SMART/RG®
forward thinking

501 SE Columbia Shores Boulevard, Suite 500
Vancouver, Washington 98661 USA
+1 360 859 1780 / smartrg.com

/ DEVICE MANAGER BY SMARTRG® Integration Guide

API Integration Guide

Release 1.3

January 2019

SMART/RG

Table of Contents

WELCOME!

Welcome to the Device Manager by SmartRG Integration Guide.

SmartRG proudly brings you the best, most innovative broadband solutions available. SmartRG enables service providers to monitor, manage, and monetize the connected home through the design and production of reliable and highly interoperable solutions with our hardware and software platforms providing seamless management.

As an early innovator in TR-069 remote management technology, SmartRG offers the finest in managed broadband and home networking solutions. Our products leverage various broadband access technologies and are outfitted with highly customizable software, meeting diverse service provider requirements. Based in the USA, SmartRG provides local, proactive software development and customer support. In the rapidly evolving broadband market, SmartRG helps service providers keep their businesses on the cutting edge through its laser-focused product lines, leveraging the very latest in broadband access and home networking technologies. SmartRG solutions enable service providers to improve their bottom line by reducing service costs and increasing customer satisfaction. Learn more at www.SmartRG.com.

Intended Audience

This guide is for anyone who wants to become familiar with integrating external systems with Device Manager by SmartRG. Most of this document specifically targets integration implementers.

Getting Assistance

Service providers: if you require help with this product, please open a SmartRG support request.

In This Guide…

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

Device Manager API Capabilities

Device Manager by SmartRG supports APIs for device and subscriber management, data searching, and control session creation.

Device Management:

- Create, edit, and delete subscribers, devices, and associated data
- Associate subscribers with managed devices
- Configure managed devices (e.g., port bridging, wireless settings, etc.)

- Turn specific services on or off (e.g., 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

- Support subscriber single sign-on implementations

Getting Started

Language / Framework Requirements

APIs are entirely web-based and language agnostic. This means you are free to choose whatever language or framework you are most comfortable with. This guide provides suggestions and examples but you are by no means limited to using them. Before proceeding:

- Choose HTTP client software to interact with the APIs. 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. Some good choices include:
  - TextMate
  - E Text Editor
  - jEdit
  - notepad++

For the examples in this guide, cURL (http://curl.haxx.se/docs/manpage.html) and a text editor were used.

Searching the API

In Device Manager version 5.2, CAQL (ClearAccess Query Language) is used with the search web service to find objects to manipulate with other web service endpoints.

Note: The properties returned for each item don't necessarily map directly to the search query. For example, a query on "subscriberCode" returns as "code".

Devices and Subscribers

Devices and subscribers are the primary objects managed by Device Manager. A device is identified by an OUI and Serial Number. While many devices have unique MAC addresses and some devices have a serial number that is the same as their MAC, the MAC is *not used* for identification purposes. It is critical to keep this in mind when integrating with Device Manager.

When creating devices 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 and 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 And Drivers

Device Manager executes applications against devices to apply configuration. An application is a cohesive set of device functionality and configuration, for example, Time Blocking or Wireless.

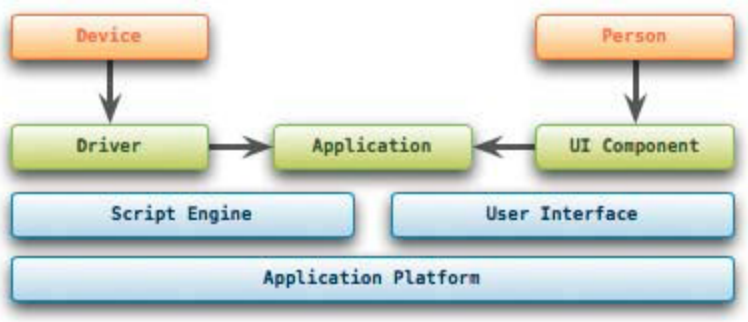

Figure 1: Applications

Device Manager applications may include the following:

- Name & ID
- Data schemas used for all devices in a system. (Typically called "Globals")
- Data schemas used for individual devices. (Typically called "Settings")
- Scripting engine APIs.

These applications are synchronized with a device during sessions with the CWMP server when the applications 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 devices need varying kinds of configuration, or may not support the application at all.

More information about applications is available elsewhere in this Integration Guide.

Device Lifecycle

Introduction

The device lifecycle consists of 4 phases: Provisioning, Activation, Management, and Termination.

Figure 2: Device Lifecycle

Provisioning

Provisioning is the process of configuring a device for use by a subscriber. There are two kinds of devices to provision:

- Future device: A placeholder configuration for a device that has never contacted Device Manager.
- Managed device: A device that has previously contacted Device Manager.

During the Provisioning phase:

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

Activation

Activation occurs the next time (after the Provisioning phase) that a device contacts the ACS.

Activation can occur on both future and managed devices that are associated with a 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

Management

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

- Support personnel interact with applications to troubleshoot problems and modify services.
- Subscribers can change their device'sconfiguration 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 assigned subscriber. This can happen for a number of reasons, such as:

- Customer churn
- Device failure
- Device upgrade

Device Manager can handle this situation in a number of ways:

- The managed device can remain active in the system, but become unassociated with the subscriber.
- The managed device can be deleted from the system.
- A new device can replace an old device (as an RMA).

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.

Anatomy of a REST Call

Request

An HTTP request is ALWAYS in the form:

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

The first line indicates 3 things: the method (VERB), the URI of the resource, and the HTTP version. These things always appear as the first line of any HTTP request and are always followed by request headers.

The headers are described in the following table.

Header	Description
Host: people.mydomain.com	This header is always required when using HTTP version 1.1 and indicates which host or virtual host from which to request the resource.
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 preference. In this example, the client primarily wants JSON formatted data. If that is not available, then XML is acceptable. If neither of those are available, send the data in any format. For detailed information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html.
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.

The headers and body are separated by a single empty line. The body contains a representation of the resource being sent to the server. Its format is dictated by the value of the Content-Typeheader.

Below is a sample 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 (POST)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".

Response

An HTTP response is ALWAYS in the format:

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

Below is a sample Response.

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

This response says: "Your request was successful and here is your result in JSON format."

"HTTP/1.1 200 OK" indicates that the response uses the HTTP 1.1 protocol, and that the request was successful.

The following headers are included:

- 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.

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	Updates 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

Resource Types

A resource generally comes in one of 3 types: a single item, a list of items, or a process. Depending on the type of resource, the semantics of the supported methods can vary slightly.

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, single items can be retrieved, modified and sent back to the server to update the resource's state.

Lists

A list resource can act as a container for other resources which can be filtered, searched, sorted and paginated. New resources can be appended to a list resource using the POST method.

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.

JSON

JavaScript Object Notation (JSON) is the data format primarily used by Device Manager 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".

For moreinformation about this format, go to: http://json.org/.

API Standards

All APIs found under the /api URI will follow the standards outlined in this chapter.

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 (or table of contents), e.g., .../api/v1/subscribers.

Documents

Document services will behave as follows:

- **GET** returns the full representation of the resource.
- **DELETE** removes 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 is returned, and the state of the resource is *not* changed.
- Data retrieved from a GET request can be sent directly with a PUT request if there are no changes made to the data.

Most documents have some common fields as described below.

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

Creating a New Document

To create a new document, such as a device:

1. GET the template of the type of document you wish to create such as http://host/prime-home/api/v1/templates/device.
2. Make the necessary changes to the template using an appropriate application.
3. POST the document to the table of contents of that document type such as: http://host/prime-home/api/v1/devices .

Modifying a Document

To make changes to a document:

1. GET the resource.
2. Make the necessary changes to the document using an appropriate application.
3. PUT the document back to the same resource.
4. To view the status of asynchronous side effects:
    a. If there were any asynchronous side-effects, the PUT response contains headers that include a URL that you can poll for status.
    b. Poll the status URI until it indicates completion.

Deleting a Document

DELETE the URLof the document you wish to remove. e.g., `http://host/prime-home/api/v1/subscribers/code:123456789`.

WADL

To include generated HTML descriptions of requested resources, append `?wadl` to any resource URI. The description will includ the supported methods, data types, references to sub-ordinate resources, and security parameters for the resource.

Error Status Codes

The most common error status codes are described below. For detailed information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

400 type (Bad Request) statuses are returned in the following format: `400 ERROR-CODE; Error Text`.

STATUS	CODE	MEANING	CAUSED BY
200		OK	The request was handled successfully.
400		Bad Request	Caused by the client providing bad information to the server.
	SYNTAX-ERROR		The structure of the provided document is invalid.
	VALIDATION-ERROR		Some value provided in the document is invalid.
	CONCURRENCY-ERROR		An out-of-date revision was provided.
	REFERENCED-ENTITY-NOT-FOUND		Data was provided which references something that is invalid. e.g., a service that does not exist.
401		Unauthorized	Credentials were not 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.
500	Internal Server Error		Internal error occurring on the server. Contact Support.

