redwoodsystems®

Smart Lights | Smart Buildings

Redwood API
User Guide

August 2012

Table of Contents

The Redwood Application Programming Interface (API) provides two-way communication with the Redwood Building-Performance Lighting Platform as well as other systems and applications. The API enables computers, smartphones, and tablets to access Redwood Systems sensor network data and lighting controls. The API is accessible from the Redwood Director appliance.

The Redwood API is available as a REST API with a limited set of functionality and a more robust version, called the Unified API. The REST API is the simplest way to retrieve the Redwood Systems data model and sensor network data. The Unified API provides enhanced functionality, including initiating lighting controls and batch processing. Both implementations use JSON framing for objects and support HTTPS.

1.1 Access Control Authentication

You must enter a username and password to access all API functionality. The API login credentials are the same as the Redwood Manager administrative login credentials.

A basic access authentication protocol communicates login credentials. You must use HTTPS to communicate login credentials and data securely.

1.1.1 Accessing the Redwood REST API

You signal the request type (read) via the HTTP GET method, and you specify the database schema via a URI. Additional semantics are passed via URI parameters. The response contains a JSON-framed object representing the database object referenced by the URI. If the URI is invalid, an HTTP 404 error is returned.

REST API URL

```
http[s]://<DirectorName>/rApi/<ResourceURI>
```

1.1.2 Accessing the Redwood Unified API

You access the API via the HTTP POST method with a full JSON-framed request object encapsulating the type of request (get or set) and the database schema components. A JSON-framed response object containing the request type results, possible error conditions, and possible elements within the schema is returned. Only synchronous requests are supported.

Unified API URL

```
http[s]://<DirectorName>/uApi
```

1.2 Structural Descriptions

The Data Model structural descriptions reference the following terms. The Data Model supports both primitive and terminal types.

Structural Description Terms	
<String>	Double-quoted ASCII string, for example, "Hello" and "A slightly longer example." If an element requires an exact ASCII string, it is included in the element's description.
<Number>	Integer or real value. Numbers are not quoted. Examples include 123, 4.5, and –6.78. Supported precisions, ranges, and dimensions are included in the element's description.
<Boolean>	Only supported values are true and false, without quotes.

1.3 URI Naming

Each element of the Data Model can be uniquely named as a relocatable URI, referred to as a <ResourceURI>.

URI Syntax
Starting with / (slash), which represents the top of the Data Model, reference individual elements by the name of the attribute.
For non-array attributes that represent composite types—non-primitive types—reference subattributes with additional slashes (/).
For array attributes that contain composite types, reference individual elements in the array by specifying the key-attribute value prefixed by a / (slash).
Attributes, both array and non-array, can also be reference types. A reference type attribute is logically a pointer type and viewed as a terminal. Although a reference type can point to a composite type, the reference type attribute cannot be further URI-extended. However, because the value of a reference type attribute is a relocatable URI, you can make additional API requests to the referred composite type using the returned URI.
Each component of the <ResourceURI> (the content between the slashes) must be encoded according to the URI-encoding rules described in Sections 2.3 and 2.4 of the URI RFC standard. (See REST API: Search Capabilities for a URI example)

URI Examples	
/	The entire Data Model. With the REST API, you can omit this when requesting the entire Data Model. For example, https://<DirectorName>/rApi and https://<DirectorName>/rApi/ are equivalent. But for all further attributes of the Data Model, a trailing / (slash) is not supported.
/name	Top-level name of the Data Model.
/location	Top-level location array. Requesting this via the REST API returns a JSON array of all location objects.
/location/10	Individual location object in the location array with the ID 10. Requesting this via the REST API returns a JSON object, which is the location object.
/location/10/ policyTrigger/ foobar/type	Trigger type attribute of a specific policy trigger named foobar, which itself is contained in a location object with the ID 10. Requesting this via the REST API returns a JSON string.

1.4 Data Model Structure

Data Model elements and attributes are outlined in the following tables. **Note:** All attributes are read-only unless specified as **Configurable**.

<DataModel> version 1.3.0	
{ "name": <String>, "currentTime": <Number>, "rootLocation": <LocationReference>, "location": [<Location>], "fixture": [<Fixture>] }	
name	Name of the system instance that this API services. Typically, it is the same as the configured system name. However, it might be different from the UI-configured name if a name change has not become active.
currentTime	Unsigned integer indicating the current time on the cluster, as seconds since the Unix epoch. Examples are 1, 234, and 56789.
rootLocation	String specifying the URI of the top-level location of the hierarchical topology. This location is not a child of any other location.

location	Array of all locations in the cluster. Each location in the array has a locally unique ID.
fixture	Array of all fixtures in the cluster. Each fixture in the array has a locally unique serial number.

