazy val sessionsControlPanelURL =
"http://%s/prime-home/api/v1/sessions/controlPanel".format(httpCon
figuration.host)
   /**
    * makes an authenticated call to create a control panel session
    for the subscriber
    *
    * @param subscriberCode the unique code of the subscriber
    * @return a control panel session with a cpSessionId
    */
  def createProxySession(subscriberCode: String):
ControlPanelSession = {
      val cpSession = new ControlPanelSession(new JSONObject)
      cpSession.setSubscriberCode(subscriberCode)
      new
ControlPanelSession(restTemplate.postForObject(sessionsControlPanel
URL, cpSession.json, classOf[JSONObject]))
  }

  /**
   * makes an authenticated call to create a control panel session
   for the subscriber
   * who owns the device indicated by the provided oui / serial
   number
   *
   * @param oui the subscriber's device's oui
   * @param sn the subscriber's device's serial number
   * @return a control panel session with a cpSessionId
   */
  def createProxySession(oui: String, sn:
  String):ControlPanelSession = {
     val cpSession = new ControlPanelSession(new JSONObject)
     cpSession.setOuiSn(oui, sn)
     new
```

```
ControlPanelSession(restTemplate.postForObject(sessionsControlPanel
URL, cpSession.json, classOf[JSONObject]))
    }
}
```

AuthBridgeHttpRequestHandler.scala

```
package com.clearaccess.example.auth

import org.springframework.web.HttpRequestHandler
import javax.servlet.http.{HttpServletResponse, HttpServletRequest}
import com.clearaccess.example.api.ControlPanelSessionService
import com.clearaccess.example.{HttpConfiguration,
HttpServletSupport}
import org.springframework.beans.factory.annotation.Autowired

/**
 * This HTTP Request Handler (see spring's HttpRequestHandler and
 web.xml):
 * - takes a 'guid' string parameter
 * - maps the 'guid' into a device mac with the provided 'GuidMapper'
 * - transforms the mac into an oui/serial number (assumes the sn is
 = mac)
 * - makes an authenticated call to the session web service with the
 device info
 * - redirects the browser to the control panel with the oui:sn and
 session id
 *
 * === caveats ===
 * 1) GuidMapper must be implemented
 * 2) the mac address maps directly to the oui/sn of the device
 * 3) the control panel is located on the same host as the web
 service
 */
// xml wired
  class AuthBridgeHttpRequestHandler @Autowired()(
  val controlPanelSessionService: ControlPanelSessionService,
  val httpConfiguration: HttpConfiguration,
  val guidMapper: GuidMapper
  ) extends HttpRequestHandler
  with HttpServletSupport {
  def handleRequest(request: HttpServletRequest,
  response:HttpServletResponse): Unit = try {
    // get the mac request parameter
    val mac = getRequestParameter(request, "mac") getOrElse {
    // otherwise, get the guid request parameter
    // then call the guidMapper to convert the guid to a device mac
    getRequestParameter(request,"guid").map(guidMapper.guidToMAC)get
    OrElse {
       throw new AssertionError("mac or guid required")
    }
  }

  // extract the oui and serial number
  val oui: String = macOUI(mac)
  val sn: String = macSN(mac)

  // call the acs to create a session
  val cpSession = controlPanelSessionService.createProxySession(oui,
  sn)

  // redirect the browser to the control panel, preauthenticated
  val returnUrl = java.net.URLEncoder.encode("http://google.com",
  "UTF-8")
  val jSessionCookiePath = java.net.URLEncoder.encode("/acsportal")
  val location = "http://%s/acs-
  portal/controlpanel/login?device=%s:%s&
  cpSessionId=%s&styleVariant=pc&returnUrl=%s&jSessionCookiePath=%s"
  .format(
    httpConfiguration.host,
```

```
      oui, sn,
      cpSession.getCpSessionId,
      returnUrl,
      jSessionCookiePath
      )
    response.sendRedirect(location)
   } catch {
    // handle AssertionError as a generic HTTP client error case
    error: AssertionError => response.sendError(400,error.getMessage)
   }
  private def macOUI(mac: String): String =
  mac.substring(0,6).toUpperCase
  private def macSN(mac: String): String = mac.toUpperCase
}
```

Deprecated

✎ The following examples make use of deprecated APIs. However, they are still included in this guide as the principles involved remain valid.

Changing SSID with CURL