DTO Properties

Device Manager DTOs are dynamic, can change independently of the service, and are versioned via application schemas. You can retrieve application schemas using the `/prime-home/api/v1/applications/schemas/` endpoint.

Dates

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

Search and CAQL

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

- Search for devices, subscribers, and other data within the Portal web application
- Select devices for bulk operations
- Request subscriber and device reports and export the data in CSV format

NOTE: For subscriber and device reports, you do not have to enter CAQL code manually. A user interface is provided that allows the user to create a report using drag-and-drop and by entering simple criteria.

About Search

To understand how to construct CAQL queries for running against the search engine, it is useful to understand how the search engine works.

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

Each document contains fields. The "subscriber" document contains fields which define the various properties for the subscriber such as "name", and "address". The "device" document contains fields which define the properties, characteristics, and data collected from the device such as "serialNumber", "informCount", and "downstreamAttenuation".

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".

A complete list of available documents and fields is documented in the "Search and CAQL" section.

CAQL searches can be constructed to be as precise as desired. For example, you might search for:

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

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" making sure that the “A” in “Attenuation” is capitalized.

Values are case *in*sensitive. For instance, searching for "Jane" returns all of the following: "Jane", "JANE", and "jane".

CAQL Tutorial

This tutorial explains terms, document types, wildcard usage, clauses, phrases, ranged searches, precedence and grouping, the NOT operator, fields and field types, sorting and zero-term queries.

Terms

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

juniper

Often, you will want to restrict a search so that it only matches a certain field. The following query demonstrates searching for any documents where the "name" field contains "juniper":

name: juniper

Many fields are aliased so you can reference them using alternate names. These aliases are provided for convenience and to improve CAQL readability. For example, the "subscriber" document contains a "subscriberCode" field which may be referenced as "code" A complete list of fields and aliases is outside the scope of this document.

Document Types

By default, search terms are matched against all document types (subscriber, device, user, etc.). You can restrict a search to a specified document type using the "with" clause. The with clause is inserted before the search terms. The following query searches for subscribers with any field containing "juniper".

subscribers with juniper

NOTE: Aliases are available for most document types. For example, the "subscriber" document type can be referenced as "subscribers", and the "device" document type can be referenced as "devices". You can view a complete list of document types and aliases in the Device Manager application. Click on the Utilities Menu and select Search from the left sidebar.

To search all document types in a query, you can specify the special document types of "anything" and "everything". Specifying "anything" returns a limited number of matches; specifying "everything" returns every match found. For example, the following query searches all documents for any field containing "juniper":

anything with juniper

Wildcards

The supported wildcards include asterisks (*) and question marks (?). The following query searches for devices with a serial number starting with "ABC":

devices with sn:ABC*

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

devices with sn:ABC????

You can use wildcards 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

You can use wildcards as the leading character. The following query finds devices with serial numbers ending in "XYZ":

devices with sn:*XYZ

You can use wildcards to search IP addresses. The following query finds devices with a WAN IP address that contains "192.1":

devices with wanip: 192.1.*.*

NOTE: Using wildcards before and within search terms does not make optimal use of the search engine and requires more system resources than trailing wildcard searches. Trailing wildcard queries can fail if the wildcard matches too many terms. It is recommended that you use inner and leading wildcards for manually entered searches only (e.g., on the Find Devices page of the Portal web application), and not use them in automated and reoccurring processes (e.g., bulk operations).

Combining Terms

Multiple terms can be combined in a single search. By default, terms are combined using AND logic. The following 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*

You can combine terms using OR logic by using the "or" keyword. The following query searches for devices with serial numbers starting with "1000" or "2000":

devices with sn:1000* or sn:2000*

In Clause

You can search for one of multiple choices by using the "in" keyword. The following query searches for subscribers with names "sam" or "juni":

subscribers with name in (sam juni)

The parentheses replace the "or" keyword. The following query returns the same matches but in this case the "or" keyword is specified:

sam or name: juni

Phrases

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

subscribers with name: "juniper jones"

Ranged Searches

CAQL supports ranged searches. The following query searches for devices with an inform count between 1 and 100:

devices with informCount: from 1 to 100

Open-ended ranges are also supported. The following query searches for devices with an inform count of at least 100:

devices with informCount: from 100

The following query searches for devices with an inform count equal or less than 10:

devices with informCount: to 10

Precedence and Grouping

The "and" keyword binds more tightly than the "or" keyword. You can use these keywords to prioritize the search criteria. The following query returns documents that have a serial number that starts with "1000" ***and*** also have any field containing "sam" and also returns any documents that do *not* have a serial number that starts with "1000" *but* do have any field containing "juni".

serialNumber:1000* and sam or juni

To return different results, you can insert parentheses to group criteria. The following query returns documents that have a serial number that starts with "1000" ***and*** contain any fields with "sam" *or* "juni".:

serialNumber:1000* and (sam or juni)

Not Keyword

To find documents that do *not* include a search term, use the "not" keyword. The following query returns documents that do not contain "sam" in any field:

not sam

The following query returns subscribers that do not contain "sam" in the "name" field:

subscribers with name: not sam

You can use the "not" keyword with the "in" keyword as well. The following query returns any subscribers that do not contain "sam" or "juni" in the name field:

subscribers with name: not in (sam juni)

Field Types

CAQL understands several types of data used in the ACS:

- Number
- Date
- IP address
- Text

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 are interpreted only as text or IP addresses.

Queries that are restricted to search a specific field will parse those values according to their format. To force parsing of a search value as a string for matching against numbers in text fields (such as serial numbers), you must enclose the value in quotes. The format of each data type is explained in this section.

Number

Numbers can be expressed with 1 or more decimal characters ("0" - "9") and with or without decimal points. The following query searches for devices with a “downstreamAttenuation” between 1.0 and 1.5:

devices with downstreamAttenuation: from 1.0 to 1.5

Date

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 Address

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.*

Text

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, you must enclose the search value with quotes. For example, if a serial number (text field) 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, enclose the valuein quotes as shown below:

device with serialNumber: "12345"

Showing Fields

Query results include a default set of fields. To request that a specific list of fields be included in the search results output, you can use a "show" clause to specify the fields that you want included in the output.. The following query returns the serial number and last inform date for every device with OUI AA1122:

devices with oui: AA1122 show serialNumber lastInform

To provide more readable output from a query, you can specify aliases for the returned fields using the "as" keyword. The following query relabels th "serialNumber" field as "Serial Number" field and the "lastInform" field as "Last Inform Date":

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

Sorting

You can sort search results using "sort" clauses. The following query finds devices with OUI AA1122, sorted by the last inform date:

devices with oui: AA1122 sort lastInform

You can control the sort order using the "asc" and "desc" modifiers. By default, sort is in ascending order. 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

You can specify multiple sorts. The following query finds devices with OUI AA1122, sorted by the last inform date, with the most recent informs at the top, and 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, specify the document type. The following query finds all devices:

devices

The following query finds all devices, showing the serial number, sorted 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 CAQLstring	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.

For more information, see the CAQL Tutorial chapter.

Device APIs

Update List of Devices Connected to ACS

Use Case: Add a new device to the list of devices connected to the ACS.

URL: http://host/prime-home/portal/devices/

Method: PUT

Request

```
<request-data>
{
manufacturer: (string),
oui: (string),
productClass: (string),
serial: (string),
provisioningCode: (string),
rootName: (string)
}
</request-data>
```

Response

```
<response-data>
{
id: (number),
manufacturer: (string),
oui: (string),
productClass: (string),
provisioningCode: (string),
serial: (string)
}
</response-data>
```

GET List of Device Types Defined in ACS

Use Case: Retrieves the list of device types defined in the ACS. A device type consists of a product class and a hardware version, which is used matched against devices checking into the ACS, and other information which is used for informational purposes. Four optional query string parameters may be passed to narrow the results: hardwareVersion, productClass, modelName, oui.

