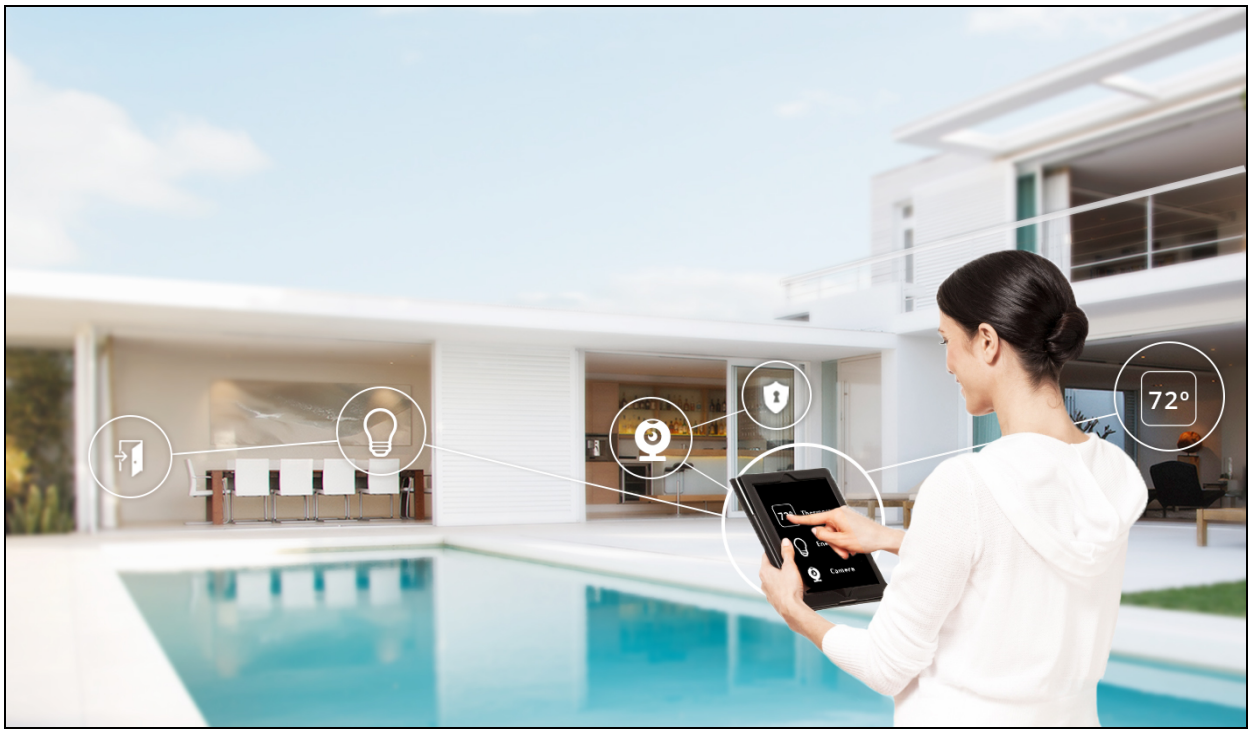
icontrol NETWORKS

REST API Getting Started Guide

7.3 Quadra



Document Information

Release version: 7.3 Quadra
Document name: REST API Getting Started Guide
Build date: 9/7/2016

Contents

Revision History

Release	Revisions
7.3 Quadra v3	Added the `version="<firmware_name>"` attribute to the CPE device example in Developer Tools
Quadra 7.3 v2	Removed "instant" as an option for arming the touchscreen anywhere it was mentioned.
Quadra 7.3 v1	Added the following sections to section to "HTTP Headers" on page 28: ❑ "X-loginEncoded: Base64-encoded strings" ❑ "X-expires and X-token: Token Expiration for REST Modules" Added warning about preventing CSRF attacks in "Session Authentication" on page 6 Updated description of `eventsByDay` API in "Get All Security and Home Automation Events by Day" on page 21 Added `instance/virtualDevice` to "Instance Types" on page 32 Added "Point media types: Cloud Objects" on page 35 Added the following note to "Get Security Events by Day" on page 19 and "Events Array" on page 22: **Note:** For all alarm events, the output of the securityEventsByDay call also includes the `isTestAlarm` and `isAcked` properties.

Release	Revisions
Padre 7.2	Deleted the following sections as the content in these sections was incorrect and/or irrelevant to Converge/Touchstone: ❑ "System Status" ❑ "Gateway Status" ❑ "Connection Status" ❑ "GPRS Connection Status In-Depth" Updated "Upgrading Firmware" on page 23 as follows: ❑ Removed content that was irrelevant to Converge/Touchstone ❑ Added references to information about the `firmwareUpdate` API Added the following caution to the `camera/isAlive` function in "Function media types" on page 35: **IMPORTANT:** On Touchstone, this function may return HTTP status 504 (gateway timeout) if no response is received within 60 seconds. This indicates that the camera may not be online yet (equivalent to isAlive=false). If you receive this timeout, you can retry the call until you receive a response.

Release	Revisions
Oahu 7.1	Renamed this document from *RESTful API User Guide* to REST API Getting Started Guide. Moved the following API sections (previously located in *RESTful API User Guide*) to the Icontrol Customer Support Knowledge Base here: https://share-icontrol.atlassian.net/wiki/display/CSKB/REST+API+User+Guides ❑ Devices Examples ❑ Delta Events ❑ Miscellaneous APIs ❑ Contact Management ❑ Rules API for Converge and Touchstone ❑ Camera Motion ❑ Operational APIs ❑ Activation and Settings APIs ❑ Cloud Integration APIs ❑ Relays+ Devices For information about API updates in Oahu, see: https://share-icontrol.atlassian.net/wiki/display/CSKB/7.1+Oahu+Core+-+API+Updates Added the following appendices (previously located in *RESTful API Reference Guide*): ❑ "Security Panel Status" on page 45 ❑ "Security Panel Trouble Types" on page 46 ❑ "Sensor/Zone Trouble Types" on page 47 ❑ "Sensor/Zone Types" on page 48 ❑ "Sensor/Zone Function Types" on page 49 ❑ "Sensor Types" on page 50 ❑ "Peripheral Types" on page 53 *RESTful API Reference Guide* has been replaced by the API reference materials provided here: https://share-icontrol.atlassian.net/wiki/display/CSKB/REST+API+Reference

1 Introduction

The Icontrol REST API provides access to Icontrol-powered devices within the end user's home. This document is intended to give developers who want to use the API to write their own clients for the Icontrol system a solid foundation for understanding the usage and capabilities of the API.

Readers of this document are assumed to have a basic understanding of the Icontrol system. The document is divided into several major parts.

- Definitions gives quick, concise definitions for many of the key concepts in the APIs. To get the most out of the rest of the document, a reader should be familiar with all the terms in this section.
- REST API Basics describes the high level API framework.
- REST API Authentication and Authorization explains access controls in the API.
- REST API Resources provides detail around some of the key objects and the representation in the system.
- Use Cases - Security and Use Cases - System Status, Monitoring, and Maintenance provide a walk through of many of the standard API use cases.

The document also includes several appendices with supplemental material.

API reference materials and user documentation are available on the Icontrol Customer Support Knowledge Base here:
https://share-icontrol.atlassian.net/wiki/display/CSKB/7.3+Quadra+Core+-+API+Documentation

2 Definitions

Client: Application or website interacting with the API.

CMS: Central Monitoring Station. An external service for monitoring and processing alarms. The employees at the CMS monitor the system and call the police, fire department, or hospital when an alarm is detected.

Deltas: A Resource that can be used to track real-time changes to a site.

Device: A physical component of the Network such as a light switch, sensor or security panel.

Function: A subordinate Resource that allows the API user to invoke the capabilities of the parent Resource or set its configuration parameters. Functions can be used, for example, to tell a camera to take a picture, or to change a thermostat setpoint. Functions are also used to change the value of any write-able Point.

Group: A set of Preferences used to control system operation. Sites can be associated with one or more Groups, allowing different Sites to exhibit different behavior.

Instance: A representation of functionality provided by a Device. Instances are subordinate to Devices, and a Device may have one or more Instances. (As an example, a camera with a PIR motion detector would be represented as a single physical Device with two attached Instances: camera and motion detector).

Media Type: The type associated with a Resource in the Icontrol REST APIs. Not all Resources have media types, but many do. Instances and Points always have media types. For example, a Door/Window sensor Instance has Media Type 'instance/sensor', and the bypassed flag for a Door/Window sensor has a Media Type 'sensor/bypassed'.

Network: All of the Icontrol powered devices and instances installed in a consumer's premise.

Partner: A company providing service to Users. In this release, the partner id is fixed to 'icontrol'.

Point: Subordinate resource describing the state of its parent resource, typically associated with a Device Instance. Points can be read-write or just read-only. The setpoint temperature for a thermostat, and the current state (open/closed) of a door-window sensor, are examples of Points.

Preferences: Configurable key-value pairs controlling system operation. The number of cameras allowed in a particular account, or how often the CPE heartbeats with the server, are both examples of Preferences.

Resource: A Resource is a noun in the REST API. Partners, Users, Networks and Instances are all examples of Resources.

Service Profile: Special type of Group used to set the level of service for an account (whether the account has cameras or touchscreens enabled, which apps are displayed on the touchscreen, etc).

Site: The server view of an account, corresponding to an installation at a physical location (i.e. 'site'). The Site contains configuration information and history for the account, and a link to the Network resource associated with the account. The Site also contains a list of Users who have access to the account, as well as Group and Preference configuration information used to control Site and Network behavior. It also has links to stored media (video clips and images) associated with the Site.

Subordinate: The term subordinate designates that a Resource exists within or because of a higher-level Resource. As an example, User Resources are subordinate to Partner Resources. Generally a subordinate Resource will appear as such in the URI structure (i.e., 'a/b' is subordinate to 'a'), but this is not always the case.

User: End user or customer.

3 REST API Basics

3.1 Key API Concepts

The Icontrol REST API is very flexible and powerful, but using it requires understanding building blocks that go beyond the standard set of "functions/parameters/return codes" that define many other APIs. Fundamentally, the API is based around nouns (resources) rather than verbs (functions). Resources may have current state (often represented in the API as 'Points') and they may contain complex methods (represented in the API as 'Functions'). Resources may also have subordinate resources – resources that are children of a parent resource, and do not exist independently of the parent.