```
#!/bin/sh

BASEURL=http://localhost:9080/prime-home/api/current
USER=admin
PASS=admin

CODE=$1
SSID=$2

if [ -z "$CODE" ]; then
   echo "must supply a code"
   exit 1
fi

if [ -z "$SSID" ]; then
   echo "must supply a ssid"
   exit 1
fi

EXP=$(cat << EOS
s/\"Settings.WLANConfigurations.1.SSID\":\"[^\"]*\"/\"Settings.WLAN
Configurations.1.SSID\":\"$SSID\"/
EOS
)

N=`basename $0`
TMP=`mktemp ./${N}.XXXXX`

curl -u "$USER:$PASS" $BASEURL/subscribers/code:$CODE | sed -E -e
$EXP > $TMP
curl -X PUT -d @$TMP -u $USER:$PASS $BASEURL/subscriber/code:$CODE

rm $TMP
```

Working with Subscribers in Java

Introduction

This example illustrates using the Subscriber API to work thorugh the basic use case of creating /
changing subscriber information.

Main Example

com.clearaccess.subscriber.Example

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.subscriber;

import
org.springframework.context.annotation.AnnotationConfigApplicationCon
text;
import com.clearaccess.subscriber.api.Address;
import com.clearaccess.subscriber.api.Credentials;
import com.clearaccess.subscriber.api.Subscriber;
import com.clearaccess.subscriber.api.SubscriberAPI;

public class Example {
   public static void main(String[] args) {
      // setup spring
      final AnnotationConfigApplicationContext context =
      newAnnotationConfig
     ApplicationContext("com.clearaccess");
      final SubscriberAPI subscriberAPI =
      context.getBean(SubscriberAPI.class);
      // get existing subscriber or create a default one
      final Subscriber subscriber =
      subscriberAPI.getSubscriber("clearaccess");

      // set subscriber properties
      subscriber.setLabels(new String[]{"active"});
      subscriber.setCredentials(new Credentials("ca", "ca1234"));
      subscriber.setAddress(new Address("501 SE Columbia Shores
      Boulevard,
     Suite 500", "Vancouver", "WA", "98661"));
      subscriber.setPhoneNumber("360-859-1780");
      subscriber.setFullName("Clear Access");
      subscriber.setEmailAddress("support@clearaccess.com");


      // save the changes to the server
      subscriberAPI.updateSubscriber(subscriber);

      // remove the subscriber from the server
      subscriberAPI.deleteSubscriber("clearaccess");
   }
}
```

Interaction with the API

com.clearaccess.subscriber.api.SubscriberAPI

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.subscriber.api;

import org.springframework.stereotype.Component;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.client.HttpClientErrorException;
import org.springframework.http.HttpStatus;
import org.json.JSONObject;
import org.json.JSONException;
import com.clearaccess.rest.RestClient;
import com.clearaccess.Configuration;

@Component
public class SubscriberAPI {
   final RestClient restClient;
   final Configuration configuration;
   @Autowired
   public SubscriberAPI(final RestClient restClient, final
   Configuration
  configuration) {
      this.restClient = restClient;
      this.configuration = configuration;
   }

   /**
    * looks up a subscriber, by code, and returns it if it exists;
    otherwise, it
    * creates a new default subscriber which can be persisted later
    * @param subscriberCode the subscriber code to look up
    * @return a subscriber with the given code
    */
   public Subscriber getSubscriber(final String subscriberCode) {
      try {
         return new Subscriber(new
JSONObject(restClient.getForObject(getSubscriberURL(subscriberCode),
String.class)));
      } catch (HttpClientErrorException e) {
        if (e.getStatusCode() == HttpStatus.NOT_FOUND) {
          return new Subscriber(subscriberCode);
        }
        throw new RuntimeException(e);
       } catch (JSONException e) {
        throw new RuntimeException(e);
       }
   }