URL: http://host/prime-home/portal/devices/types

Method: GET

Query

<query>
first=(number) // the first result to return
count=(number) // the number of results to return
</query>

Response

<response-data>
[{
id: (number),
friendlyName: (string),
friendlyManufacturerName: (string)
}]
</response-data>

Create Device Type

Use Case: Add a new device type

URL: http://host/prime-home/portal/devices/types

Method: PUT

Request

<request-data>
{
friendlyName: (string),
friendlyManufacturerName: (string),
hardwareVersion: (string),
productClass: (string),
modelName: (string),
imageURL: (string),

wanInterfaceType: (string)
}
</request-data>

Response

<response-data>
{
id: (number),
friendlyName: (string),
hardwareVersion: (string),
productClass: (string),
modelName: (string),
imageURL: (string),
wanInterfaceType: (string)
}
</response-data>

GET List of Device Types Checking in to ACS

Use Case: Retrieves a list of a single device type that are checking into the ACS. A device type consists of a product class and a hardware version.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}

Method: GET

Response

<response-data>
{
id: (number),
friendlyName: (string),
friendlyManufacturerName: (string),
hardwareVersion: (string),
productClass: (string),
modelName: (string),
imageURL: (string),
wanInterfaceType: (string)
}
</response-data>

Modify Device Type

Use Case: Updates a single device type

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}

Method: POST

Request

<request-data>
{
friendlyName: (string),
friendlyManufacturerName: (string),
hardwareVersion: (string),
productClass: (string),
modelName: (string),
imageURL: (string),
wanInterfaceType: (string)
}

</request-data>

Response

<response-data>
{
id: (number),
friendlyName: (string),
friendlyManufacturerName: (string),
hardwareVersion: (string),
productClass: (string),
modelName: (string),
imageURL: (string),
wanInterfaceType: (string)
}
</response-data>

Remove Device Type

Use Case: Removes a single device type. If there are any dependencies on this device type, an error is returned.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}

Method: DELETE

Response

N/A

GET Device Statistics by Manufacturer

Use Case: Retrieves the counts of different devices by manufacturer.

URL: http://host/prime-home/portal/stats/manufacturers

Method: GET

Response

<response-data>
{
countsPerManufacturer: [{
manufacturer: string,
count: number
}],
lastInforms: [{

id: number,
labels: [string, string],
manufacturer: string,
productClass: string,
OUI: string,
serialNumber: string
rootName: string,
firstInform: timestamp,
lastInformDelta: number,
lastInform: timestamp,
informCount: number,
deviceTypeId: number,
friendlyManufacturerName: string,
friendlyName: string,
imageUrl: string,
wanIpAddress: string,
capabilities: [string, string]
}]
}
</response-data>

GET Counts of Device Classes by Manufacturer

Use Case: Retrieves the counts of different device product classes by manufacturer.

URL: http://host/prime-home/portal/stats/manufacturers/{manufacturer}

Method: GET

Response

<response-data>
{
countsPerProductClass: [{
productClass: string,
count: number
}],
lastInforms: [{
id: number,
labels: [string, string],
manufacturer: string,
productClass: string,
OUI: string,
serialNumber: string

rootName: string,
firstInform: timestamp,
lastInformDelta: number,
lastInform: timestamp,
informCount: number,
deviceTypeId: number,
friendlyManufacturerName: string,
friendlyName: string,
imageUrl: string,
wanIpAddress: string,
capabilities: [string, string]
}]
}
</response-data>

GET Daily Device Counts for 7 Days

Use Case: Retrieves the number of new devices seen each day, over the last 7 days.

URL: http://host/prime-home/portal/stats/recentActivity/newDevices

Method: GET

Response

<response-data>
[{
date: string("yyyy-mm-dd"),
count: number
}]
</response-data>

GET Count of Devices Checked In for 7 Days

Use Case: Retrieves the number of devices that have checked in each day, over the last 7 days.

URL: http://host/prime-home/portal/stats/recentActivity/devicesInforming

Method: GET

Response

<response-data>
[{
date: string("yyyy-mm-dd"),

count: number
}]
</response-data>

GET Count of Devices by Device Type

Use Case: Retrieves the number of devices by device type, ordered descending by count.

URL: http://host/prime-home/portal/stats/deviceTypes

Method: GET

Query

<query>
first=(number) // the first result to return
count=(number) // the number of results to return
days=(number) // only count devices that have informed within these last n days
</query>

Response

<response-data>
[{
friendlyManufacturerName: string,
friendlyName: string,
count: number
}]
</response-data>

GET Count of Subscribers by Label

Use Case: Retrieves the number of subscribers by label, ordered descending by count.

URL: http://host/prime-home/portal/stats/labels/subscriberCounts

Method: GET

Query

<query>
first=(number) // the first result to return
count=(number) // the number of results to return
</query>

Response

<response-data>
[{
label: string,
fgColor: "#rgb",
bgColor: "#rgb",
count: number
}]
</response-data>

GET Device Solicit Status

Use Case: Gets the solicit status.

URL: http://host/prime-home/portal/solicits

Method: GET

Query

<query>
deviceIds:
A comma separated list of device ids to retrieve. Default returns all devices.
</query>

Response

<response-data>
[{
id: (number),
status: (string: UNKNOWN|PENDING|SUCCESS|
DECLINED|BEST_ATTEMPT|UNREACHABLE),
lastStatus: (timestamp),
lastSuccess: (timestamp),
lastBestAttempt: (timestamp)
}]
</response-data>

Update Device Status

Use Case: Queues devices to update their status and causes and scheduled scripts to be run against the device. Passed data is an array of objects, each must contain ID and lastStatus. The value for lastStatus will be populated in the return value. When querying for related status, lastStatus will contain a value greater or equal to this value to indicate that the requested solicit has been completed.

URL: http://host/prime-home/portal/solicits/{deviceId}

Method: PUT

Request

```
<request-data>
[{
id: (number of the device id),
}]
</request-data>
```

Response

```
<response-data>
[{
id: (number),
lastStatus: (number)
}]
</response-data>
```

GET Firmware Resource List by Device Type

Use Case: List the firmware compatible with the specified device type, ordered by release date. Prior version IDs refer to other firmware instances that this firmware may replace without losing settings.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware

Method: GET

Query

```
<query>
first=(number) // the first result to return
count=(number) // the number of results to return
deviceType=(number) // filter by deviceTypeId
</query>
```

Response

<response-data>
[{
id: (number),
deviceTypeId: (number),
softwareVersion: (string),
releaseDate: (string),
username: (string),
password: (string)
}]
</response-data>

Add Firmware Resource by Device Type

Use Case: Add new firmware to all devices of the specified type.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware

Method: PUT

Request

<request-data>
{
softwareVersion: (string),
binaryURL: (string),
releaseDate: (string)
}
</request-data>

Response

<response-data>
{
id: (number),
deviceTypeId: (number),
softwareVersion: (string),
binaryURL: (string),
releaseDate: (string)
}
</response-data>

GET Firmware Description

Use Case: Retrieves a single firmware description for the specified device type. The description consists of information such as its upgrade path and the publicly accessible URL where the firmware's image resides.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware/{firmwareId}

Method: GET

Response

<response-data>
{
id: (number),
deviceTypeId: (number),
softwareVersion: (string),
binaryURL: (string),
releaseDate: (string)
}
</response-data>

Remove Firmware Description

Use Case: Removes a single firmware description.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware/{firmwareId}

Method: DELETE

Response

N/A

Modify Firmware Description

Use Case: Updates a single firmware description.

Note: While it is possible to change the device type ID, a subsequent update to the same URL will result in a 404 error because the URL also includes the deviceTypeId.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware/{firmwareId}

Method: POST

Request

```
<request-data>
{
deviceTypeId: (number),
softwareVersion: (string),
binaryURL: (string),
releaseDate: (string)
}
</request-data>
```

Response

```
<response-data>
{
id: (number),
deviceTypeId: (number),
softwareVersion: (string),
binaryURL: (string),
releaseDate: (string)
}
</response-data>
```

GET Firmware Resource Labels by ID for Device Type

Use Case: Retrieves the labels associated to the firmware resource specified by the device type and the firmware resource ID.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware/{firmwareId}/labels

Method: GET

Response

```
<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}
</response-data>
```

Modify Firmware Labels

Use Case: Updates the labels for a firmware.

URL: http://host/prime-home/portal/devices/types/{deviceTypeId}/firmware/{firmwareId}/labels

Method: POST

Request

<request-data>["labelName",...]</request-data>

Response

<response-data>
[{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}]
</response-data>

GET Count of Devices in ACS by Device Type

Use Case: Retrieves a count of the devices are in the system, broken out by device type.

URL: http://host/prime-home/portal/devices/summary

Method: GET

Response

<response-data>
{
totalDevices: number,
totalsByType: {
"type 1": number,
"type 2": number
}
}
</response-data>

GET Device Details

Use Case: Gets basic device information for a device including its identification, inform information and device type.

URL: http://host/prime-home/portal/devices/{id}

Method: GET

Response

<response-data>
{
id: number,
labels: [string, string],
manufacturer: string,
productClass: string,
OUI: string,
serialNumber: string
rootName: string,
excludeFromBulkOperation: boolean
firstInform: timestamp,
lastInformDelta: number,
lastInform: timestamp,
informCount: number,
deviceTypeId: number,
wanInterfaceType: string,
friendlyManufacturerName: string,
friendlyName: string,
imageUrl: string,
wanIpAddress: [string, string],
capabilities: [string, string],
managementGroupId: number (optional),
subscriberId: number (optional),
deviceOperations: [{
id: number,
bulkOperation: {
id: number,
name: string,
start: timestamp,
firmware: { (optional)
version: string
}
},

```
status: {
state: string (PENDING|ACTIVE|COMPLETE),
completeState: string (SUCCESS|FAILURE|CANCELLED),
failureReason: string (optional)
}
}]
}
</response-data>
```

Get Device Template

Use Case: Get empty JSON template to populate for creating a device

URL: http://host/prime-home/api/v1/templates/device

Method: GET

Request

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

Response

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

Create Device

Use Case: Creates a new future device.