For example, if we have two resources, ‘a’ and ‘a/b’, then ‘a/b’ is subordinate to ‘a’. In a real-world case, ‘a’ may represent a thermostat, and ‘a/b’ might be a Point on ‘a’ representing the target temperature of the thermostat. A third resource, ‘a/c’, might be a function that allows the caller to change the target temperature to a new value.

Resources may also have types, referred to in the API as ‘mediaTypes’. In the example above, the mediaType of ‘a’ (the thermostat) might be ‘instance/thermostat’, and the mediaType of ‘a/b’ (the target temperature) might be ‘thermostat/temperature’. A list of the mediaTypes used by Icontrol can be found in "MediaTypes" on page 32. MediaTypes are key to the REST API architecture – if you want to know what type of thing a resource is, look at its mediaType.

In most cases, the XML representation of a resource will call out its subordinate resources. This allows the API user to browse through the API starting from key root resources, discovering many of the Resources, Points and Functions within the API. We encourage you to explore the API with a standard web browser.

To illustrate, in the example above, the XML representation of resource ‘a’ could be

```
<(resource class) mediaType=’instance/thermostat’>
<point mediaType=”thermostat/temperature” href = “a/b” >70</point>
<function mediaType=”instance/config” href = “a/c” method=”POST”>
<input mediaType=” thermostat/temperature” type=”text”
name=”temperature”/>
</function>
</(resource class)>
```

This XML representation shows a thermostat resource with target temperature currently set to 70 degrees, and defines a way to change the target temperature. ‘Resource class’ here is the top-level tag, which is not defined by the API framework. In the Icontrol system, resource classes are the high-level objects in the system – sites, users, devices, instances, groups, and preferences.

3.2 Requests

Requests are made to the API via the standard HTTP methods (GET, POST, PUT and DELETE) to resources as identified by a URI. It is generally safe to assume that GET, PUT and DELETE are idempotent; however, POST requests typically have side effects and therefore caution must be used when repeating POST transactions.

Where possible, parameters are passed to the system using standard query strings either at the end of the URI or in the body of a request. The API also supports custom HTTP headers (see "HTTP Headers" on page 28 for details).

Many clients are unable to use some of the less common HTTP methods, such as PUT and DELETE. To handle this situation, the API allows the client to override the actual HTTP method by explicitly specifying the method that it would use, if only it could. For instance, a client could create a DELETE request by using an HTTP GET and specifying as a parameter ‘X-HTTP-Override-Method=DELETE’. The X-HTTP-Override-Method can be parameterized as part of the URI, or can be added as an HTTP header.

3.3 Responses

3.3.1 Response Basics

Responses to queries will be HTTP responses, with appropriate response codes and headers. The structure of the content returned in the HTTP body will conform to the Content-Type header: text/xml for XML or JSON, image/jpeg for JPEGs, etc. Text–based data will usually be returned in XML format, although a request for a value associated with a Point will return a simple strings.

The API makes use of standard HTTP response codes to indicate the success or failure of a request. Response codes in the 200 range (200, 201, 202, etc) indicate that the request was successful. In general, response codes in the 400-499 range are problems with the request itself, that is, the resource or format of the query is invalid and should not be retried. Response codes in the 500-599 range are errors internal to the system and indicate that the request may be retried. Responses in the 300-399 range are not errors but redirects. It is the responsibility of the client to follow the 'Location' header provided by the redirect to get to the desired resource.

A full list of HTTP response codes used by the API can be found in "Response Codes " on page 44.

Some clients handle HTTP responses very coarsely, and treat all non-200 code responses as errors without providing any additional information. To handle these situations, the client should specify the 'X-format=xml' parameter in the request. If the X-format=xml parameter is specified, the response will always have code 200, however the body of the response will be XML specifying the actual response code, any response headers, and the what the body of the request would normally be.

For example, here's a typical response when X-format=xml has been specified

```
<response code="200">
  <header name="X-version">3.5</header>
  …
  <header name="Allow">POST, GET</header>
  <body count="1075">
    <site id="00603502002c">
      <description>00603502002c</description>
     <accountStatus>active</accountStatus>
      …
   </site>
  </body>
</response>
```

Note: The X-format header will apply even to resources that are not normally returned text, such as images.

3.3.2 Asynchronous Requests

Some requests may return a 202 response code, this indicates that the server has accepted the request, but the action is being carried out asynchronously. A 'Location' header will always accompany a 202 – this header provides the client with a URI where the status of the request may be monitored. The exact nature of this status will depend on the resource being requested.

As an example, a successful request to arm a security panel will always return HTTP code 202. The API user should monitor the URI specified in the 'Location' header to detect when the panel is actually armed.

3.3.3 500 responses and the X-error Header

In some cases, especially for broad errors such as 500 (server internal error), additional information is required for the client to understand the source of the error. When this is the case, the Icontrol system will provide an X-error header with a UCE error code. For more information, see the "Exception Codes (UCE)" section in *System Operations Guide*.

3.4 Use of HTTP Headers

The Icontrol REST API makes use of HTTP headers in both requests and responses. Request headers are used to provide authentication (see "REST API Authentication and Authorization " on page 5) or to specify special handling for clients that have difficulty dealing with the full set of standard HTTP responses. Response headers are used for cache control, to point to an alternate resource where a request in progress can be monitored, or to specify errors that are not covered by the standard HTTP response codes.

Non-standard headers specific to the Icontrol APIs are prefixed by 'X-', for instance 'X-HTTP-Override-Method', which is included in the request to specify that the request be interpreted as if it used a different HTTP method than it actually has. Request headers of this type may also be passed as standard parameters – for instance, the request

```
GET  /rest/icontrol/users/1234?email=bob@icontrol.com&X-HTTP-
Override-Method=POST
```

Is equivalent to

```
POST /rest/icontrol/users/1234
email=bob@icontrol.com
```

A full set of the HTTP headers used by the REST API can be found in "HTTP Headers" on page 28.

4 REST API Authentication and Authorization

4.1 Authentication

All users (including end-users and administrators) accessing the API must be authenticated in order to access non-public resources.

Initial API authentication is always done over HTTPS with a username, password, and a valid application key. If a request is successfully authenticated with the username/password and application key, the response will contain an HTTP session token that can be used to authenticate subsequent requests. See Session Authentication.

A request that fails authentication will return HTTP code 401 (unauthorized). If authentication succeeds but the requestor is not authorized to access the desired resource, the request will return 403 (forbidden).

4.1.1 Authenticating Requests

Any request can be authenticated. The username and password are passed in the HTTP headers 'X-login' and 'X-password', respectively. In addition, an additional HTTP header 'X-AppKey' is required to identify the type of client/application. The server will check the application key against a list of approved applications and reject the request if the key is not found in the list.

For convenience, the API defines a special resource, 'login' that can be used for authentication.

```
GET /rest/<partnerid>/login
```

By default, the returned data contains references to user and site resources. By using an optional 'expand' parameter, with possible value a combination of 'users', 'sites', and 'instances', the returned data expands the resource(s) specified. This reduces additional calls to the server for certain clients (e.g., mobile client).

4.1.2 Session Authentication

REST transactions are authenticated within the context of a session. After the user is authenticated by username and password, the REST server will create an HTTP session and return the HTTP session's session ID (by cookie) along with the response. The HTTP session will be expired in 30 minutes if there are no other requests coming in.

Subsequent requests can be authenticated without sending the user's username and password by passing this session ID in the header.

For example, the request:

```
GET /rest/<partnerId>/login
X-login: bob
X-password: secret
X-AppKey: defaultKey
```

returns a response containing the header:

```
Cookie: LzmEcNmhWQLSDSYn38w6vcDzLE4JwZfNHN73XSjgdELZCiahwbLIRPi
```

Subsequent requests can use the following headers for authentication:

```
Cookie: LzmEcNmhWQLSDSYn38w6vcDzLE4JwZfNHN73XSjgdELZCiahwbLIRPi
```

IMPORTANT: To prevent CSRF attacks, you must include the X-expires header in the login request to generate the X-token. For more information, see "X-expires and X-token: Token Expiration for REST Modules" on page 30.

As a practice, all clients should use authentication tokens rather than the user password in requests. There are security rules enforced by the server, and governed by preferences, which limit the number of password authentications that can be performed for a given user within a given time window. The default values for these are 5 password authentications within 30 minutes. As a result, a client that does not use the authentication tokens runs the risk of temporarily locking the user out of the system.

4.1.3 Terminating a Session

To terminate a session and invalidate a session ID, call the 'logout' resource. Following the example above:

```
POST /rest/<partnerId>/logout
Cookie: LzmEcNmhWQLSDSYn38w6vcDzLE4JwZfNHN73XSjgdELZCiahwbLIRPi
```

terminates the session by invalidating the passed-in session ID.

4.2 Authorization

Not all resources are available to every authenticated user. Each resource contains authorization gates that allow access to the resource based on the class of user performing the request. Clients that pass in user credentials that are valid but not authorized to access the desired resource will receive a "403 forbidden" response. No other information will be provided.

As a quick outline, an end user has access to all of the resources within his or her Site. Admin user 's access varies according to the admin level, but generally allows viewing all system information and often changing both user and system configuration.

For more information, see: https://share-icontrol.atlassian.net/wiki/display/CSKB/7.3+Quadra+Core+-+REST+API+Reference

5 REST API Resources

The base-level REST resource has URI /rest/<partnerId>, where <partnerId> is the unique Id for a partner/deployment in a given installation. In this release, the partner id is fixed to 'icontrol'. All other REST resources are subordinate to the /rest/<partnerid> resource.

5.1 Sites

A site represents a subscriber's account. It contains configuration and historical data, as well as all installed devices at the subscriber's premise.

The URI for a Site resource is /rest/<partnerid>/sites/<siteId>.