<Location>

```
{
    "id": <Number>,
    "name": <String>,
    "childLocation": [<LocationReference>],
    "childFixture": [<FixtureReference>],
    "sensorStats": <LocationSensorStats>,
    "policyTrigger": [<PolicyTrigger>],
    "sceneControl": <SceneControl>
}
```

id	Key attribute for a location. In an array of locations, each location has a unique ID as an unsigned integer. Examples are 1, 234, and 56789.
name	Name of the location.
childLocation	Array of URIs of all locations that are children of this location within a hierarchical topology view. Each URI string must be locally unique—the same location cannot be listed multiple times.
childFixture	Array of all child fixture URIs. Each URI string must be locally unique.
sensorStats	Object that aggregates all location sensor statistics.
policyTrigger	Array of all EXTERNAL type policy triggers (unscheduled events). Each policy trigger must have a locally unique name.
sceneControl	Object that aggregates scene control data for this location.

<Fixture>

```
{
    "serialNum": <String>,
    "name": <String>,
    "type": <String>,
```

```
    "sensorStats": <FixtureSensorStats>
}
```

serialNum	Key attribute for a fixture. In an array of fixtures, each fixture must have a unique serial number.
name	Name of the fixture.
type	Fixture type. Available values are LUMINAIRE, HYBRID*, FIXTURELESS, and WALL_SWITCH*. (* type variations will have different endings)
sensorStats	Object that aggregates all fixture sensor statistics.

<PolicyTrigger>

```
{
    "name": <String>,
    "type": <String>,
    "activeValue": <Number>
}
```

name	Name of the trigger (key attribute for a location). In an array of policy triggers in a location, each policy trigger must have a unique name.
type	Policy trigger type. The only available value is EXTERNAL.
activeValue	**Configurable**. A value of 0 specifies a non-active trigger. A value of 1 specifies that the trigger is active.

<SceneControl>

```
{
    "activeSceneName": <String>,
    "scene": [<Scene>]
}
```

activeSceneName	**Configurable**. Specifies which scene is active. An empty string ("") specifies that no scene is active. Only strings that match the name of an existing scene in the scene list of this scene control array are supported. If this value is set to a supported value, the scene with the matching name value is active. If the matching scene is removed, this attribute resets to the empty string.

scene	Array of all scene objects. Each scene in the array must have a locally unique name.

<Scene>

```
{
    "name": <String>,
    "order": <Number>
}
```

name	Name of the scene—same name as configured and displayed via the UI. This is the key attribute for a scene. In an array of scenes, each scene must have a unique name.
order	Arbitrary value (nonzero, positive integer) as set by the UI that represents the display order of scenes in the UI. It can be used by other clients to maintain a consistent display order.

<LocationSensorStats >

```
{
    "power": <PowerStats>,
    "ceilingTemperature": <TemperatureStats>,
    "roomTemperature": <TemperatureStats>,
    "motion": <MotionStats>
}
```

power	Object that aggregates all power-related statistics for a location.
ceilingTemperature	Object that aggregates all ceiling temperature-related statistics for a location.
roomTemperature	Object that aggregates all room temperature-related statistics for a location.
motion	Object that aggregates all motion-related statistics for a location.

<FixtureSensorStats>

```
{
     "power": <PowerStats>,
     "temperature": <TemperatureStats>,
     "motion": <MotionStats>,
```

}	
power	Object that aggregates all power-related statistics for a fixture.
temperature	Object that aggregates all temperature-related statistics for a fixture.
motion	Object that aggregates all motion-related statistics for a fixture.

<PowerStats>	
{ "instant": <Number> }	
instant	Real-time power consumed by the object (location or fixture) in Watts. Unsigned real value with up to three digits of precision. Examples are 1, 2.34, and 5678.9.

<TempertureStats>	
{ "instant": <Number> }	
instant	Real-time temperature for the object (location or fixture) in Celsius. Signed real value with up to two digits of precision. Examples are 1, –2.34, and 5678.9.

<MotionStats>	
{ "instant": <Number> }	
instant	Time when the last motion was detected for the object (location or fixture) in seconds since the Unix epoch. Examples are 1, 234, and 56789.

1.5 REST API: JSONP Callback

To enable cross-domain applications, the Redwood REST API supports JSONP callback functionality with response framing that is enhanced to facilitate asynchronous clients. Additional data specified in the query string is ignored, but included in the JSONP response.