URL: http://host/prime-home/api/v1/devices

Method: POST

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",
  }
}
```

DELETE Device

Use Case: Delete the device. This will fail if there is an associated subscriber.

URL: http://host/prime-home/api/v1/devices/deviceId

Method: DELETE

Response

N/A

Lock Device

Use Case: Lock a device for web service writes.

URL: http://host/prime-home/portal/devices/{id}/lockDevice

Method: POST

GET Device Attribute List

Use Case: Gets a list of device attributes represent the tree of data provided from the device. See the TR-098 for information on the InternetGatewayDevice data model or another Technical Report depending on the device's capabilities. A filter may be applied in the query string to restrict results.

URL: http://host/prime-home/portal/devices/{id}/attributes/

Method: GET

Query

<query>
recursive: true/false (default: false)
- whether to recurse through sub objects

withValues: true/false (default: true)
- whether to return object properties
withLeaves: true/false (default: true)
- whether to return leaf objects. ignored if
withValues = true
attributeDate: time (timestamp)
- to return values as of date
filter:
by simple string:
InternetGatewayDevice.ManagementServer
or
InternetGatewayDevice.LANDevice.*
by JSON object:

```
{
InternetGatewayDevice: {
DeviceInfo: { SoftwareVersion: '*' },
WANDevice: {
'*': {
WANCommonInterfaceConfig: { WANAccessType: '*' },
WANConnectionDevice: {
'*': {
WANDSLLinkConfig: '*',
WANEthernetLinkConfig: '*',
WANIPConnection: '*',
WANPPPConnection: '*',
}
}
}
}
}
}
```

'*' can be used to denote 'all indices of an array like object', 'the value of', 'all attributes of the object'
</query>

Response

<response-data>
A JSON tree of attributes that may be addressed like:
InternetGatewayDevice.DeviceInfo.SoftwareVersion
</response-data>

Modify Device Attributes

Use Case: Creates an attribute update action which sets the attributes to the specified values and adds it to the device's pending action list.

URL: http://host/prime-home/portal/devices/{id}/attributes/

Method: POST

Request

```
<query>
reboot: true/false (default: false)
- reboot the device after the operation (PUT, DEL)
</query>
<request-data>
{
"a.b.c.d": "a val",
"a.b.c.e": "b val"
}
</request-data>
```

Response

```
<response-data>{scriptRunId: number}</response-data>
```

Create Device Attributes

Use Case: Creates an attribute create action which creates new objects under the 'parent' attribute and adds it to the device's pending action list.

URL: http://host/prime-home/portal/devices/{id}/attributes/

Method: PUT

Request

```
<query>
reboot: true/false (default: false)
- reboot the device after the operation (PUT, DEL)
</query>
<request-data>
{
"parent": "a.b.c",
"attributes": {
```

```
"d": "d val",
"e": "e val"
}
}
</request-data>
```

Response

```
<response-data>{scriptRunId: number}</response-data>
```

Remove Device Attributes

Use Case: Creates an attribute delete action which deletes the given objects and adds it to the device's pending action list.

URL: http://host/prime-home/portal/devices/{id}/attributes/

Method: DELETE

Request Device Attribute Refresh

Use Case: Creates an action which will cause the device to send values for the requested attributes.

URL: http://host/prime-home/portal/devices/{id}/attributes/sync

Method: PUT

Request

```
<request-data>
{
InternetGatewayDevice: {
DeviceInfo: { SoftwareVersion: '*' },
WANDevice: {
'*': {
WANCommonInterfaceConfig: { WANAccessType: '*' },
WANConnectionDevice: {
'*': {
WANDSLLinkConfig: '*',
WANEthernetLinkConfig: '*',
WANIPConnection: '*',
WANPPPConnection: '*',
}
}
}
}
```

```
}
}
'*' can be used to denote 'all indices of an array like object', 'the value of', 'all attribute of the object'
query parameters: depth=N, sets the depth of how far down the tree to sync. optional. defaults to '2'
</request-data>
```

Response

```
<response-data>
{
scriptName: 'name',
scriptRunId: number
}
</response-data>
```

GET Device New Action List

Use Case: Retrieves a list of recent actions added to this device. The following query parameter may be specified:

- since: A timestamp represented as the number of milliseconds since 1/1/1970 UTC. Records returned will be queued on or before this timestamp. Default is two hours ago.

URL: http://host/prime-home/portal/devices/{id}/actions

Method: GET

Query

```
<query>
id
since=(timestamp)|"auto"
optional. returns only actions which have been added since the specified timestamp. If this value
is "auto", actions added after two hours ago, or the time the oldest action pending completion was
queued, whichever is older.
</query>
```

Response

```
<response-data>
[{
id: (number),
queued: (number),
runAfter: (timestamp),
queuedDelta: (number),
runAfterDelta: (number),
actor: (string),
```

actorName: (string),
processed: (boolean),
state: (string: NEXT_SESSION|SCHEDULED|RUNNING|
COMPLETED|INTERNAL_ERROR|SYNTAX_OR_EVAL_ERROR|
ABORT|NO_RESPONSE_FROM_DEVICE|DEVICE_FAULT|DOWNLOAD_FAILURE|
SUBSCRIBER_FAULT|CANCEL),
errorMessage: (string),
scriptId: (number)
}]
</response-data>

Schedule Device Action for Next Contact

Use API to trigger a device to synchronize specific applications

ACS API Endpoint: http(s)://<ACS URL>/prime-home/api/v1/devices/<Device ID #>/actions

Example: http(s)://qadm10.smartrg.com/prime-home/api/v1/devices/12321/actions

Use Case

The 'actions' API endpoint can be used to mark an application or script for synchronization with the device in question. This can be used by third party applications to pull required data into the ACS for further analysis, or it can be use by third party applications to push modified device data down to the device in question. (Only applications are shown in the example)

NOTE: All API calls are asynchronous, meaning that they will return their status code regardless of whether or not the device has been solicited and data has been collected from or set to the device in question.

Summarized Steps

1. Perform a GET on the URL above to retrieve the devices actions
2. Modify the JSON data so that the intended application has the relevant settings below
    a. Application that can PUSH and PULL (WiFi, Port Forwards, etc.)
        i. PULL: "needsInitialization": true
    b. Application that can ONLY PUSH or PULL (LAN Hosts, Network Time, etc.)
        i. "pendingSync": true
3. If the device needs to be actively solicited
    a. Modify the JSON by adding the following entry in the root of the JSON structure
        i. "solicit": true
        ii. See <Some Example>
4. Perform a PUT to the same API endpoint with the modified JSON

Detailed Steps Explained

Step 1: GET the 'actions' end point to retrieve the JSON data for the desired device

HTTP Request Command: GET

JSON Response

```
{
    "solicitableStatus": "OK",
    "revision": -1104981854,
    "scripts": [
    ......
],
"services": {
......
},
"refreshScript": [
......
],
"applications": {
    "Hosts": { // PULL Only Application
        "pendingSync": false,
        "dataOwner": "DEVICE"
    },
    "Time": { // PUSH Only Application
        "pendingSync": false,
        "dataOwner": "SERVER"
    },
    "WIFI": { // PUSH and PULL Application
        "needsInitialization": false,
        "pendingSync": false,
        "dataOwner": "SERVER"
    },
    "PF": { // PUSH and PULL Application
        "needsInitialization": false,
        "pendingSync": false,
        "dataOwner": "SERVER"
    },
    ......
  }
}
```

Step 2: Modify Response JSON to fit desired application behavior

```
{
    "solicitableStatus": "OK",
    "revision": -1104981854,
    "scripts": [
    ......
],
    "services": {
    ......
},
"refreshScript": [
......
],
    "applications": {
        "Hosts": { // PULL Only Application
            "pendingSync": true, // Set to true to PULL Hosts
            "dataOwner": "DEVICE"
        },
        "Time": { // PUSH Only Application
            "pendingSync": true, // Set to true to PUSH Time
            "dataOwner": "SERVER"
        },
        "WIFI": { // PUSH and PULL Application
            "needsInitialization": true, // Set to true to PULL WiFi
            "pendingSync": true, // Set to true to mark for synchronization
            "dataOwner": "SERVER"
        },
        "PF": { // PUSH and PULL Application
            "needsInitialization": false,
            "pendingSync": true,  // Set to true to PUSH Port Forwards
            "dataOwner": "SERVER"
        },
        ......
    }
    “Solicit”: true  // OPTIONAL Set to true to solicit the device.
}
```

Step 3: PUT the modified JSON back to the 'actions' endpoint

HTTP Request Command: PUT

Modified JSON:

```
{
      "solicitableStatus": "OK",
      "revision": -1104981854,
      "scripts": [
      ......
      ],
      "services": {
      ......
      },
      "refreshScript": [
      ......
],
"applications": {
      "Hosts": {  // PULL Only Application
            "pendingSync": true,
            "dataOwner": "DEVICE"
      },
      "Time": {  // PUSH Only Application
            "pendingSync": true,
            "dataOwner": "SERVER"
      },
      "WIFI": { // PUSH and PULL Application
            "needsInitialization": true,
            "pendingSync": false,
            "dataOwner": "SERVER"
      },
      "PF": { // PUSH and PULL Application
            "needsInitialization": false,
            "pendingSync": false,
            "dataOwner": "SERVER"
      },
      ......
   }
   "Solicit": true
}
```

Use API to check if applications have synchronized

ACS API Endpoint: http(s)://<ACS URL>/prime-home/api/v1/devices/<Device ID #>/actions/status

Example: http(s)://qadm10.smartrg.com/prime-home/api/v1/devices/12321/actions/status

Summarized Steps (using Synchronization above)

1. Build URL that contains the 'since' time and the apps that have been set to sync.
    a. since=<Time in MS>
        i. This is the time at which the ACS will check each for the status of each application
    b. pendingSyncs=<Application Code>
        i. This parameter must be entered for each application that is expected to synchronize
    c. Example
        i. actions/status?since=1547501412000&pendingSyncs=WANStatus&pendingSyncs=Hosts

HTTP Request Command: GET

Example For Above Solicit:
http(s)://qadm10.smartrg.com/prime-home/api/v1/devices/12321/actions/status?since=1547501412000&pendingSyncs=Hosts&pendingSyncs=Time&pendingSyncs=WiFi&pendingSyncs=PF

HTTP Response


```
{
    "queuedRuns": [],
    "syncApplications": [
        {
            "complete": true,
            "appCode": "Hosts"
        },
        {
            "complete": false,
            "appCode": "Time"
        },
        {
            "complete": false,
            "appCode": "WiFi"
        },
        {
            "complete": false,
            "appCode": "PF"
        }
    ],
```

```
    "updatedServices": [],
    "completed": false
}
```

GET Queued Action Detail

Use Case: Retrieves the detail for a single action which has been added to a device.

URL: http://host/prime-home/portal/devices/{id}/actions/{scriptRunId}

Method: GET

Response

```
<response-data>
{
id: (number),
queued: (number),
actor: (string),
actorName: (string),
state: (string: NEXT_SESSION|SCHEDULED|RUNNING|
COMPLETED|INTERNAL_ERROR|SYNTAX_OR_EVAL_ERROR|
ABORT|NO_RESPONSE_FROM_DEVICE|DEVICE_FAULT|
SUBSCRIBER_FAULT|CANCEL),
errorMessage: (string),
scriptId: (number),
params:[{
name: (string),
value: (string)
}]
}
</response-data>
```

Modify Script ID and Parameters for Queued Action

Use Case: Updates the script ID and parameters for a single queued action. Only scripts created by the current user and are "SCHEDULED" can be modified.

URL: http://host/prime-home/portal/devices/{id}/actions/{scriptRunId}

Method: POST

Request

```
<request-data>
{
params: [{
name: (string),
```

value: (string)
}]
}
</request-data>

Response

{
id: (number),
queued: (number),
actor: (string),
actorName: (string),
state: (string: NEXT_SESSION|SCHEDULED|RUNNING|
COMPLETED|INTERNAL_ERROR|SYNTAX_OR_EVAL_ERROR|
ABORT|NO_RESPONSE_FROM_DEVICE|DEVICE_FAULT|
SUBSCRIBER_FAULT|CANCEL),
errorMessage: (string),
scriptId: (number),
params:[{
name: (string),
value: (string)
}]
}

DELETE Queued Action

Use Case: Removes a single queued action. Only scripts created by the current user and are "SCHEDULED" can be removed.

URL: http://host/prime-home/portal/devices/{id}/actions/{scriptRunId}

Method: DELETE

Trigger Device Event

Use Case: Fires an event for the related device. Any actions resulting from this event will be queued for this call return. This restful service is for internal functions only and is not available for third-party integration.

URL: http://host/prime-home/portal/devices/{id}/events

Method: PUT

Request

<request-data>
{

event: (string: INITIAL_CONTACT|REBOOT|
SUBSCRIBER_ASSOCIATED|SERVICE_ENABLE|
SERVICE_DISABLE|VALUE_CHANGE|
TRANSFER_COMPLETED|
DIAGNOSTIC_COMPLETED|INFORM)
}
</request-data>

Response

<response-data>
{
event: (string: INITIAL_CONTACT|REBOOT|
SUBSCRIBER_ASSOCIATED|SERVICE_ENABLE|
SERVICE_DISABLE|VALUE_CHANGE|
TRANSFER_COMPLETED|
DIAGNOSTIC_COMPLETED|INFORM)
ids:[(number), ...]
}
</response-data>

GET Device Labels

Use Case: Gets the labels for a device.

URL: http://host/prime-home/portal/devices/{id}/labels

Method: GET

Response

<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}
</response-data>

Modify Device Labels

Use Case: Sets all labels for a device.

URL: http://host/prime-home/portal/devices/{id}/labels

Method: POST

Request

<request-data>["labelName",”labelName1”,….”labelName n”]</request-data>

Response

<response-data>
[{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}]
</response-data>

Create device labels

Use Case: Adds the given labels for a device.

URL: http://host/prime-home/portal/devices/{id}/labels

Method: PUT

Request

<request-data>["labelName",”labelName1”,….”labelName n”]</request-data>

Response

<response-data>
[{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}]
</response-data>

GET Device Statistics Summary

Use Case: Returns a summary of miscellaneous statistics about a device.

URL: http://host/prime-home/portal/devices/{id}/activity

Method: GET

Response

```
<response-data>
{
informCount: number,
lastInformDelta: number,
scriptQueue: {
pending: number
}
}
</response-data>
```

GET Solicit Status

Use Case: Gets the solicit status.

URL: http://host/prime-home/portal/devices/{id}/solicit

Method: GET

Response

```
[{
id: (number),
status: (string: UNKNOWN|PENDING|SUCCESS|
DECLINED|BEST_ATTEMPT|UNREACHABLE),
lastStatus: (timestamp),
lastSuccess: (timestamp)
lastBestAttempt: (timestamp)
}]
```

Create Device Update Queue

Use Case: Queues devices to update their status and causes and scheduled scripts to be run against the device. Passed data is an array of objects, each must contain id and lastStatus. The value for lastStatus will be populated in the return value. When querying for related status, lastStatus will contain a value greater or equal to this value to indicate that the requested solicit has been completed.

URL: http://host/prime-home/portal/devices/{id}/solicit

Method: PUT

Request

[{
id: (number),
lastStatus: (number)
}]

Response

<response-data>
[{
id: (number),
lastStatus: (number)
}]
</response-data>

GET Device Session Trace List

Use Case: Returns a summary of recent trace logs for the device. Results are ordered with the most recent session first.

URL: http://host/prime-home/portal/devices/{id}/trace

Method: GET

Response

<response-data>
[{
traceId: number,
sessionId: string,
hardwareVersion: string,
softwareVersion: string,
informEvents: [string, string, ...],
begin: number (utc millisecond timestamp),

durration: number (milliseconds),
informCount: number
}, ...]
</response-data>

Create Device Trace Log

Use Case: Begin a new trace log for the device.

URL: http://host/prime-home/portal/devices/{id}/trace

Method: PUT

Request

<request-data>
{
sessionId: string,
begin: number (utc millisecond timestamp),
hardwareVersion: string,
softwareVersion: string,
infromEvents: [string,string,...]
}
</request-data>

Response

<response-data>
{
traceId: number,
sessionId: string,
hardwareVersion: string,
softwareVersion: string,
infromEvents: [string, string,...],
informCount: number
}
</response-data>

GET Device Trace Level

Use Case: Gets the logging level for the device.

URL: http://host/prime-home/portal/devices/{id}/trace/level

Method: GET

Response

```
<response-data>
{
loggingLevel: string ("NONE","INFO","DEBUG","TRACE")
}
</response-data>
```

Configure Device Logging Level

Use Case: Configure logging level for the device.

URL: http://host/prime-home/portal/devices/{id}/trace/level

Method: POST

Request

```
<request-data>
{
loggingLevel: string ("NONE","INFO","DEBUG","TRACE")
}
</request-data>
```

Response