   /**
    * sends the subscriber back to the server. if the subscriber was
    not initially retrieved from
    * the server, it will add it.
    * @param subscriber the subscriber to save
    * @return the subscriber, in the state which was returned by the
    update
    */
   public Subscriber updateSubscriber(final Subscriber subscriber) {
      if (subscriber.isNotPersisted()) {
```

```
        return
toSubscriber(restClient.postForObject(getSubscribersURL(),subscriber.
toString(), String.class));
      } else {
        return
toSubscriber(restClient.putForObject(getSubscriberURL(subscriber.ge
tCode()), subscriber.toString()));
      }
   }

/**
 * removes a subscriber, by code
 * @param subscriberCode the code of the subscriber to delete
 */

public void deleteSubscriber(final String subscriberCode) {
restClient.delete(getSubscriberURL(subscriberCode));
}

private Subscriber toSubscriber(final String json) {
   try {
      return new Subscriber(new JSONObject(json));
   } catch (JSONException e) {
      throw new RuntimeException(e);
      }
   }

   private String getSubscribersURL() {
      return String.format("%s/prime-home/api/current/subscribers",
   configuration.getUrl());
   }

   private String getSubscriberURL(final String subscriberCode) {
      return
   String.format("%s/prime-home/api/current/subscribers/code:%s",
   configuration.getUrl(), subscriberCode);
      }
   }
```

Data Classes

com.clearaccess.subscriber.api.Subscriber

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.subscriber.api;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Subscriber extends JSONObject {
   /**
    * builds this subscriber from the state in the json object.
    * @param json the json representation of the subscriber
    * @throws JSONException from super-constructor
    */
public Subscriber(final JSONObject json) throws JSONException {super
(json.toString());
}

/**
 * creates a new subscriber in an initial state.
 * @param code the code of the new subscriber
 */
public Subscriber(final String code) {
   try {
     put("code", code);
     put("attributes", new JSONObject());
     put("credentials", new JSONObject());
     put("labels", new JSONArray());
     put("subscriptions", new JSONArray());
    } catch (JSONException e) {
     throw new RuntimeException(e);
    }
}

public String getCode() {
   try {
     return getString("code");
    } catch (JSONException e) {
     throw new RuntimeException(e);
    }
  }

  public Address getAddress() {
    final JSONObject attributes = optJSONObject("attributes", new
    JSONObject());
    final String street =
 attributes.optString("Subscriber.Address.1.Street");
    final String city =
 attributes.optString("Subscriber.Address.1.City");
    final String state =
 attributes.optString("Subscriber.Address.1.State");
    final String postalCode =
 attributes.optString("Subscriber.Address.1.PostalCode");
    return new Address(street, city, state, postalCode);
  }

  public void setAddress(final Address address) {
    try {
```

```
        final JSONObject attributes = optJSONObject("attributes",new
        JSONObject());
        attributes.put("Subscriber.Address.1.Street",address.getStreet
        ());
        attributes.put("Subscriber.Address.1.City",address.getCity());
        attributes.put
      ("Subscriber.Address.1.State",
        address.getState());
        attributes.put("Subscriber.Address.1.PostalCode",address.getPo
        stalCode());
        attributes.put("Subscriber.Address.1.Type", "Home");
        put("attributes", attributes);
      } catch (JSONException e) {
        throw new RuntimeException(e);
      }
    }
    public String getFullName() {
       final JSONObject attributes = optJSONObject("attributes", new
       JSONObject());
       return attributes.optString("Subscriber.FullName");
    }
    public void setFullName(final String fullName) {
       try {
          final JSONObject attributes = optJSONObject("attributes",
          new JSONObject());
          attributes.put("Subscriber.FullName", fullName);
          put("attributes", attributes);
        } catch (JSONException e) {
          throw new RuntimeException(e);
        }
    }
    public String getEmailAddress() {
       final JSONObject attributes = optJSONObject("attributes", new
JSONObject());
       return attributes.optString("Subscriber.EmailAddress");
     }

     public void setEmailAddress(final String emailAddress) {
       try {
         final JSONObject attributes = optJSONObject("attributes", new
       JSONObject());
         attributes.put("Subscriber.EmailAddress", emailAddress);
         put("attributes", attributes);
       } catch (JSONException e) {
         throw new RuntimeException(e);
       }
   }

   public String getPhoneNumber() {
       final JSONObject attributes = optJSONObject("attributes", new
JSONObject());
       return attributes.optString("Subscriber.Phone.1.Number");
   }

   public void setPhoneNumber(final String phoneNumber) {
       try {
         final JSONObject attributes = optJSONObject("attributes",
new JSONObject());
         attributes.put("Subscriber.Phone.1.Number", phoneNumber);
         attributes.put("Subscriber.Phone.1.Type", "Home");
         put("attributes", attributes);
       } catch (JSONException e) {
         throw new RuntimeException(e);
       }
    }
```

```
   public Credentials getCredentials() {
      final JSONObject attributes = optJSONObject("credentials", new
JSONObject());
      final String login = attributes.optString("login");
      final String password = attributes.optString("password");
      return new Credentials(login, password);
   }

   public void setCredentials(final Credentials credentials) {
      try {
        final JSONObject json = optJSONObject("credentials", new
  JSONObject());
        json.put("login", credentials.getLogin());
        json.put("password", credentials.getPassword());
        put("credentials", json);
      } catch (JSONException e) {
        throw new RuntimeException(e);
      }
   }

   public String[] getLabels() {
      try {
        final JSONArray json = optJSONArray("labels");
        if (json == null) return new String[0];
        final String[] labels = new String[json.length()];
        for (int i = 0; i < json.length(); i++) {
          labels[i] = json.getJSONObject(i).getString("name");
        }
        return labels;
      } catch (JSONException e) {
        throw new RuntimeException(e);
      }
   }

   public void setLabels(final String[] labels) {
      try {
        final JSONArray json = new JSONArray();
        for (String labelName : labels) {
          final JSONObject label = new JSONObject();
          label.put("name", labelName);
        }
        put("labels", json);
      } catch (JSONException e) {
        throw new RuntimeException(e);
      }
   }

   public boolean isNotPersisted() {
      return !has("revision");
   }

   public JSONObject optJSONObject(final String key, final
JSONObject defaultValue) {
      final JSONObject value = optJSONObject(key);
      return value != null ? value : defaultValue;
   }
}
```