REST API URL request with JSONP callback

```
http[s]://<DirectorName>/rApi/<ResourceURI>[?jsonpCallback=<String>]
```

REST API response with JSONP callback

```
<jsonpCallbackStringValue>(
    {
        "request": <requestResourceUriString>,
        "response": <responseObject>
    }
)
```

request	Complete, relocatable URI including the query data. The protocol information, host domain name, and API entry point are not included. For example, for the request https://mydomain.com/rApi/location/10?jsonpCallback=myCallback, this attribute contains /location/10?jsonpCallback=myCallback.
response	Requested object. Identical to what is returned if a JSONP callback is not specified.

1.6 REST API: Search Capabilities

The REST API supports searching for locations, fixtures, policy triggers by attribute. The request must be an exact match and is case sensitive. You can include only one attribute in the search request. All matches are returned. If no matches are found, an HTTP 404 error is returned.

If array attributes contain composite types, instead of specifying an individual element via the key-attribute value, you can reference elements with a <non-key-attribute, value> search tuple at the end of the <ResourceURI>. The search tuple is specified as $<attribute-name>:<attribute-value>. In a search tuple, only attributes in the Data Model of type <String> and <Number> are supported.

If a key-attribute value is specified instead of the search tuple and the key attribute is of type <String>, and the first character of the string value is a $, it must be URI encoded (passed as %24).

REST API URI naming examples for search requests	
`/location/$name:Office`	All locations whose "name" attribute is set to the string Office.
`/location/$id:102`	Location whose "id" attribute is 102.
`/location/$name:Office/sensorStats`	sensorStats objects for each location whose "name" attribute is set to the string Office.
`/location/0/policyTrigger/%24odd%20trigger%2Fname/type`	Type attribute of a specific <PolicyTrigger> with the name "$odd trigger/name". In the URI, the leading $, the space, and / are URI-encoded.

For a search query (defined as containing a search-tuple specification in the <ResourceURI>), the search response object is a JSON-framed array as follows:

REST API response for search request

```
{
    "request": <requestResourceUriString>,
    "response": <responseObject>
}
```

`request`	Complete, relocatable URI that specifies the URI name of a specific matching element in the <DataModel> instance that matched against the search query. This does not include any query data that may have been included as part of the search query.
`response`	The specific matching element in the <DataModel> instance that matched against the search query. The URI specified in the "request" attribute is the URI name for this element..

1.7 Unified API Request and Response Framing

For the Redwood Unified API, all requests and responses are structured as JSON framed objects. The content is encapsulated in a top-level bracketed pair, which represents the request or response.

Unified API request structure

```
{
    "protocolVersion": <String>,
    "schemaVersion": <String>,
    "requestType": <String>,
    "requestData": <DataModel>
}
```

protocolVersion	Version of the JSON framing structure for this request. Must be in the form x.y.z, with each x.y.z component an integer. The y and z components are optional and do not need to be included if both are 0. The only valid values are 1.0.0, 1.0, and 1.
schemaVersion	Schema version used for this request. The only valid values are 1.3.0 and 1.3. The format is identical to protocolVersion.
requestType	Get the current values for the data requested. See requestData below.
	Set the current values for the data requested.
requestData	JSON object that represents the portion of the Data Model that the request is acting on. It is encapsulated in a bracketed pair. For a get request, the attributes requested are specified as either attribute:null, attribute:[], or attribute:{}. The rValues in the JSON attribute:value pair are special markers that represent the attribute of the value requested: • The null marker is used for primitive attribute types. • The [] marker is used for array attribute types. • The {} marker is used for object attribute types. The Data Model might terminate with non-primitive elements, indicating that the full content of the non-primitive element is being requested. The Data Model includes hierarchical composed structures. By default, the fully composed substructure is output when a get request's Data Model contains non-primitive terminals.