```
<response-data>
{
loggingLevel: string ("NONE","INFO","DEBUG","TRACE")
}
</response-data>
```

GET Device Session Trace Summary

Use Case: Returns a summary of the session trace log for a device.

URL: http://host/prime-home/portal/devices/{id}/trace/level{trace id}

Method: GET

Response

<response-data>
{
traceId: number,
sessionId: string,
hardwareVersion: string,
softwareVersion: string,
begin: number (utc millisecond timestamp),
durration: number (milliseconds),
informCount: number
}
</response-data>

GET Device Session Trace Detail

Use Case: Retrieve the details of a single CPE session. Results are ordered as they were logged.

URL: http://host/prime-home/portal/devices/{id}/trace/level{trace id}/log

Method: GET

Response

<response-data>
[{
logId: (number), type: "event", event: (string)
timestamp: (number) (milliseconds after trace start),
}, {
logId: (number), type: "script begin",
timestamp: (number: milliseconds after trace start),
scriptName: (string), scriptType: (string),
params: [{
name: (string),
value: (string),
type: (string)},...}],

source: (string)
}, {
logId: (number), type: "script custom",
timestamp: (number: milliseconds after trace start),
level: (string: "INFO"|"WARN"|"ERROR"|"FATAL"),
message: (string)
}, {
logId: (number), type: "script call",
timestamp: (number: milliseconds after trace start),
stacktrace: (string),
operation: (string)
}, {
logId: (number), type: "script response",
timestamp: (number: milliseconds after trace start),
operation: (string)
}, {
logId: (number), type: "script end"
timestamp: (number: milliseconds after trace start),
}, {
logId: (number), type: "script abort"
timestamp: (number: milliseconds after trace start),
}, {
logId: (number), type: "soap receive",
timestamp: (number: milliseconds after trace start),
payload: (string)
}, {
logId: (number), type: "soap transmit",
timestamp: (number: milliseconds after trace start),
payload: (string)
},...]
</response-data>

Access Devices on Activation Server (GET)

Use Case: Gets devices known by the activation server.

URL: http://host/prime-home/portal/activation/devices

Method: GET

Query

<query>first=(number)//thefirstresulttoreturn
count=(number)//thenumberofresultstoreturn
q=(string)devicetosearchfor.SearchesbyOUI/SerialNumber/provisioningcode
</query>

Response

<response-data>[
{
oui: (string),
serialNumber: (string),
provisioningCode: (string),
manufacturer: (string),
model: (string)
}
]</response-data>

SUBSCRIBER APIS

GET ACS Subscriber List

Use Case: Retrieves a list of subscribers in the system.

URL: http://host/prime-home/portal/subscribers

Method: GET

Response

<response-data>
[
{
id: (number),
fullname: (string),
email: (string)
}
]
</response-data>

Create New ACS Subscriber

Use Case: Creates a new subscriber.

URL: http://host/prime-home/api/v2/subscribers

Method: POST

Request

{"dto":{"Subscriber":{"FullName":"John Doe","EmailAddress":"johndoe@email.com"}},"subscriptions":[],"labels":[],"credentials":{},"-code":"johndoe123"}

Response

{"dto":{"Subscriber":{"FullName":"John Doe","EmailAddress":"johndoe@email.com"}},"revision":"955C2225275E3822A7098F7D578ECB94",
"subscriptions":[],"labels":[],"subscriberId":2271,"credentials":{},"code":"johndoe123"}

Request Password Change

Use Case: Requests an automated change of password with email notification.

URL: http://host/prime-home/portal/subscribers/resetPassword

Method: PUT

Request

<request-data>
{
email: (string),
locale: (string)
}

</request-data>

Response

200 OK

GET Statistics for Subscribers

Use Case: Gets summary information for all active subscribers.

URL: http://host/prime-home/portal/subscribers/summary

Method: GET

Response

<response-data>
{ totalActiveSubscribers: (number) }
</response-data>

GET Statistics for Specific Subscriber

Use Case: Retrieve a subscriber indicated by {personId}. A subscriber is a person which may have any number of devices.

URL: http://host/prime-home/portal/subscribers/{personId}

Method: GET

Response

<response-data>{id: (number), fullname: (string), email: (string) }</response-data>

Modify Subscriber

Use Case: Update a subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}

Method: POST

Request

<request-data>{ code: (string) }</request-data>

Response

<response-data>{ id: (number), code: (string) }</response-data>

Delete Subscriber

Use Case: Delete an existing subscriber. This will fail if there is a management group with active services.

URL: http://host/prime-home/portal/subscribers/{personId}

Method: DELETE

GET Subscriber Label Set

Use Case: Retrieves the subscriber's set of labels.

URL: http://host/prime-home/portal/subscribers/{personId}/labels

Method: GET

Response

<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}

Define Labels for Subscriber

Use Case: Sets the labels for a subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}/labels

Method: POST

Request

<request-data>["labelName",...]</request-data>

Response

<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}
</response-data>

Create Subscriber Labels

Use Case: Adds the labels for a subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}/labels

Method: PUT

Request

<request-data>["labelName",...]</request-data>

Response

<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}
</response-data>

GET Subscriber Management Groups

Use Case: Retrieves the list of management groups associated to the subscriber. A management group consists of a managed device and a number of service package subscriptions on that device.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups

Method: GET

Query

<query>
There are two types of queries available:
CA=(version): returns only groups which have a clearaccess firmware, and a firmware version greater than or equal to (version). No paging is available for this search type
OR:
q=xxyy: the serial number/pcode/oui are done using a case insensitive "like" operator and are "ORed" paging is available for this search type using first and count parameters in the querystring.
If no parameters are passed. All devices for the subscriber are returned. This may be paginated.
</query>

Response

<response-data>
[{
id: (number),
managedDevice: {
id: (number)
},
subscriptions: [{
id: (number),
packageName: (string)
enabled: (bool)
}]
}]
</response-data>

Create Subscriber Management Group

Use Case: Creates a management group for the subscriber. If managedDevice is present, deviceSignature is ignored. If present, deviceSignature must contain exactly one of: 1. oui and serialNumber, 2. or provisioningCode, 3. or cpProvisioningId.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups

Method: PUT

Request

<request-data>{
managedDevice: {
id: (number)
},
deviceSignature: {
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
}
}
</request-data>

Response

<response-data>
{
id: (number),
managedDevice: {
id: (number)
},
deviceSignature: {
id: (number),
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
},
subscriptions: [
{
id: (number),
packageName: (string)enabled: (bool)
},
...
]
}</response-data>

GET Subscriber Management Group Details

Use Case: Returns a management group for the subscriber. A management group consists of a managed device or device signature, and a number of service package subscriptions. If managedDevice is present, deviceSignature is ignored. If present, deviceSignature must contain exactly one of: 1. oui and serialNumber, 2. provisioningCode, or 3. cpProvisioningId.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}

Method: GET

Response

```
<response-data>{
id: (number),
managedDevice: {
id: (number)
},
deviceSignature: {
id: (number),
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
},
subscriptions: [
{
id: (number),
packageName: (string)enabled: (bool)
},
...
]
}</response-data>
```

Modify Subscriber Management Group

Use Case: Updates a management group for the subscriber. If managedDevice is present, deviceSignature is ignored. If present, deviceSignature must contain exactly one of: 1. oui and serialNumber, 2. provisioningCode, or 3.cpProvisioningId.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}

Method: POST

Request

```
<request-data>{
```

managedDevice: {
id: (number)
},
deviceSignature: {
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
}
}
</request-data>

Response

<response-data>
{
id: (number),
managedDevice: {
id: (number)
},
deviceSignature: {
id: (number),
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
},
subscriptions: [
{
id: (number),
packageName: (string)enabled: (bool)
},
...
]
}</response-data>

Delete Subscriber Management Group

Use Case: Deletes this management group. This method will fail if there are any subscriptions which have not already been removed.

URL: http://host prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}

Method: DELETE

GET Subscriptions for Management Group

Use Case: Lists the subscriptions in the current management group.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/subscriptions

Method: GET

Response

```
<response-data>[
{
id: number,
status: (string: 'ENABLED'|'ENABLING'|'DISABLED'|'DISABLING')package: {
id: number,
name: (string)
}
}
]</response-data>
```

Add Subscription to Management Group

Use Case: Adds a subscription to the management group.

If the status becomes "ENABLED", the ACS will attempt to enable the service on the device specified in the management group. If the status becomes "DISABLED", the ACS will disable the service.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/subscriptions

Method: PUT

Request

<request-data>{
package: {
id: (number),
name: (string)
},
status: (string: 'ENABLED'|'ENABLING'|'DISABLED'|'DISABLING')
}</request-data>

Response

<response-data>{
id: number,
package: {
id: (number),
name: (string)
},
status: (string: 'ENABLED'|'ENABLING'|'DISABLED'|'DISABLING')
}</response-data>

GET Service Subscription Details

Use Case: Retrieve information about a subscription. to a service with the specified {packageName}.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/subscriptions/{packageName}

Method: GET

Response

<response-data>{
id: number,
package: {
id: (number),
name: (string)
},
status: (string: 'ENABLED'|'ENABLING'|'DISABLED'|'DISABLING')
}</response-data>

Modify Subscription Package Status

Use Case: Updates the status of the given subscription package.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/subscriptions/{packageName}

Method: POST

Request