As an example, here is a typical site resource:

```
GET "/rest/icontrol/sites/101
<site id="101" href="/rest/icontrol/sites/101">
    <functions count="5">
```

```
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource by date range
and event types." name="alarmSessions"
action="/rest/icontrol/sites/101/alarmSessions" method="GET">
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the camera images and videos."
name="cameraImagesAndVideos"
action="/rest/icontrol/sites/101/cameraImagesAndVideos" method="GET">
            <input required="true" description="start record no"
type="text" name="start"/>
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
        <function gates="Admin,Self" description="Get the camera
images and videos." name="fetchImage"
action="/rest/icontrol/sites/101/fetchImage" method="GET">
            <input required="true" description="imageId" type="text"
name="imageId"/>
            <input required="true" description="imageType"
type="text" name="imageType"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource."
name="securityEvents"
action="/rest/icontrol/sites/101/securityEvents" method="GET">
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource by date range
and event types." name="securityEventsByDate"
action="/rest/icontrol/sites/101/securityEventsByDate" method="GET">
            <input description="comma separated event types"
type="text" name="eventTypes"/>
            <input required="true" description="start date"
type="text" name="startDate"/>
            <input required="true" description="end date" type="text"
name="endDate"/>
            <input required="true" description="comma separated list
of zone ids" type="text" name="zoneIds"/>
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
```

```
      </functions>
    <extId>exr474170</extId>
    <cmsInfo href="/rest/icontrol/sites/101/cmsInfo"/>
    <profile>
        <setting>
            <country>US</country>
            <locale>en</locale>
            <temperatureUnit>F</temperatureUnit>
            <postalCode>78758</postalCode>
        </setting>
    </profile>
    <timezone>US/Central</timezone>
    <createdAt>1324395528087</createdAt>
    <network href="/rest/icontrol/sites/101/network"/>
    <siteUsers count="0" href="/rest/icontrol/sites/101/users"/>
    <accountStatusV>activated</accountStatusV>
    <activationCode>338154</activationCode>
    <address href="/rest/icontrol/sites/101/address">
        <verified>false</verified>
    </address>
    <suspended>false</suspended>
    <active>true</active>
    <internal>false</internal>
    <readyForRMA>false</readyForRMA>
    <source>internal</source>
    <creator>ext3456</creator>
    <contacts count="0" href="/rest/icontrol/sites/101/contacts"/>
</site>
```

A quick description of some of these fields and subordinate resources:

Instance Type	Comments
functions	Functions that can be performed on a site resource.
extId	The account ID in an external system.
cmsInfo	Central monitoring station (CMS) related information, like the CMS account id, whether the account is monitored, emergency contacts etc.
profile	Defines account profile information, including the locale, temperature scale, and postal code.
timezone	Timezone where the site is located.
createdAt	Time, in milliseconds past the epoch (Jan 1, 1970 UTC) when the account was created

Instance Type	Comments
network	Link to the Network resource, parent resource of all hardware/software resources in a subscriber's premise.
siteUsers	Link to a list of Users who have access to the account.
accountStatusV	Account status.
activationCode	Code used during account activation.
suspended	Whether the account is suspended.
active	Whether the account is active.
internal	Whether the account is an internal account.
readyForRMA	Whether the account is ready for RMA.
source	Whether the account is created internally or externally.
creator	The external id of the user who created this account.
contacts	Link to users to be notified via email/sms.

5.2 Devices

Device resources represent physical objects installed in the user's home, such as cameras, thermostats, sensors, and the CPE. As an example, here is a typical Device resource:

```
<device id="101" hardwareId="133781220515569290" status="online"
firmwareVersion="6" hardwareVersion="7" type="lighting"
model="13261660" manufacturer="CentraLite Systems"
technology="zigbee"
href="/rest/icontrol/sites/101/network/devices/133781220515569290">
<name>Dimmable light 1</name>
<instances count="1">
      <instance
href="/rest/icontrol/sites/1234/network/instances/133781220515569290.
0"/>
</instances>
</device>
```

All Devices have an id, name, status, and type. It may have other attributes like model, manufacturer, firmware version etc.

Devices will also have one or more Instances associated with them. Instances are explained in more detail in the next section, but in brief, if a Device resource describes what an object physically is, an Instance resource describes what an object logically does. In the example above, the Device resource describes a dimmable light, and the Instance resource associated with the Device has Points describing the current state of the dimmer, as well as Functions to turn the switch on or off or to change the dimmer's level.

Though not common, a device may have multiple Instances, representing cases where the physical device encapsulates two or more distinct pieces of functionality. An example here is a camera with a built in motion-detector, which would be represented as a single Device with two Instances – one Instance representing the camera functionality, and one representing the motion-detector functionality.

A list of devices, along with references to individual devices, can be found as a resource subordinate to site:

```
GET /rest/<partnerId>/sites/<siteId>/network/devices
```

5.3 Instances

An instance resource represents a set of functionality in a physical Device. For instance, a physical camera will have both a Device Resource (describing the fact that the device is a physical camera with a particular manufacturer and model) and an Instance Resource representing what can be done with the camera (take a picture, view video). Typically an Instance and physical Device are one to one, however in some cases a single physical device may have several dependent Instances as described above.

Here's an example of an Instance resource:

```
<instance index="0" id="133781220491568969.0"
mediaType="instance/lightDimmer" tags="Light" status="ok">

    <name>Dimmable light 1</name>
    <points
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/">
        <point mediaType="light/label"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/label" ts="1336141628904">light 2 dim</point>
        <point mediaType="light/dimAllowed"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/dimAllowed" ts="1336141628904">true</point>
        <point mediaType="light/type"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/type" ts="1336141628904">dimmableLight</point>
        <point mediaType="light/isOn"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/isOn" ts="1336141628904">false</point>
        <point mediaType="light/level"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/level" ts="1336141628904">0</point>
        <point mediaType="light/troubles"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/troubles" ts="1336141628904"/>
    </points>
    <functions>
```

```
        <function method="POST" mediaType="instance/config"
action="/rest/icontrol/sites/101/network/instances/133781220491568969
.0/points/" name="Set Points">
            <input name="isOn" mediaType="light/isOn"
type="text">false</input>
            <input name="level" mediaType="light/level"
type="text">0</input>
        </function>
    </functions>
</instance>
```

Instance resources typically contain Points and Functions. Points represent the state of the Instance, for example, light level for a light dimmer or temperature for a thermostat. Functions describe what can be done with that Instance, for instance, changing the light level, setting a thermostat target temperature.

More information on Points and Functions can be found in the next two sections.

5.4 Points

The term 'point' is short for "data point" and represents a single piece of state on the resource, such as light level, current temperature or a boolean value concerning whether or not a motion sensor is detecting motion.

The URI of a Point can be found in the XML representation of a Point's parent resource. For instance, the XML for an Instance resource contains information about each of its subordinate Points. However, the value of a Point is generally contained in the XML of the parent. Here's an example point for a dimmable light instance:

```
<point mediaType="light/level"
href="/rest/icontrol/sites/101/network/instances/133781220491568969.0
/points/level" ts="1336141628904">0</point>
```

5.4.1 Point Values

Points can be read only or read/write. Any read/write Points may be manipulated through the API provided the client has the appropriate authorization. Read-only Points are typically physical values read by a device. For example a thermostat may have a read-only value for temperature, but read/write points for the heat set point and the fan mode.

Points that may be read can be accessed via a GET to their URI. The value of the point is returned directly and should not be assumed to be XML. For instance

```
GET
/rest/icontrol/sites/101/network/instances/133781220491568969.0/point
s/temperature
```

returns:

```
79
```

Point values on Instances can also be seen in the XML used to describe the Instance resource itself. So for example,

```
GET /rest/icontrol/sites/101/network/instances
```

or

```
GET /rest/icontrol/sites/101/network/instances /133781220491568969.0
```

would both return XML that included, among other things, the current value of the temperature. See preceding section on Instances for an example of an Instance resource.

5.4.2 Changing Point Values

All resources that have writeable Points have a Function that allows you to change the Point value. This Function has the mediaType 'instance/config'.

Here is an example of that function, and its usage, on a dimmable light:

```
    <function method="POST" mediaType="instance/config"
action="/rest/icontrol/sites/101/network/instances/133781220491568969
.0/points/" name="Set Points">
        <input name="isOn" mediaType="light/isOn"
type="text">false</input>
    </function>


POST
/rest/icontrol/sites/101/network/instances/133781220491568969.0/point
s/
isOn?value=true
```

A 200 code will be returned assuming the request was successfully submitted. Client should listen for real time updates via deltas (see "Deltas – Device Changes in Real-Time" on page 14) to see if the points are successfully updated in the targeted instance. If there is a problem changing the value of a point the appropriate error code will be returned.

5.5 Functions

Functions are capabilities that may be invoked on a resource. For example, asking a camera to invoke its "take picture" function. Functions may also simply return information, or can be used to set the value of Points.

5.5.1 Function Descriptors

The XML representation of a Function contains information that describes that Function: what it does (the mediaType), a URI to invoke it (the action) the HTTP method to invoke it with (the method), as well as whatever parameters it takes. The following is an example Function that invokes the take picture capability of a camera:

```
<function method="POST" mediaType="camera/directSnapshot"
action="/rest/icontrol/sites/101/network/instances/12101.5.0/function
s/takePicture" name="takePicture">
    <input name="imageSize" mediaType="text/plain" type="select">
        <option>small</option>
        <option selected="true">medium</option>
        <option>large</option>
```

```
    </input>
</function>
```

In this example, the parameter 'imageSize' is defined by a 'select' tag, and has only a limited number of valid possible options. Other Functions parameters, such as the parameter to set a dimmable light's level, use the 'input' tag to show that they take a broader range of input:

```
<function method="POST" mediaType="instance/config"
action="/rest/icontrol/sites/101/network/instances/133781220491568969
.0/points/" name="Set Points">
        <input name="isOn" mediaType="light/isOn"
type="text">true</input>
        <input name="level" mediaType="light/level"
type="text">100</input>
</function>
```

(The '100' in the input value above reflects the current value of the dimmer point, and is provided for convenience for any client that would like to display the Functions themselves in some sort of UI. It is not part of the actual Function definition.)

5.5.2 Invoking a Function

To invoke a Function, call the Function's action with the appropriate HTTP method and parameters. The response body is allowed to be XML, a simple string, a JPEG photo, an MPEG video clip or any other type of content – check the "Content-type" header if there's any uncertainty.

Note that the specific Function URI is indicated in the action attribute of the Function Descriptor.

5.6 Deltas – Device Changes in Real-Time

Delta events are an attribute of Icontrol's REST API that provides the client with real-time site updates. When a delta resource is requested, all device events are recorded via a response from the system. A deltas request will be left open for a given time frame (60 seconds) until a system event occurs, in which case the system will return information about the event. If no event occurs within the time frame, a time out response is given. By default, the system checks every 10 seconds to determine whether the time out has occurred. This setting is determined by the `asyncServlet.scavangeInterval` system property.