com.clearaccess.subscriber.api.Address

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
```

```
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.subscriber.api;

public class Address {
   private final String
     street,
     city,
     state,
     postalCode;

   public Address(final String street, final String city, final
String state, final String postalCode) {
     this.street = street;
     this.city = city;
     this.state = state;
     this.postalCode = postalCode;
   }

   public String getStreet() {
   return street;
   }

   public String getCity() {
   return city;
   }

   public String getState() {
   return state;
   }

   public String getPostalCode() {
   return postalCode;
   }
}
```

com.clearaccess.subscriber.api.Credentials

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.subscriber.api;

public class Credentials {
   private final String
      login,
      password;

   public Credentials(String login, String password) {
      this.login = login;
      this.password = password;
   }

   public String getLogin() {
      return login;
   }

   public String getPassword() {
      return password;
   }

}
```

Support

com.clearaccess.Configuration

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess;

import org.springframework.stereotype.Component;

@Component
public class Configuration {
   private final String
      url = "http://192.168.25.190:8080",
      user = "admin",
      password = "admin";

   public String getUrl() {
      return url;
   }

   public String getUser() {
      return user;
   }

   public String getPassword() {
      return password;
   }
}
```

com.clearaccess.rest.RestClient

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.rest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.client.RequestCallback;
import org.springframework.web.client.ResponseExtractor;
import org.springframework.http.HttpMethod;
import org.springframework.http.client.ClientHttpRequest;
import org.springframework.http.client.ClientHttpResponse;
import org.apache.commons.io.IOUtils;

import java.io.IOException;

@Component
public class RestClient extends RestTemplate {
   /**
    * spring's RestTemplate does not support authentication, this
    class
    * requires the use of our own http client request factory
    * @param clientHttpRequestFactory our own client request factory
    which can perform
    * authentication.
    */
   @Autowired
   public RestClient(HttpClientFactory clientHttpRequestFactory) {
     super(clientHttpRequestFactory);
   }

   /**
    * spring's RestTemplate does not support a 'put' which can also
    return data.
    * @param uri the uri of the PUT request
    * @param requestData the data to send
    * @return the data which was returned
    */
   public String putForObject(final String uri, final String
   requestData) {
      return execute(uri, HttpMethod.PUT, new RequestCallback() {
        public void doWithRequest(ClientHttpRequest
clientHttpRequest) throws IOException {
clientHttpRequest.getBody().write(requestData.getBytes());
        }
      }, new ResponseExtractor<String>() {
       public String extractData(ClientHttpResponse
clientHttpResponse) throws IOException {
          return new
String(IOUtils.toByteArray(clientHttpResponse.getBody()));
        }
     });
   }
}
```