```
<request-data>
{ status: (string: 'ENABLED'|'ENABLING'|
'DISABLED'|'DISABLING') }
</request-data>
```

Response

```
<response-data>
{
id: number,
package: {
id: (number),
name: (string)
},
status: (string: 'ENABLED'|'ENABLING'|
'DISABLED'|'DISABLING')
}
</response-data>
```

Remove Subscription Package

Use Case: Deletes the specified subscription. Fails if the status is anything except DISABLED.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/subscriptions/{packageName}

Method: DELETE

GET Management Group Device Signature

Use Case: Returns the device signature of the management group. A device signature represents a device which may connect to the ACS at some future time. Upon doing so, the device matching a device signature is automatically associated to the subscriber and any service package which are marked as "enabled" will be applied. A device signature must contain exactly one of: 1. oui and serialNumber, 2. provisioningCode, or 3. cpProvisioningId.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/deviceSignature

Method: GET

Response

<response-data>{
id: (number),
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
}</response-data>

Modify Management Group

Use Case: Updates the properties of this management group.

URL: http://host/prime-home/portal/subscribers/{personId}/labels/managementGroups/{managementGroupId}/deviceSignature

Method: POST

Request

<request-data>{
oui: (string),
serialNumber: (string),
provisioningCode: (string),
cpProvisioningId: (number)
}</request-data>

Response

<response-data>{
id: (number),
oui: (string),
serialNumber: (string),

provisioningCode: (string),
cpProvisioningId: (number)
}</response-data>

GET Subscriber Attribute Tree

Use Case: Retrieves the subscriber's attribute tree. Refer to the integration guide for schema details.

URL: http://host/prime-home/portal/subscribers/{personId}/attributes

Method: GET

Response

<response-data>{ A: { B: [c, d]}}</response-data>

Modify Subscriber Attributes

Use Case: Updates the attributes for the given subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}/attributes

Method: PUT

Request

<request-data>{"a.b.c.": "d"}</request-data>

Response

<response-data>{ A: { B: [c, d]}}</response-data>

Create Subscriber Attributes

Use Case: Creates the attributes for the given subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}/attributes

Method: Post

Request

<request-data>{"a.b.c.*.d": "d"}</request-data>

Response

<response-data>{ A: { B: {1, {d: "d"}}</response-data>

DELETE Subscriber Attributes

Use Case: Removes a portion of the attribute tree for the given subscriber.

URL: http://host/prime-home/portal/subscribers/{personId}/attributes

Method: DELETE

Query

<query>
name=(string)
required. The portion of the attribute tree to remove.
</query>

Response

{"Subscriber":{"FullName":"<Full name>","EmailAddress":"<Email Address"}}

Change Password (by Subscriber)

Use Case: Self-administered subscriber password within the control panel.

URL: http://host/prime-home/portal/subscribers/{personId}/changePassword

Method: POST

Request

<request-data>
{
currentPassword: string
newPassword: string
}
</request-data>

Response

200 OK

Create Subscriber Login Sessions

Use Case: Creates Subscriber Login Session.

URL: http://host/prime-home/portal/control-panel/session

Method: PUT

Request

{
login: (string),
password: (string)
}

Response

{"expired":true,"sessionId":"0CBA1518CA010F71A292332B6D279859CF5B19B5","subscriberId":7,"login"(String)}

Modify Subscriber Login Session

Use Case: Updates a subscriber's login session.

URL: http://host/prime-home/portal/control-panel/session/{tokenId}

Method: POST

Request

{
“cpSessionId”:”<session id>”
}

Response

200 OK

Remove Subscriber Login Session

Use Case: Removes the session.

URL: http://host/prime-home/portal/control-panel/session/{tokenId}

Method: DELETE

GET Session Expiration Status

Use Case: Gets the session expiration status.

URL: http://host/prime-home/portal/control-panel/session/

Method: GET

Response

{"expired":true,"sessionId":"0CBA1518CA010F71A292332B6D279859CF5B19B5","subscriberId":7,"login"(String)}

GET ACS User List

Use Case: Lists the set of ACS users.

URL: http://host/prime-home/portal/users

Method: GET

Response

[{login: string, fullname: string, email: string }]

Modify ACS User List

Use Case: Add a user to the ACS.

URL: http://host/prime-home/portal/users

Method: PUT

Request

<request-data>
{
login: string,
fullname: string,
email: string,
password: string,
domains: ["code"]
}
</request-data>

Response

```
{
Id: string,
login: string,
fullname: string,
email: string,
password: string,
domains: ["code"]
}
```

GET User Details

Use Case: Gets user details for the given login name.

URL: http://host/prime-home/portal/users/{login}

Method: GET

Response

```
{
Id: string,
login: string,
fullname: string,
email: string
}
```

Modify User Details

Use Case: Change the user's details.

URL: http://host/prime-home/portal/users/{login}

Method: Post

Request

```
<request-data>
{
login: string,
fullname: string,
email: string,
password: string,
enabled: boolean,
```

domains: ["code"]
}
</request-data>

Response

{
Id: string,
login: string,
fullname: string,
email: string
}

DELETE User

Use Case: Removes a user.

URL: http://host/prime-home/portal/users/{login}

Method: DELETE

GET User Notifications for past 10 Minutes

Use Case: Gets the current set of notifications generated by the ACS to be viewed by the user. Includes all notifications which have occurred in the last 10 minutes.

URL: http://host/prime-home/portal/users/{login}/notifications

Method: GET

Response

<response-data>
[{
id: int,
text: string,
level: string(info,warn,error),
source: string(system,user)
}]
</response-data>

Create User Notification

Use Case: Adds a user notification.

URL: http://host/prime-home/portal/users/{login}/notifications

Method: PUT

Request

```
<request-data>
{
id: int,
code: string,
labels: [],
}
</request-data>
```

Response

```
{{
id: int,
code: string,
labels: [],
}
```

GET User Preferences (Not used)

Use Case: Retrieves the ACS user's preferences. Currently this is not used.

URL: http://host/prime-home/portal/users/{login}/preferences

Method: GET

Response

200 OK

Modify User Preferences

Use Case: Updates the user's preferences.

URL: http://host/prime-home/portal/users/{login}/preferences

Method: POST

Request

<request-data>{ language: (en|es|fr) }</request-data>

Response

200 OK

GET User Roles

Use Case: Lists the user's set of roles.

URL: http://host/prime-home/portal/users/{login}/roles

Method: GET

Response

[“admin”] or [“csr”]

Modify User Roles

Use Case: Updates user's roles.

URL: http://host/prime-home/portal/users/{login}/roles

Method: POST

Request

<request-data>['role1','role2',...]</request-data>

Response

http://host/prime-home/portal/users/{login}/roles

GET User Labels

Use Case: Gets the labels for a user.

URL: http://host/prime-home/portal/users/{login}/labels

Method: GET

Response

<response-data>
{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}
</response-data>

Modify User Labels

Use Case: Updates the labels for a user.

URL: http://host/prime-home/portal/users/{login}/labels

Method: POST

Request

<request-data>["labelName",...]</request-data>

Response

<response-data>
[{
id: number,
text: string,
fgColor: "#rgb",
bgColor: "#rgb"
}]
</response-data>

GET Meta Data for Subscribers

Use Case: Gets a tree representing the subscriber attribute schema (metadata), filtered by the given expressions. It does not need to be qualified in the same way that the device attribute schema does because a subscriber can only have one kind of data tree.

URL: http://host/prime-home/portal/schema/device/subscriber

Method: GET

Query

```
<query>
by simple string:
Subscriber
Subscriber.Address
by JSON object:
{
Subscriber: {
Address: "*",
FullName: "*"
}
}
</query>
```

Response

```
{
schema: {
__schema__: {

}ModelItem: {
__schema__: {
modifiable: (boolean,
arraysonly),
writeable: (boolean,
attributesonly),
type: (OBJECT|STRING|ARRAY|BOOLEAN|DATE_TIME|UNSIGNED_INT)source: (Model|Schema)displayName: (string)
}
}
}
}
```

Troubleshooting - 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 in the API Standards chapter 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: Verify that service names, labels and any other names which refer to something else are correct.

Problem: A 400 status caused by a data validation error

Example: WIFI Key is Too Short for WPA

```
{
    "settings": {
    "Settings.AccessPoint.1.SecurityMode":"WPA-WPA2-Personal", "Settings.AccessPoint.1.Secret":"abcd",
    }
}
```

Example: 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 detailed information regarding the exact validation error.

JSON Formatting

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

Example: Un-closed braces

```
{
    a: 1
```

Example: Un-closed string

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

Example: Trailing comma

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

Solution: Fix your JSON syntax. For more information, go to http://www.jsonlint.com/.

Schema

Three schema files (in JSON format) are included with this User Manual. Two are included with the PDF:

- Schema-5.1.1.1
- Schema-5.2.0.5

The third file, WiFI + PPP settings, is shown below.

WiFi + PPP settings schema

You can access the following schema at URL: GET /prime-home/api/v1/applications/schemas/.