To request deltas, one should pass the following get request (site id of 1234, where site is synonymous with the subscriber's account)

```
GET /rest/async/icontrol/sites/1234/deltas
```

By requesting this URL, the system will continually poll the site for any events. If an event occurs, a response will occur and the deltas request will be closed. If there is a desire to continue to listen to events, the previous event's response should be processed and an identical deltas request should be issued.

In regards to API command requests, delta event responses provide an imperative functionality. The response of an API command request does not indicate whether the command was executed correctly or not. However, by listening to delta events, one can determine whether the CPE processed an executed the command. More specifically, to determine whether an API command was implemented successfully (or not) one should listen during and after a command is issued. An event/cpeCommand (indicating that the CPE has issued the command) should be observed, as well as an event related specifically to the specifics of the command (e.g. event/lighting). In this way, one can effectively associate an API command request with its implementation in real-time. Note that a deltas response does not unequivocally prove that a command has been successfully executed, as listening for deltas will indiscriminately return all events, regardless of what caused the event. For example, suppose a command was requested to turn on a light. A client listening for deltas events might then observe an event/lighting that indicates the specified light was turned on. This would lead to the conclusion that the CPE successfully processed and executed the command. However, it is possible that the light was physically turned on during the same time frame that the command should have been executed, thus meaning the light was turned on by a physical action and not the command. In conclusion, this example shows that faulty conclusions regarding command execution is possible, as delta events are not specifically tied to API command requests. Yet, by intuitively examining delta responses, an accurate conclusion can usually be made on whether an API command was executed successfully or not.

Below is an example of a typical deltas response. An API command requesting that a light be turned on is passed by the client, and the following is the delta response(s) that results.


```
[{"ts":1.38782692316e12,"instance":"133781220542999411.0","mediaType"
:"event\/lighting","metaData":[{"name":"isOn","value":"true"},
{"name":"level","value":"0"},{"name":"eventTime","value":"2013-12-
23T13:28:43-0600"}]}]
```

```
[{"id":"527519","ts":1.387826925531e12,"mediaType":"event\/cpeCommand
","metaData":[{"name":"commandType","value":"lightingUpdate"},
{"name":"success","value":"true"},{"name":"eventTime","value":"2013-
12-23T13:28:45-0600"}]}]
```


Here, ts is the time the event happened on an instance. Instance is the id of the instance. In meta data, isOn and level are both points of the instance.

In order to continue listening for incoming events, the client would process the above event and then issue the same deltas request again.

The Deltas Resource will also notify clients about what happened in a site. The following is an example of a zone event:


```
[{"id":"5677081","ts":1.33615807503e12,"name":"isFaulted","value":"tr
ue","instance":"112.0","channel":"B","mediaType":"event/zone","metaDa
ta":[{"name":"sensorNearFarSignal","value":"253/255"},
{"name":"sensorNearFarRF","value":"240/240"},
{"name":"sensorBatteryVoltage","value":"2874"},
{"name":"sensorTemperature","value":"2408"}]}]
```

5.7 Users

User resource represents a user that can access the Icontrol system. Users include administrators and subscribers. Here, we focus on subscriber user.

Here is an example User resource

```
GET /rest/<partnerid>/users/101

<user type="User" status="active" id="101"
href="/rest/icontrol/users/101">
    <emailAddress>xyz@icontrol.com</emailAddress>
    <login>xyz</login>
    <sites count="1">
        <site href="/rest/icontrol/sites/101">
            <createdAt>0</createdAt>
        </site>
    </sites>
    <password>a94a8fe5ccb19ba61c4c0873d391e987982fbbd3</password>
    <primary>true</primary>
</user>
```

A User resource contains information about a particular user, as well as subordinate resources showing the Sites the User is associated with. Each User resource has a unique userId, which is used to identify the resource in the API.

As a shortcut to get to the User resource associated with the request, the client can call the following:

```
GET /rest/<partnerId>/user/me
```

which will redirect to the correct User resource.

6 Use Cases - Security

6.1 Security Devices and Instances

Like other physical components of the Icontrol system, the security TouchScreen and its associated zones/sensors are represented as Devices and Instances within the REST API. You can identify which Devices and Instances represent the security system by looking at the mediaTypes of the Instances:

Physical Equipment	Instance mediaType
Security TouchScreen	instance/panel
Zone/Sensor	instance/sensor

6.2 Arming the Security Panel

The current arm state is in the security panel Instance's 'panel/armType' Point. The allowed values are 'away', 'stay', 'night' and null. A null arm type means the system is not armed.

To arm the panel, use the 'panel/arm' Function. It requires a security code and an arm type (one of the four non-null arm types). After the arm function is invoked, Client should listen to the deltas resource for an event of mediaType 'event/armDisarm' to make sure the system is armed.

6.3 Detecting Alarms

The typical way to detect alarm events – along with other real time events – is through the Deltas Resource (see "Deltas – Device Changes in Real-Time" on page 14). Using Deltas, a client makes repeated calls to the Deltas Resource as a way of listening for events on the network. If an alarm occurs, an 'event/securityStateChange' event with alarm status will be passed back to the caller. Here's a example showing the Deltas calls and responses during an alarm:

```
GET /rest/async/icontrol/sites/1234/deltas
```

```
[{"ts":1.33615807503e12,"mediaType":"event/securityStateChange","meta
Data":[{"name":"armType"},{"name":"status","value":"alarm"},
{"name":"trouble","value":"false"}]}]
```

If you simply need to know what the system alarm state is now, you can get the 'panel/status' point value for the security TouchScreen instance:

```
GET /rest/icontrol/sites/1234/network/instances/1234.0/points/status
```

The return can be one of the following values:

```
ready, notReady, armed, readyArmed, alarm, arming, entryDelay.
```

6.4 Security TouchScreen Trouble Conditions

There are a number of non-alarm trouble conditions that security TouchScreen can get into that need to be flagged to the end-user or CSR. These include low battery, AC power failure, etc.

Security TouchScreen troubles are represented by the 'troubles' point. Multiple troubles can exist at the same time. For example:

```
GET
/rest/icontrol/sites/1234/network/instances/1234.0/points/troubles
```

Returns:

```
ACPwrLoss,true;batLow,true
```

Here, the TouchScreen has two troubles, AC power loss and low battery. Both are critical.

Client can listen to the deltas resource for real-time zone troubles.

6.5 Security Zones

Security zones are listed as Instances. To get a list of Security zones, look for all Instances with mediaType 'instance/sensor'.

Here is an example door window zone:

```
<instance index="0"
href="/rest/icontrol/sites/1/network/instances/11-1080267560.0"
id="11-1080267560.0" mediaType="instance/sensor" tags="Zone"
status="ok">
<name>Front Door</name>
<points href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/">
<point mediaType="sensor/label"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/label" ts="1336761420518">test zone label</point>
<point mediaType="sensor/faulted"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/isFaulted" ts="1336761420518">false</point>
<point mediaType="sensor/bypassed"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/isBypassed" ts="1336761420518">false</point>
<point mediaType="sensor/type"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/type" ts="1336761420518">door</point>
<point mediaType="sensor/function"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/functionType" ts="1336761420518">armAWAY</point>
<point mediaType="sensor/troubles"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/troubles" ts="1336761420518"/>
<point mediaType="sensor/displayOrder"
href="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/displayOrder" ts="1336761420518">0</point>
</points>
<functions>
<function method="POST" mediaType="instance/config"
action="/rest/icontrol/sites/1/network/instances/11-
1080267560.0/points/" name="Set Points">
<input name="label" mediaType="sensor/label" type="text">test zone
label</input>            <input name="isBypassed"
mediaType="sensor/bypassed" type="text">false</input>
</function>
</functions>
</instance>
```

6.6 Zone Type and Function Types

Zone type and function type are identified by 'type' and 'functionType' points (see "Sensor/Zone Types" on page 48 and "Sensor/Zone Function Types" on page 49).

6.7 Zone Troubles

Like TouchScreen troubles, each Zone can have its own list of troubles. The troubles are identified by the 'troubles' point (see "Sensor/Zone Trouble Types" on page 47). Client can listen to the deltas resource for real-time zone troubles.

7 Use Cases - System Status, Monitoring, and Maintenance

7.1 Account History

7.1.1 Get Security Events

This call returns the security events that occurred over the last 24 hours.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/securityEvents/
```

Request Parameters

Parameter	Description	Example Value
maxResults	Indicates the maximum number of results to return.	10

Result

Returns an Events Array with HTTP Status 200.

7.1.2 Get Security Events by Day

This call returns a list of security-related system events and associated timestamps. You must specify start and end dates.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/securityEventsByDay
```

Parameter	Description
startDate	The start date, in the format YYYY-MM-DD. This value is required.
endDate	The end date, in the format YYYY-MM-DD. This value is required.
maxResults	Indicates the maximum number of results to return.

Result

Returns an Events Array with HTTP Status 200.

Note: For all alarm events, the output of the securityEventsByDay call also includes the `isTestAlarm` and `isAcked` properties.

Example Usage

```
GET /rest/<partner>/sites/<site>/securityEventsByDay?startDate=2013-01-28&endDate=2013-01-29&maxResults=100
```

7.1.3 Get Security Events by Date

The `securityEventsByDate` call is similar to `securityEventsByDay`, but it can be used to return events for specific zones.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/securityEventsByDate
```

Request Parameters

Parameter	Description
eventTypes	An optional comma-separated list of event types. Arm, disarm, zone, system, and alarm are the event types that can be passed with this request. If no value is specified, all event types are returned.
startDate	The start date, in the format YYYY-MM-DD. This value is required.
endDate	The end date, in the format YYYY-MM-DD. This value is required.
zoneIds	Returns events from the specified zone IDs only. **Note:** This parameter must be specified in all requests, but the value is evaluated only when zone is specified in the eventTypes parameter.
maxResults	Indicates the maximum number of results to return.

Example Usage

```
GET /rest/<partner>/sites/<site>/securityEventsByDate?startDate=2014-01-21&endDate=2014-01-22&zoneIds=1%2C1&maxResults=25
```

Result

Returns an Events Array with HTTP Status 200.

7.1.4 Get All Security and Home Automation Events

The events call returns the home automation and security events that occurred over the last 24 hours.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/events/
```

Request Parameters

Parameter	Description	Example Value
maxResults	Indicates the maximum number of results to return.	10

Result

Returns an Events Array with HTTP Status 200.