com.clearaccess.rest.HttpClientFactory

```
/*
 * Copyright (c) 2010 ClearAccess, Inc.
 * This code is provided AS-IS for illustration purposes.
 */

package com.clearaccess.rest;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import
org.springframework.http.client.CommonsClientHttpRequestFactory;
import org.springframework.stereotype.Component;
import org.springframework.beans.factory.annotation.Autowired;
import com.clearaccess.Configuration;

import java.net.URL;
import java.net.MalformedURLException;

@Component
public class HttpClientFactory extends
CommonsClientHttpRequestFactory {
     private Configuration configuration;
     @Autowired
     public HttpClientFactory(Configuration configuration) {
       this.configuration = configuration;
     }

     /**
       * intercepts the super.getHttpClient and modifies its state to
       include
       * authentication credentials
       * @return the http client
       */
     @Override
     public HttpClient getHttpClient() {
       try {
          final HttpClient client = super.getHttpClient();
          final UsernamePasswordCredentials credentials =
            new UsernamePasswordCredentials(configuration.getUser(),
configuration.getPassword());
          final URL url = new URL(configuration.getUrl());
          final AuthScope authScope = new AuthScope(url.getHost(),
url.getPort(), AuthScope.ANY_REALM);
          client.getState().setCredentials(authScope, credentials);
          return client;
       } catch (MalformedURLException e) {
          throw new RuntimeException(e);
     }
   }
}
```

Appendix A: CAQL BNF

BNF

```
<caql>              ::= <doc-clause> <terms> <show-clause> <sort-clause>
                      | <doc-type> <show-clause> <sort-clause>

<doc-clause>        ::= <doc-type> "with"
                      | "" <!-- <doc-clause> is optional -->

<terms>             ::= <term>
                      | <term> <terms> <!-- one or more with implied AND logic -->
                      | <term> and <terms> <!-- one or more with explicit AND logic -->
                      | <term> or <terms> <!-- one or more with OR logic -->

<term>              ::= <all-field-value>
                      | <field-name> ":" <field-value>
                      | <field-name> ":" "in" "(" <literal-list> ")"

<field-value>       ::= <literal>
                      | <range>

<all-field-value>   ::= <text>
                      | <octet> "." <octet> "." <octet> "." <octet> <!-- IP address -->
                      | <range>

<literal>           ::= <text>
                      | <number>
                      | <mm> "/" <dd> "/" <yyyy> <!-- date -->
                      | <octet> "." <octet> "." <octet> "." <octet> <!-- IP address -->

<literal-list>      ::= <literal>
                      | <literal> <literal-list> <!-- one or more -->

<range>             ::= "from" <literal> "to" <literal>
                      | "from" <literal>
                      | "to" <literal>

<show-clause>       ::= "show" <show-terms>
                      | "" <!-- <show-clause> is optional -->

<show-terms>        ::= <show-term>
                      | <show-term> <show-terms> <!-- one or more -->

<show-term>         ::= <field-name>
                      | <field-name> as <field-alias> <!-- optional alias -->

<sort-clause>       ::= "sort" <sort-terms>
                      | "" <!-- <sort-clause> is optional -->

<sort-terms>        ::= <sort-term>
                      | <sort-term> <sort-terms> <!-- one or more -->

<sort-term>         ::= <field-name>
                      | <field-name> <sort-direction>

<sort-direction>    ::= "asc"
                      | "desc"
```