```
"PPP": [
{
"definition": "WANPPPConnections",
"description": "array"
},
{
"definition": "WANPPPConnections.{}.Password",
"description": "string"
},
{
"definition": "WANPPPConnections.{}.Username",
"description": "string"
},
{
"definition": "WANPPPConnections.{}.TR098Path",
"description": "optional string"
}
]
"WiFi" : [
{
{
"definition": "WIFI",
"description": "object"
},
{
"definition": "WIFI.AccessPoint",
"description": "array"
},
{
```

"definition": "WIFI.AccessPoint.{}.Enable",
"description": "boolean"
},
{
"definition": "WIFI.AccessPoint.{}.Secret",
"description": "optional string [minLength=5 maxLength=63]"
},
{
"definition": "WIFI.AccessPoint.{}.SSID",
"description": "string [minLength=1 maxLength=32]"
},
{
"definition": "WIFI.AccessPoint.{}.SecurityMode",
"description": "enum [WEP-64, WPA-Personal, WEP-128, WPA2-Personal, None, WPA-WPA2-Personal]"
},
{
"definition": "WIFI.AccessPoint.{}.EnableAdvertisement",
"description": "boolean"
},
{
"definition": "WIFI.AccessPoint.{}.EnableLocalNetworkAccess",
"description": "boolean"
},
{
"definition": "WIFI.AccessPoint.{}.Radios",
"description": "array"
},
{
"definition": "WIFI.AccessPoint.{}.Radios.{}.Reference",
"description": "numeric [minValue=1]"
},
{
"definition": "WIFI.Radio",
"description": "array"
},
{
"definition": "WIFI.Radio.{}.OperatingStandards",
"description": "string [minLength=1]"
},
{
"definition": "WIFI.Radio.{}.ChannelBandwidth",
"description": "enum [20MHz, 40MHz, 80MHz, Auto]"
},

```
{
"definition": "WIFI.Radio.{}.Channel",
"description": "optional numeric [minValue=1 maxValue=200]"
},
{
"definition": "WIFI.Radio.{}.FrequencyBand",
"description": "enum [2.4GHz, 5GHz]"
}
}
]
```

EXAMPLES OVERVIEW

The following sections contain code samples for the following:

- Interacting with Search
- Interacting with the Device Lifecycle
- Creating Control Panel sessions for Single Sign-on
- Some deprecated and obsolete samples

Details for each example are shown in the following format.

Attribute	Description
Title	Descriptive name for the API.
Use Case	Description of the use case for the API.
Method	Step-by-step process.

HTTP Interaction	**Request**
	METHOD /resource
	{
	"key":"value"
	}
	Response
	{
	"key":"changed value"
	}

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

SEARCH EXAMPLES

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"
}
```

Services

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

EXAMPLES- INTERACTING WITH 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": []
}
```

HTTP Interaction - Creating a Device

Request

POST /prime-home/portal/query/execute
device with oui: 001A2B999999

Response

[ ]

Finding a Device

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"
```

```
}
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


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

```
                "status": "DISABLED"
            },
            (...)
        },
        "solicitableStatus": "OK"
}
```


Request

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

Response


```
{
    "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": {"WIFI": {
                "AccessPoint": {
                    "2": {
                        "Enable": "true",
```

```
                    "EnableAdvertisement": "true",
                    "EnableLocalNetworkAccess": "true",
                    "Radios": {"1": {"Reference": "2"}},
                    "SSID": "NETGEAR-3800-5G-demo",
                    "Secret": "kdleufke82393jdke",
                    "SecurityMode": "WPA2-Personal"
                },
                "3": {
                    "Enable": "true",
                    "EnableAdvertisement": "true",
                    "EnableLocalNetworkAccess": "true",
                    "Radios": {"1": {"Reference": "1"}},
                    "SSID": "NETGEAR-3800-2.4G-demo",
                    "Secret": "ekfueopzqlj384jfi",
                    "SecurityMode": "WPA2-Personal"
                },
            },
            "Radio": {
                "1": {
                    "ChannelBandwidth": "20MHz",
                    "FrequencyBand": "2.4GHz",
                    "OperatingStandards": "b,g,n"
                },
                "2": {
                    "Channel": "44",
                    "ChannelBandwidth": "40MHz",
                    "FrequencyBand": "5GHz",
                    "OperatingStandards": "n"
                }
            }
        }}
    }}
},
"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
```

Response


```
{
    "applications": {
        "WIFI": {"dto": {
            "Settings": {"WIFI": {
                "AccessPoint": {
                    "2": {
                        "Enable": "true",
                        "EnableAdvertisement": "true",
                        "EnableLocalNetworkAccess": "true",
                        "Radios": {"1": {"Reference": "2"}},
                        "SSID": "life-cycle-use-case-ssid",
                        "Secret": "kdleufke82393jdke",
                        "SecurityMode": "WPA2-Personal"
                    },
                    "3": {
                        "Enable": "true",
                        "EnableAdvertisement": "true",
                        "EnableLocalNetworkAccess": "true",
                        "Radios": {"1": {"Reference": "1"}},
                        "SSID": "NETGEAR-3800-2.4G-demo",
                        "Secret": "ekfueopzqlj384jfi",
                        "SecurityMode": "WPA2-Personal"
                    },
                },
                "Radio": {
                    "1": {
                        "ChannelBandwidth": "20MHz",
                        "FrequencyBand": "2.4GHz",
                        "OperatingStandards": "b,g,n"
```

```
                        },
                        "2": {
                            "Channel": "44",
                            "ChannelBandwidth": "40MHz",
                            "FrequencyBand": "5GHz",
                            "OperatingStandards": "n"
                        }
                    }
                }}
            }},
            (...)
        },
        "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": {"WIFI": {
                "AccessPoint": {
                    "2": {
                        "Enable": "true",
                        "EnableAdvertisement": "true",
                        "EnableLocalNetworkAccess": "true",
                        "Radios": {"1": {"Reference": "2"}},
                        "SSID": "life-cycle-use-case-ssid",
```

```
                    "Secret": "kdleufke82393jdke",
                    "SecurityMode": "WPA2-Personal"
                },
                "3": {
                    "Enable": "true",
                    "EnableAdvertisement": "true",
                    "EnableLocalNetworkAccess": "true",
                    "Radios": {"1": {"Reference": "1"}},
                    "SSID": "NETGEAR-3800-2.4G-demo",
                    "Secret": "ekfueopzqlj384jfi",
                    "SecurityMode": "WPA2-Personal"
                },
            },
            "Radio": {
                "1": {
                    "ChannelBandwidth": "20MHz",
                    "FrequencyBand": "2.4GHz",
                    "OperatingStandards": "b,g,n"
                },
                "2": {
                    "Channel": "44",
                    "ChannelBandwidth": "40MHz",
                    "FrequencyBand": "5GHz",
                    "OperatingStandards": "n"
                }
            }
        }}
    }},
(...)
},
"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

N/A

EXAMPLES - 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: oui, serial number, and session ID.
5. Redirect the browser.

HTTP Interaction (Web Services)

Obtaining the Single Sign-On Session Information

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-e=001A2B:001A2B999999.`

   This 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-e=001A2B:001A2B999999`

   This is the actual control panel UI.

EXAMPLES - DEPRECATED

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

Working with Subscribers in Java

This example illustrates using the Subscriber API to work through 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.AnnotationConfigApplicationContextin Jaa
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(newJSONObject(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.getCode()), 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.getPostalCode());
                  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",
        newJSONObject());
        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;
        }
}
```

Data Classes - 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;
    }
```

Data Classes - 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;
	}

}
```

Support - 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()));
                }
            });
        }
}
```

Support - 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);
        }
    }
}
```

EXAMPLES - OBSOLETE

NOTE: The following examples show APIs that are no longer supported as of version 5.1.2.

Changing SSID at index 1 with CURL

Introduction

As WiFi now supports multiband WiFi connections, the following example is no longer supported as it assumes there is only one WiFi Access Point. Instead use get, modify, and put it on the existing Access Point after determining the correct index.

Main Example

```
#!/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=$(cat << EOS
s/\"Settings.WLANConfigurationsAccessPoint.1.SSID\":\"[^\"]*\"/\"Settings.WLAN
ConfigurationsAccessPoint.1.SSID\":\"$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
```

APPENDIX A: CAQL 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"
```

Revision History

Rev	Date	Description
1.0	June 2016	Initial document release
1.1	October 2016	Release for v5.2
1.2	August 2017	Added sections for Get Device Template and Create Device. Updated old URL endpoints for Delete Device and Create New ACS Subscriber
1.3	January 2019	Complete overhaul of "Schedule Device Action For Next Contact" section. (Begins on pg 47)