7.1.5 Get All Security and Home Automation Events by Day

This call returns a list of security and home-automation events for the specified date range.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/eventsByDay/
```

Request Parameters

Parameter	Description	Example Value
startDate	Events start date with format YYYY-MM-DD	2014-02-02
endDate	Optional parameter. Events end date with format YYYY-MM-DD. For performance reasons, a maximum of 2 days of history can be requested.	2014-02-03
maxResults	Maximum number of results to return	10

Results

Returns an Events Array with HTTP Status 200. Events list includes:

- Security events
- Home-automation events
- OnDemandImage events

Returns HTTP Status 404 for invalid site ID.

7.1.6 Get All Security and Home Automation Events by Date

This call returns security and home automation events over the specified dates.

API Call

```
GET https://[host]/rest/[partner]/sites/[siteId]/eventsByDate/
```

Request Parameters

Parameter	Description	Example Value
eventTypes	A comma- separated list of event types. Valid values are arm, disarm, system, zone, alarm, doorLock, cameraMotion, light, and thermostat. If no value is specified, all event types are returned.	zone, cameraMotion
startDate	The start date, in the format YYYY-MM-DD. This value is required.	2015-02-02
endDate	The end date, in the format YYYY-MM-DD. This value is required.	2015-02-03

Parameter	Description	Example Value
zoneIds	Returns events from the specified zone IDs only. **Note:** This parameter must be specified in all requests, but the value is evaluated only when zone is specified in the eventTypes parameter.	1,2
maxResults	Indicates the maximum number of results to return.	10

Result

Returns an Events Array with HTTP Status 200.

7.1.7 Events Array

The following example shows the output of a securityEvents, securityEventsByDate, or securityEventsByDay call.

Note: For all alarm events, the output of the securityEventsByDay call also includes the `isTestAlarm` and `isAcked` properties.

```
<entries count="51"
href="/rest/icontrol/sites/122/securityEventsByDay">
   <entry channel="C" ts="1359438249409" instance="112.0"
name="isFaulted" id="7817837" mediaType="event/zone">
      <value>true</value>
      <metaData name="sensorNearFarSignal"/>
      <metaData name="sensorNearFarRF"/>
      <metaData value="2013-01-28T23:44:09-0600" name="eventTime"/>
   </entry>
   <entry channel="C" ts="1359436950671" id="7817811"
mediaType="event/communication">
      <value>cellLossRes</value>
      <metaData value="true" name="isTotalStateChange"/>
      <metaData value="total" name="showTotalOnly"/>
      <metaData value="2013-01-28T23:22:30-0600" name="eventTime"/>
   </entry>
   <entry channel="S" ts="1359435115713" id="7817729"
mediaType="event/communication">
      <value>cellLoss</value>
      <metaData value="true" name="isTotalStateChange"/>
      <metaData value="total" name="showTotalOnly"/>
      <metaData value="2013-01-28T22:51:55-0600" name="eventTime"/>
   </entry>…
</entries>
```

The following example shows the output of an events, eventsByDate, or eventsByDay call.

```
<entries count="2" href="/rest/icontrol/sites/122/events">
   <entry channel="B" ts="1393467573123" instance="12125485.7.0"
name="isInMotion" id="9542" mediaType="event/cameraMotion">
      <value>false</value>
      <metaData value="2014-02-26T20:19:33-0600" name="eventTime"/>
   </entry>
   <entry channel="B" ts="1393467503191" instance="111.0"
name="isFaulted" id="76101" mediaType="event/zone">
      <value>true</value>
      <metaData value="3000" name="sensorBatteryVoltage"/>
      <metaData value="2430" name="sensorTemperature"/>
      <metaData value="255/255" name="sensorNearFarSignal"/>
      <metaData value="-30/-18" name="sensorNearFarRF"/>
      <metaData value="2014-02-26T20:18:23-0600" name="eventTime"/>
   </entry>
</entries>
```

7.2 Upgrading Firmware

The `firmwareUpdate` API upgrades the firmware version on the CPE for both Converge and Touchstone. For more information about this API function, see:

- "Function media types" on page 35
- The "Update Firmware" section here: https://share-icontrol.atlassian.net/wiki/display/CSKB/7.3+Quadra+Core+-+Miscellaneous+REST+APIs

7.3 Thermostat Schedules

The thermostat schedule comes with 4 distinct time slots (Wake, Day, Evening, and Sleep). These time points are pre-determined; one cannot add more, delete, or modify the names of these time points. However, the time at which these zones are triggered, as well as the specific temperature changes at these time, can be modified. In addition, a heat schedule and a cool schedule are both provided that follow the same format but are distinct. To create a rule, the following URL should be passed with a JSON string as a parameter:

```
POST https://[HOST]/rest/icontrol/sites/
[siteId]/network/instances/141021006497617942 6.0/functions/ruleCreate
```

The JSON string should be formatted as follows:

```
{"rulename":"Thermostat 2 Schedule","entries":
[{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,THU,FR
I,SAT 6:00"},
{"mode":"cool","temperature":2944,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 18:00"},
{"mode":"cool","temperature":2777,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"},
{"mode":"heat","temperature":2111,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 6:00"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"heat","temperature":2111,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 18:00"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"}],"instance":"1410210064976179426.0"}
```


The rule name can be changed to whatever is desired. Also, the temperature and time can be modified as desired. Temperature inputs are measured in Celsius*100 (for example, 3000 represents 30 degrees Celsius), and time is measured in military time. The instance should be the same as the instance of the thermostat that is being modified. In this example, each time slot (Wake, Day, Evening, and Sleep) are given the same temperature and time value for every day of the week. However, the time and temperature for each time slot can be modified on a per day basis. To help explain how this can be done, the following example JSON string is shown for reference:


```
{"rulename":"Thermostat 2 Schedule","entries":
[{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,FRI
6:00"},
{"mode":"cool","temperature":3000,"timeSlot":"THU 4:00"},
{"mode":"cool","temperature":2555,"timeSlot":"SAT 5:00"},
{"mode":"cool","temperature":2944,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 18:00"},
{"mode":"cool","temperature":2777,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"},
{"mode":"heat","temperature":2111,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 6:00"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"heat","temperature":2111,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 18:00"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"}],"instance":"1410210064976179426.0"}
```

The above example is almost identical to the first JSON string. However, the Wake time slot for the COOL schedule has been modified by creating a unique Wake time on Thursday and Saturday. Moreover, the Wake temperature on Thursday has been increased to 3000. The way to modify on a day to day basis is to first exclude the unique days that will have a different time/temperature. If every day is different, then each day should have a unique bracket. Be sure not to duplicate days for any of the given time slots (Wake, Day, Evening, Sleep). Once all the unique days have been initialized, simply modify the time/temperature as desired. Here is another example that will create a unique time/temperature in the HEAT schedule in the evening time slot. The evening portion (in the HEAT schedule) has been sectioned off for clarity:


```
{"rulename":"Thermostat 2 Schedule","entries":
[{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,THU,FR
I,SAT 6:00"},
{"mode":"cool","temperature":2944,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"cool","temperature":2555,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 18:00"},
{"mode":"cool","temperature":2777,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"},
{"mode":"heat","temperature":2111,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 6:00"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 8:00"},
{"mode":"heat","temperature":2000,"timeSlot":"SUN 18:00"},
{"mode":"heat","temperature":2100,"timeSlot":"MON 18:15"},
{"mode":"heat","temperature":2200,"timeSlot":"TUE 18:30"},
{"mode":"heat","temperature":2300,"timeSlot":"WED 18:45"},
{"mode":"heat","temperature":2400,"timeSlot":"THU 19:00"},
{"mode":"heat","temperature":2500,"timeSlot":"FRI 19:15"},
{"mode":"heat","temperature":2600,"timeSlot":"SAT 19:30"},
{"mode":"heat","temperature":1666,"timeSlot":"SUN,MON,TUE,WED,THU,FRI
,SAT 22:00"}],"instance":"1410210064976179426.0"}
```


Updating a pre-existing temperature schedule is almost identical to creating a temperature schedule. The same JSON format as above should be utilized, even if only one specific time slot on a given day is desired to be changed. The important caveat to note is that the JSON string has to be complete, meaning that every time slot and day for each schedule mode (heat and cool) has to be accounted for from within the JSON string. If this is not the case, a 500 server error will be thrown. Other than that, one can modify as desired , whether it be just one time slot and day, multiple time slots and days, or all time slots and days. Any of the JSON strings shown above could be used to update a pre-existing rule. Also note that the "rulename" needs to be identical to the thermostat schedule/rule that one desires to change. Lastly, the following URL should be called in order to request that a thermostat schedule be updated. As before, one should pass the JSON string along with the POST request:

`POST` https://[HOST]/rest/icontrol/sites/[siteId]/network/instances/1410210064976179426.0/functions/ruleContentUpdate

Appendix A: Function Descriptors

This appendix describes the format of the Function resource descriptors in the API. By looking at the Function descriptors, it is often possible to tell which parameters a Function requires as well as the allowed values of those parameters.

A Function descriptor in XML looks like the following:

```
<function method="POST" mediaType="panel/arm"
action="/rest/icontrol/sites/101/network/instances/100026f3001f3b.0/f
unctions/arm" name="arm">

    <input name="code" mediaType="text/plain" type="text"/>
    <input name="armType" mediaType="text/plain" type="select">
        <option>away</option>
        <option selected="true">stay</option>
        <option>night</option>
    </input>
</function>
```

Function descriptors indicate how to invoke various resources that are not a simple link away. They are analogous to HTML forms and the syntax is borrowed somewhat from them.

Functions can have input child elements which dictate any parameters that can be supplied to the Function. Each input element has a type attribute which may either be text or select and a name attribute which is the name of the parameter to be supplied in the request.

For example, an input tag of type text:

```
<input type=”text” name="level"/>75</input>
```

This input tag shows that the Function expects a text parameter with the name level, and that the default value for that parameter is 75. Often, but not always, this default value is the current value of that parameter for the particular resource. Note that not all input tags will have a default value.

Text input tags may have a mediaType attribute as well, used to indicate the specific type of value.

Here’s an example of a ‘select’ input tag:

```
<input type=”select” name=”level”>
   <option selected=”true”>high</option>
   <option>medium</option>
   <option>low</option>
</input>
```

Select tags are used to indicate there is a discrete set of values that the Function will accept. The selected option designates the default value (often, but not always, the current value).

Appendix B: HTTP Headers

The Icontrol REST API uses both standard and custom HTTP headers extensively. Below is a list of the headers used in the API.

Note that custom (X-xxxx) headers may also be passed as request parameters if the caller cannot support request headers.

B.1 Request Headers

B.1.1 X-format: response format

The optional 'X-format' header controls the http response format. If the 'X-format' header is not specified in an http request, the response will be in xml format with the custom headers listed as response headers. For example:

```
GET /icontrol/users/1
X-format: XML
```

Will return

```
<user type="User" status="active" id="1"
href="/rest/icontrol/users/1">
    <emailAddress>johndoe@ucontrol.com</emailAddress>
    <login>junit</login>
    <sites count="1">
        <site href="/rest/icontrol/sites/1"/>
    </sites>
     <password>c340cv261c4c0873d391e987982fbbd3</password>
    <passwordHint>hint</passwordHint>
    <primary>true</primary>
</user>
```

If the caller cannot handle response headers, the request header 'X-format: xml' can be used to get a response where both the response code and the response headers are contained within the body of the response.

If we specify 'X-format: XML' in the above request, then the return will be:

```
<response code="403">
<header name="X-version">3.5</header>
<header name="Cache-Control">no-cache</header>
<body>
<user type="User" status="active" id="1"
href="/rest/icontrol/users/1">
    <emailAddress>johndoe@ucontrol.com</emailAddress>
    <login>junit</login>
    <sites count="1">
        <site href="/rest/icontrol/sites/1"/>
```

```
    </sites>
     <password> c340cv261c4c0873d391e987982fbbd3</password>
    <passwordHint>hint</passwordHint>
    <primary>true</primary>
</user>
</body>
</response>
```

For some clients, JSON format is preferred. 'X-format: json' can be used to format the response in JSON.

Here is the response in JSON format:

```
{"href":"/icontrol/users/1","emailAddress":"johndoe@ucontrol.com","lo
gin":"junit","sites":{"site":
[{"href":"/rest/icontrol/sites/1"}],"count":1},"password":"
c340cv261c4c0873d391e987982fbbd3","passwordHint":"hint","primary":tru
e,"id":"1","status":"ACTIVE","type":"User"}
```

B.1.2 X-AppKey: Application Key

This header is required for all clients. Client must use a valid application key assigned to the client application by Icontrol or its customers.

B.1.3 X-HTTP-Override-Method: Overriding HTTP Methods

Due to various proxies, HTTP packages and other limitations it is sometimes necessary to override the HTTP method provided in the request either via the HTTP headers or as last resort a query parameter.

For example, to delete a user from the system call

```
DELETE /rest/icontrol/users/314149
```

However a proxy may choose to block DELETE requests.

To override the method with a header, pass an "X-HTTP-Override-Method" header with the value of the desired method.

```
GET /rest/icontrol/users/314159
X-HTTP-Override-Method: DELETE
```

To override the method with a query parameter pass in a parameter of "X-method" with the desired method as its value. Care should be taken not to provide a hyperlink with this parameter as various search spiders and other tools will assume that the GET request is safe for traversal and thus (if authorized) may inadvertently delete resources.

B.1.4 X-login: Authentication login name

User/login name of the user calling the API. All callers of the API must initially authenticate with username and password.

B.1.5 X-password: Authentication password

Password of the user calling the API. All callers of the API must initially authenticate with username and password, however, the password may be exchanged for a session token.

B.1.6 X-requestID: Match request and response

The X-requestID request header allows clients to easily match request and response. The client can put arbitrary data in the X-requestID request header. This header and data will then be echoed back to the client in the response.

B.1.7 X-loginEncoded: Base64-encoded strings

To use non-ASCII characters in X-login and X-password, values for X-login and X-password must be sent as Base64-encoded strings. The "X-loginEncoded" header informs the server about the string encoding status of X-login and X-password. Possible values for "X-loginEncoded" are true or false, as described in the following table. If the client uses a non UTF-8 character set for an encoded string, the client must send the character set in the "Content-Type" header.

X-loginEncoded Values	Description
true	The server treats X-login and X-password as a Base64-encoded string and tries to decode the string before authenticating it.
false	The server treats X-login and X-password as a plain text string.
Not present	Same as false

B.1.8 X-expires and X-token: Token Expiration for REST Modules

When the `rest.csrf.token.enabled.modules` server property is enabled, the X-expires header is required to generate a CSRF token. To create a token, the client needs to pass the X-expires header with a value in milliseconds (maximum 10 minutes or 60,000 milliseconds). The server returns a response that contains the X-token header with the token value. Subsequent requests use the X-token header to pass in the token value, and the token expiration timer resets to the defined X-expires value with each request.

For example, the request:

```
GET 'https://localhost/rest/icontrol/login'
X-login: bob
X-password: test
X-expires: 60000
X-AppKey: defaultKey
```

returns a response that contains the header with the token value:

X-token: t6478392383397688788

For more information, see the "rest.csrf.token.enabled.modules" section in *System Operations Guide*.

B.2 Response Headers

B.2.1 Cache-Control: Caching Responses

The Cache-Control header should be present in all REST API responses. Normally it will be set to “no-cache”.

B.2.2 ETag

The ETag header may be set in responses that include a body that is not XML or represents information that could potentially be cached. Clients can save the data in the ETag, and on subsequent requests, send the request header 'If-None-Match' with the ETag data. If the requested resource has not changed, the system will return code 304 ('unmodified') rather than returning a new copy of the data.

This is the standard ETag mechanism defined by the HTTP standard.

B.2.3 Location: Redirection or Monitoring

The Location header may be present in responses that redirect to another resource (3xx class response codes) or if the initial request will be fulfilled asynchronously. In the asynchronous case the request cannot complete immediately, so the system returns code 201 or 202 and the resource indicated by the location header can be queried to monitor the status of the outstanding request.

B.2.4 X-error: Custom Errors

The X-error response header may be used to override the response code to provide more details about the failure. The value of the X-error header is a standard UCE code. For more information, see the "Exception Codes (UCE)" section in *System Operations Guide*.

B.2.5 X-existing: Request Conflicted with an Existing Resource

This response header is often seen when a request to create or modify a resource returns a code 409, 'resource exists'. In the Icontrol system, many resources have unique IDs, and devices must also have unique names. The value of X-existing will be 'id' if the request conflicted with an existing resource ID, or 'name' if the request attempted to give an already-used name to a resource.

B.2.6 X-requestID

Header echoed back to requestor, used to match request and response. If a request contains an X-requestID header, the response will also contain this header. The contents of the response header will be whatever was initially specified in the request.

B.2.7 X-version

The X-version header will be set to the version of the REST API being implemented. The current version of the REST API is "3.5".

Appendix C: MediaTypes

Most resources and Function inputs have a “mediaType” attribute. This attribute indicates what type of thing the resource is.

The following is a list of the mediaTypes used by the REST API, organized by resource types.

C.1 Instance Types

Instance Type	Comments
instance/panel	The TouchScreen
instance/sensor	Sensor device
instance/light	Light device
instance/lightSwitch	Light switch device
instance/lightDimmer	Light dimmer device
instance/thermostat	Thermostat
instance/camera	Camera
instance/peripheral	Peripheral device
instance/doorLock	Door lock
instance/virtualDevice	Cloud object device

C.2 Point media types: TouchScreen

Instance Type	Comments/Restrictions
panel/armType	Values include away, stay, and night
panel/status	Values include ready, notReady, armed, readyArmed, alarm, arming, and entryDelay
panel/bbConnected	Boolean
panel/cellularConnected	Boolean
panel/inTrouble	Boolean
panel/inTestMode	Boolean
panel/troubles	TouchScreen troubles (see "Security Panel Trouble Types" on page 46)
panel/apps	The apps installed on the TouchScreen
panel/alarm	The alarm type

C.3 Point media types: Sensor

Instance Type	Comments
sensor/label	The sensor’s display name

Instance Type	Comments
sensor/isFaulted	Boolean
sensor/isBypassed	Boolean
sensor/type	Type of sensor (see "Sensor/Zone Types" on page 48)
sensor/functionType	Zone function type
sensor/troubles	Sensor troubles (see "Sensor/Zone Trouble Types" on page 47)
sensor/displayOrder	The order in which the sensor is displayed
sensor/temperature	Integer
sensor/batteryVoltage	Integer
sensor/nearFarRF	Sensor near/far RF strength
sensor/nearFarSignal	Sensor near/far signal strength
sensor/sensorType	String
sensor/serialNumber	String

C.4 Point media types: Camera

Point media types for camera instance

Instance Type	Comments
camera/label	The camera's display name
camera/isVideoRecordable	Boolean
camera/troubles	The camera's trouble indicators
camera/motionSensitivity	Sensitivity level
camera/videoFormat	Video format
camera/videoCodec	Video codec
camera/motionCapable	Boolean

C.5 Point media types: CameraMotion

Instance Type	Comments
camera/inMotion	Boolean. Indicates whether the camera has detected motion.
camera/motionSensitivity	Sensitivity level

C.6 Point media types: Light

Point media types for a lighting instance

Instance Type	Comments
light/label	The light's display name

Instance Type	Comments
light/dimAllowed	Boolean
light/type	Possible values are onOffLight, dimmableLight, onOffSwitch, dimmableSwitch
light/isOn	Boolean
light/level	0 - 256
light/troubles	The light's trouble indicator
light/energyMgmtEnabled	Boolean
light/energyUsage	Energy usage value

C.7 Point media types: Peripherals

Instance Type	Comments
peripheral/troubles	Trouble indicator
peripheral/type	The type of peripheral. ❑ siren ❑ keyfob ❑ keypad ❑ takeoverKeypad ❑ wifiRepeater ❑ router ❑ zigbeeVSMCT101_1Btn ❑ zigbeeVSMCT102_2Btn ❑ zigbeeVSMCT103_3Btn ❑ zigbeeVSMCT104_4Btn ❑ zigbeeVSMCT124_4Btn ❑ zigbeeVSMCT220EmerBtn ❑ zigbeeVSMCT241Pendant

C.8 Point media types: Thermostat

Instance Type	Comments
thermostat/label	The display name

Instance Type	Comments
thermostat/systemMode	Mode of the thermostat
thermostat/temperature	The current temperature
thermostat/fanMode	On/auto
thermostat/holdMode	Boolean
thermostat/heatSetPoint	The heater target temperature
thermostat/coolSetPoint	The cooling target temperature
thermostat/troubles	The thermostat's trouble indicator
thermostat/currentPreset	Night or comfort

C.9 Point media types: Door Lock

Instance Type	Comments
doorLock/label	The door lock's display name
doorLock/isOn	Boolean. True if door is locked, False if it is unlocked **IMPORTANT:** This instance type is deprecated. Use doorLock/locked instead.
doorLock/troubles	The door lock's trouble indicator
doorLock/locked	Boolean. True if door is locked, False if it is unlocked

C.10 Point media types: Cloud Objects

Instance Type	Comments
cloudObject/troubles	Troubles vary per partner integration.

C.11 Function media types

Function media types

Instance Type	Comments
panel/arm	Arm the TouchScreen
panel /disarm	Disarm the TouchScreen
panel/addUserCode	Add a user code
panel/updateUserCode	Update user code
panel/deleteUserCode	Delete a user code
panel/getUserCodes	Retrieve user codes
camera/access	Access a camera
camera/snapshot	Take a snapshot

Instance Type	Comments
camera/directSnapshot	
camera/videoClip	Take a video
camera/accessTerminate	Terminate access
thermostat/status	Get the status
panel/reboot	Reboot the CPE
panel/firmwareUpdate	Update the CPE to the latest active firmware
panel/updateZoneDisplayOrder	Update the zone display order (Insight only)
camera/delete	Remove the camera from the system (Insight only)
camera/isAlive	Boolean. True if camera is online, False if it is unresponsive **IMPORTANT:** On Touchstone, this function may return HTTP status 504 (gateway timeout) if no response is received within 60 seconds. This indicates that the camera may not be online yet (equivalent to isAlive=false). If you receive this timeout, you can retry the call until you receive a response.
lighting/delete	Remove the light from the system (Insight only)
thermostat/createScheduleRule	Create a thermostat scheduling rule
thermostat/updateScheduleRule	Update a thermostat scheduling rule
thermostat/deleteScheduleRule	Delete a thermostat scheduling rule
thermostat/getScheduleRule	Get a thermostat scheduling rule
thermostat/delete	Remove the thermostat from the system (Insight only)
sensor/delete	Remove the sensor/zone from the system (Insight only)
sensor/updateZoneTypeAndFunction	Update the zone type and function type (Insight only)

C.12 Event media types

Instance Type	Comments
event/alarmSessionComplete	
event/alarmSessionComplete	
event/alarmTestMode	
event/armDisarm	
event/communication	
event/doorLock	
event/doorlockAdded	

Instance Type	Comments
event/doorlockDeleted	
event/doorlockTrouble	
event/doorlockUpdated	
event/homeAutomation	
event/lightingTrouble	
event/onDemandImage	
event/panic	
event/peripheralTrouble	
event/smashAndGrab	
event/systemTrouble	
event/thermostatTrouble	
event/zone	
event/zoneTrouble	

C.13 CPE Event media types

CPE events are events related to changes to the system.

Instance Type	Comments
event/accountChangeHistory	
event/cameraAccess	
event/cellularConnectionTypeUpdated	
event/cpeCommand	
event/firmwareUpdateJobCompleted	
event/firmwareUpdateJobCreated	
event/firmwareUpdateJobDownloadCompleted	
event/firmwareUpdateJobQueued	
event/firmwareUpdateJobRebootCompleted	
event/firmwareUpdateJobRequested	
event/firmwareUpdateJobUpdateStarted	
event/keypadUserAdded	
event/keypadUserDeleted	
event/keypadUserUpdated	
event/lighting	
event/lightingAdded	

Instance Type	Comments
event/lightingDeleted	
event/lightingUpdated	
event/peripheralAdded	
event/peripheralDeleted	
event/peripheralUpdated	
event/ruleAdded	
event/ruleDeleted	
event/ruleUpdated	
event/sceneUpdateEvent	
event/securityStateChange	
event/thermostat	
event/thermostatAdded	
event/thermostatDeleted	
event/thermostatUpdated	
event/widgetAdded	
event/widgetDeleted	
event/widgetReordered	
event/widgetUpdated	
event/zoneAdded	
event/zoneDeleted	
event/zoneUpdated	

Appendix D: Hyperlinks

All resources exist at or are accessible from URIs. Throughout the API it is frequently necessary for one resource to refer to one or more other resources. This is most often done via hypertext. While the API uses descriptive hierarchical URI structures, care must be taken if the client attempts to construct a URI's path programmatically.

All hyperlinks are indicated by the href attribute within the XML document. Hyperlinks may appear in three forms: Relative, Local, or Global.

D.1 Global Hyperlinks

Hyperlinks beginning with "http://" or "https://" are global and absolute. These are rare and usually are informational. When used, the content of the href attribute must be taken as the URI.

D.2 Local Hyperlinks

If a hyper link begins with a "/" it is assumed to be local and it is formed by replacing the path (and query) part current document's URI with the content of the href attribute. See the hyperlink examples below.

D.3 Relative Hyperlinks

If a hyperlinks does not begin with one of the three strings "http://", "https://", or "/" as describe above, then it is a relative hyperlink. The hyperlink refers to the current document's URI appended with "/" and the content of the href attribute. See the hyperlink examples below.

D.4 Hyperlink Examples

Here are some examples of hyperlinks:

```
GET https://icontrol.com/rest/icontrol/

<sites>
<site href="sites/1234"/>
<site href="/rest/example/sites/1234"/>
<site href="http://example.com/rest/example/sites/1234"/>
</sites>
```

In this example, the first URI would resolve to `https://icontrol.com/rest/icontrol/sites/1234`, the second would resolve to `https://icontrol.com/rest/example/sites/1234` and the third would resolve to `http://example.com/rest/example/sites/1234`.

Appendix E: Developer Tools

A REST API browser web app can be deployed on development servers that have Icontrol Converge product installed. This browser allows developer to visualize the API using common browser as the GUI.

Command line tool like curl (http://curl.haxx.se/) can also be used. In the following examples, the server is assumed to be icontrol.com, the partner identifier is icontrol, the user login is test, and the user password is pass.

Before getting started, create an alias for starting curl:

```
# alias getit=’ curl -s -L -H "X-login: test" -H "X-password: pass" –
H “X-AppKey: defaultKey”’
```

First, obtain the user account information:

```
# getit https://icontrol.com/rest/icontrol/login | xml fo

<?xml version="1.0" encoding="UTF-8"?>
<login type="INTERACTIVE">
  <displayName>bob</displayName>
  <user href="/rest/icontrol/users/101"/>
  <site href="/rest/icontrol/sites/101"/>
</login>
```

Then get information about the site:

```
# getit https://icontrol.com/rest/icontrol/sites/101 | xml fo

<site id="101" href="/rest/icontrol/sites/101">

    <functions count="5">
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource by date range
and event types." name="alarmSessions"
action="/rest/icontrol/sites/101/alarmSessions" method="GET">
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the camera images and videos."
name="cameraImagesAndVideos"
action="/rest/icontrol/sites/101/cameraImagesAndVideos" method="GET">
            <input required="true" description="start record no"
type="text" name="start"/>
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
```

```
        </function>
        <function gates="Admin,Self" description="Get the camera
images and videos." name="fetchImage"
action="/rest/icontrol/sites/101/fetchImage" method="GET">
            <input required="true" description="imageId" type="text"
name="imageId"/>
            <input required="true" description="imageType"
type="text" name="imageType"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource."
name="securityEvents"
action="/rest/icontrol/sites/101/securityEvents" method="GET">
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
        <function gates="Admin,Self" outputType="entries"
description="Get the security events for site resource by date range
and event types." name="securityEventsByDate"
action="/rest/icontrol/sites/101/securityEventsByDate" method="GET">
            <input description="comma separated event types"
type="text" name="eventTypes"/>
            <input required="true" description="start date"
type="text" name="startDate"/>
            <input required="true" description="end date" type="text"
name="endDate"/>
            <input required="true" description="comma separated list
of zone ids" type="text" name="zoneIds"/>
            <input required="true" description="maximum no of records
to fetch" type="text" name="maxResults"/>
        </function>
    </functions>
    <extId>exr474170</extId>
    <cmsInfo href="/rest/icontrol/sites/101/cmsInfo"/>
    <profile>
        <setting>
            <country>US</country>
            <locale>en</locale>
            <temperatureUnit>F</temperatureUnit>
            <postalCode>78758</postalCode>
        </setting>
    </profile>
    <timezone>US/Central</timezone>
    <createdAt>1324395528087</createdAt>
```

```
    <network href="/rest/icontrol/sites/101/network"/>
    <siteUsers count="0" href="/rest/icontrol/sites/101/users"/>
    <accountStatusV>activated</accountStatusV>
    <activationCode>338154</activationCode>
    <address href="/rest/icontrol/sites/101/address">
        <verified>false</verified>
    </address>
    <suspended>false</suspended>
    <active>true</active>
    <internal>false</internal>
    <readyForRMA>false</readyForRMA>
    <source>internal</source>
    <contacts count="0" href="/rest/icontrol/sites/101/contacts"/>
</site>
```

The following sample contains a list of devices:

```
# getit https://icontrol.com/rest/icontrol/sites/101 | xml fo

<devices count="5" href="/rest/icontrol/sites/101/network/devices">

    <functions count="1">
        <function gates="Admin,Self" outputType="devices"
description="Find devices by siteId and type." name="find"
action="/rest/icontrol/sites/101/network/devices/find" method="GET">
            <input required="true" description="comma delimited list
of device type" type="text" name="type"/>
        </function>
    </functions>
    <device id="101" hardwareId="1030918ff13177" firmwareVersion="8_
0_0_000000_201607111519" hardwareVersion="9" type="ts"
version="internal_8_0_0_000000_201607111519" id="2669" model="TCA203"
manufacturer="Technicolor" technology="zigbee"
href="/rest/icontrol/sites/101/network/devices/1030918ff13177">
        <instances count="1">
            <instance
href="/rest/icontrol/sites/101/network/instances/1030918ff13177.0"/>
        </instances>
    </device>
    <device id="241" hardwareId="111" status="online"
firmwareVersion="4.0.2" serialNumber="3781220499046587" type="sensor"
technology="zigbee"
href="/rest/icontrol/sites/101/network/devices/111">
        <name>zone door</name>
```

```
        <instances count="1">
            <instance
href="/rest/icontrol/sites/101/network/instances/111.0"/>
        </instances>
    </device>
    <device id="202" hardwareId="12101.5" status="online"
firmwareVersion="v1.0.00 build 08" serialNumber="259c7ffd8c"
type="camera" model="WVC80N" manufacturer="linksys" technology="wifi"
href="/rest/icontrol/sites/101/network/devices/12101.5">
        <name>My Camera</name>
        <instances count="1">
            <instance
href="/rest/icontrol/sites/101/network/instances/12101.5.0"/>
        </instances>
    </device>
    <device id="102" hardwareId="133781220491568969" status="online"
firmwareVersion="6" hardwareVersion="7" type="lighting"
model="13531996" manufacturer="CentraLite Systems"
technology="zigbee"
href="/rest/icontrol/sites/101/network/devices/133781220491568969">
        <name>light 1 dimmable</name>
        <instances count="1">
            <instance
href="/rest/icontrol/sites/101/network/instances/133781220491568969.
0"/>
        </instances>
    </device>
    <device id="103" hardwareId="1410210064976009507" status="online"
firmwareVersion="1.92" type="thermostat" technology="zigbee"
href="/rest/icontrol/sites/101/network/devices/1410210064976009507">
        <name>Thermostat</name>
        <instances count="1">
            <instance
href="/rest/icontrol/sites/101/network/instances/1410210064976009507.
0"/>
        </instances>
    </device>
</devices>
```

Appendix F: Response Codes

The following table contains HTTP response codes that are used by the API.

Code	Short Meaning	Description
200	OK	Everything went ok
201	Created	The requested resource was created
202	Accepted	The request was accepted and is now in progress.
204	No Content	Everything went ok and there was nothing to return.
400	Bad Request	This error either means that the URL was not constructed correctly (bad syntax), the wrong request method was used, or that the REST framework does not provide functionality for the URL requested.
401	Unauthorized	Not authorized. Authorization is needed to invoke this URI.
403	Forbidden	User does not have permission to access this resource This error code occurs when a valid request is sent (with valid credentials) but the server declines to respond. This error code could be due to a Converge operation being performed on the Touchstone platform, or vice versa.
404	Not Found	The requested resource was not found
405	Method not allowed	The resource exists but cannot perform the requested method. Check the Allows: header for a list of allowed methods
408	Timeout	The resource exists, but the requested method timed out before completion. Due to compatibility issues with browsers (IE) that insist on automatically retrying responses that have an HTTP code of 408, this code will only be returned as 'http.408' through an X-error header on a code 500 response.
409	Conflict	The resource that the call was attempting to create already exists, or creating it would conflict with an existing resource. Often seen when renaming a device, since device names must be unique.
500	Internal server error	Something on the server failed. The request may be repeated after a delay.
502	Internal server error	Something on the server failed. The request may be repeated after a delay.
503	Temporarily Unavailable	The requested resource (CPE) is not currently available.

Appendix G: Security Panel Status

The following are the allowed values for security panel status, which can be found in the security panel instance's Point with mediaType 'panel/status'.

Status	Description
ready	The system in ready state.
notReady	Cannot arm because a zone is opened (and not bypassed)
armed	The system is armed
alarm	The system is in alarm
arming	The system is arming
entryDelay	The system in in entry delay

Appendix H: Security Panel Trouble Types

There are various trouble conditions that a security panel can have. The panel's Instance contains a Point of type 'panel/troubles', which contains a semicolon separated list of all current trouble conditions. The conditions themselves are represented as a pair of trouble type and a critical flag: trouble1,true;trouble2,false.

The set of trouble conditions is listed below.

Trouble Condition	Cause
ACPwrLoss	AC power loss
batLow	The battery is low
comLoss	Loss of communication with TouchScreen
replaceBat	The battery needs to be replaced
ACPwrTamp	AC power is tampered
batRemoved	The battery is removed
batChargeError	Problem charging battery
zigbeeTransceiverJam	Zigbee transceiver is jammed
gen	Generic trouble with the TouchScreen
routerTrouble	Trouble with the router
software	Software trouble on the TouchScreen, such as a core dump

Appendix I: Sensor/Zone Trouble Types

There are various trouble conditions that a security sensor/zone can have. The sensor/zone Instance contains a Point of type 'sensor/troubles', which contains a semicolon separated list of all current trouble conditions. The conditions themselves are represented as a pair of trouble type and a critical flag: trouble1,true;trouble2,false.

The set of trouble conditions is listed below.

Trouble	Description
sensCom	Sensor communication
senTamp	Sensor tampered
zoneSwingerShutdown	Zone in swinger shutdown mode
zigbeeSenTest	Zigbee sensor in test
zigbeeSenTamp	Zigbee sensor tampered
zigbeeSenLowBat	Zigbee sensor low battery detected
zigbeeSenFailBat	Zigbee sensor battery failed
zigbeeSenJam	Zigbee sensor jammed
zigbeeSenDirty	Zigbee sensor dirty
zigbeeSenLowTemp	Zigbee sensor low temperature
zigbeeSenHighTemp	Zigbee sensor high temperature
zigbeeSenBootloadFail	Zigbee sensor bootload failure
takeoverEol	End of line resistor not present on a Panel interface module (PIM) sensor
takeoverTamp	Tampered PIM sensor
takeoverLowBat	Low battery detected on a PIM sensor
takeoverFailBat	PIM battery failure on a PIM sensor
takeoverJam	PIM sensor jammed
takeoverNoPower	PIM sensor lost power
takeoverDirty	PIM sensor dirty
takeoverLowTemp	PIM sensor low temperature
takeoverHighTemp	PIM sensor high temperature
takeoverBootloadFail	PIM sensor bootload failure
unknown	Unknown

Appendix J: Sensor/Zone Types

The sensor/zone Instance contains a Point of type ‘sensor/type’. The current supported types are listed below:

- door
- window
- motion
- panic
- medical
- environmental
- glassbreak
- carbonMonoxide
- duress
- smoke
- water

Appendix K: Sensor/Zone Function Types

The sensor/zone Instance contains a Point of type 'sensor/function' that describes the function type of the sensor/zone. The current supported function types are listed below:

Function	Description
entryExit	For doorways that are used to enter and exit the premises and windows. If the system is armed, an alarm is sent if a valid key pad code is not entered.
perimeter	If faulted when the system is armed or during an Entry/Exit delay, an alarm is sent immediately.
interiorFollower	Monitors the internal living spaces of the premises and triggers an immediate alarm if the system is armed in Away mode.
troubleDayAlarmNight	Provides an instant alarm if faulted when armed in Alarm Away mode.
silent24Hr	Usually assigned to a zone containing an emergency button. Sends a report to the central station, but provides no keypad display or sound.
audible24Hr	Usually assigned to a zone containing an emergency button. Sends a report to the central station and provides an alarm sound at the keypad as well as an audible external alarm.
fire24Hr	Generates an immediate fire alarm if triggered.
interiorWithDelay	Provides entry delay (using the programmed entry time), if tripped when the panel is armed in the Away mode.
inform24Hr	When this security zone is tripped, there is never an alarm. However, an event is recorded in the history, and the TouchScreen emits a configured sound.
armSTAY	Only issues alarm if the system is armed in Arm Stay mode
armAWAY	Only issues alarm if the system is armed in Arm Away mode
disarm	System was disarmed
silentBurglary	24-hour Silent Alarm Usually assigned to a zone containing an emergency button.

Appendix L: Sensor Types

The instance/sensor Instance contains a Point of the type 'sensor/sensorType'. The current supported types are listed below:

- dryContact
- motion
- smoke
- glassBreak
- water
- gas
- vibration

The following types are currently supported, but will be deprecated in a future release:

- zigbeeDoorWindow
- zigbeeSmoke
- zigbeeMotion
- zigbeeGlassBreak
- zigbeeWater
- zigbeeIntertia
- zigbeeBigGEMotion
- zigbeeSurenRFMotion
- zigbeeCODetector
- zigbeeVSDoorWindow
- zigbeeVSCLIPMotion
- zigbeeVSK9_85Motion
- zigbeeVSFlood
- zigbeeVSSmoke
- zigbeeVSDW_1Wired
- zigbeeVSCO

- zigbeeVSMCT100Universal
- zigbeeVSDiscoveryPIRMot
- zigbeeVSTower40MCWMot
- zigbeeVSK940MCWMotion
- zigbeeVSDiscoveryK9_80Mot
- zigbeeVSDiscoveryQuad_80Mot
- zigbeeVSMCT441Gas
- zigbeeVSMCT501GlassBreak
- zigbeeVSMCT560Temp
- zigbeeVSNextpK9_85Motion
- zigbeeMTLDoorWindow
- zigbeeMTLSmoke
- zigbeeMTLGlassBreak
- zigbeeMTLGEMotion
- zigbeeMTLSurenMotion
- zigbeeMicroDoorWindow
- zigbeeGECODetector
- zigbeeMTLGECODetector
- zigbeeSMCMotion
- zigbeeSMCSmoke
- zigbeeSMCSmokeNoSiren
- zigbeeSMCGlassBreak
- zigbeeSMCCOSensor

The following items have been deprecated:

- boschPirMotion
- boschDUALMotion
- boschSmoke

- boschDoor
- boschInertia
- boschGlassBreak
- boschHeat
- boschRecessedDoor
- boschLrPir
- boschMiniDoor
- boschCoDetector
- boschWater
- boschGas

Appendix M: Peripheral Types

The instance/peripheral instance contains a Point of type 'peripheral/type'. The current supported types are listed below:

- siren
- keyfob
- keypad
- takeoverKeypad
- wifiRepeater
- router
- zigbeeVSMCT101_1Btn
- zigbeeVSMCT102_2Btn
- zigbeeVSMCT103_3Btn
- zigbeeVSMCT104_4Btn
- zigbeeVSMCT124_4Btn
- zigbeeVSMCT220EmerBtn
- zigbeeVSMCT241Pendant