Unified API response structure

```
{
    "protocolVersion": <String>,
    "schemaVersion": <String>,
    "responseType": <String>,
    "responseErrorType": <String>,
    "responseErrorDetail": <String>,
    "responseData": <DataModel>
}
```

`protocolVersion`	Version of the JSON framing that the API service is actively using in the form x.y.z, with each x.y.z component an integer. The y and z components are omitted if both are 0. Commonly, it is identical to the protocolVersion specified in the request. Compatibility across protocol versions might or might not be supported.
`schemaVersion`	Version of the schema that the API service is actively using. The format is identical to protocolVersion. Compatibility across protocol versions is not supported.
`responseType`	**getResponse**: Indicates that this is a valid response to a get request.
	setResponse: Indicates that this is a valid response to a set request.
	errorResponse: Indicates that the request was not properly serviced. See responseErrorType for more information.
`responseErrorType` **Note**: Authentication failures are handled at the HTTP[S] level.	**unsupportedService**: The API service is not available on the system. This typically occurs because a request is being made to a Redwood Engine, and the API service is available only on the Redwood Director. Contact your customer service representative for assistance.
	badRequestFraming: The request is poorly framed JSON. Further details are provided via responseErrorDetail.
	unspecifiedProtocolVersion: The protocol version is not specified in the request.
	unsupportedProtocolVersion: The API service does not support the protocol version specified.
	unspecifiedSchemaVersion: The schema version is not specified in the request.
	unsupportedSchemaVersion: The API service does not support the schema version specified.
	unspecifiedRequestType: The request type is not specified in the request.
	unsupportedRequestType: The request type is invalid.
	unspecifiedRequestData: The request data is not specified in the request.
	unsupportedRequestData: The request data does not map into the schema. Further details are provided via responseErrorDetail. This typically occurs because an element within the request data does not exist in the Data Model that the API services.

	accessFailure: The request is for something that the given authentication level does not allow. Commonly, this occurs because a set request is specified on a portion of the Data Model that does not allow set requests. The majority of the Data Model is available for read-only (get) access. Only enabling and disabling triggers support read/write access.
	rangeFailure: The request specifies an invalid rValue for an attribute of type <Number>. This commonly occurs because the rValue specified on a set request is out of range for the attribute being modified.
	internalFailure: The request is unserviceable for unknown reasons. Further details are provided via responseErrorDetail. Contact your customer support representative if this error is reported.
`responseErrorDetail`	Provides further details on the nature of an error for certain classes of responseErrorType.
`responseData`	JSON object that represents the portion of the Data Model that fulfills the get request when the responseType is getResponse. It is encapsulated in a bracketed pair.

1.8 Examples

The following examples show how requests are structured and the resulting responses. The examples are based on the following <DataModelExample> instance, which is composed of two offices with one fixture each, and a conference room with two fixtures.

<DataModelExample>

```
{
    "name": "Example System",
    "currentTime": 12345,
    "rootLocation": "/location/0",
    "location": [
        {
            "id": 0,
            "name": "All Locations",
            "childLocation": [
                "/location/11",
                "/location/100",
                "/location/101",
                "/location/102"
            ],
            "policyTrigger": [
                {
                    "name": "Example Trigger 1",
                    "type": "EXTERNAL",
```

```
                "activeValue": 0
            },
            {
                "name": "Example Trigger 2",
                "type": "EXTERNAL",
                "activeValue": 0
            }
        ],
        "sensorStats": {
            "power": {
                "instant": 15
            },
            "ceilingTemperature": {
                "instant": 26
            },
            "motion": {
                 "instant": 1340729008
            }
        },
        "childFixture": []
    },
    {
        "id": 11,
        "name": "Unassigned",
        "childFixture": [
            "/fixture/sn4",
            "/fixture/sn5"
        ],
        "sensorStats": {
            "power": {
                "instant": 10
            },
            "ceilingTemperature": {
                "instant": 20
            },
            "motion": {
                "instant": 1340729008
            }
        }
    },
    {
        "id": 100,
        "name": "Office",
        "childFixture": [
            "/fixture/sn1"
        ],
        "sceneControl": {
            "activeSceneName": "",
            "scene": [
                {
                    "name": "Scene 100",
                    "order": 1
                },
                {
```

```
                    "name": "Scene 50",
                    "order": 3
                },
                {
                    "name": "Scene 75",
                    "order": 2
                }
            ]
        },
        "sensorStats": {
            "power": {
                "instant": 4.5
            },
            "ceilingTemperature": {
                "instant": 25.5
            },
            "motion": {
                "instant": 1340729008
            }
        }
    },
    {
        "id": 101,
        "name": "Conference Room",
        "childFixture": [
            "/fixture/sn2",
            "/fixture/sn3"
        ],
        "sensorStats": {
            "power": {
                "instant": 10.5
            },
            "ceilingTemperature": {
                "instant": 26.25
            },
            "motion": {
                "instant": 1340729008
            }
        }
    },
    {
        "id": 102,
        "name": "Office",
        "childFixture": [
            "/fixture/sn4"
        ],
        "sensorStats": {
            "power": {
                "instant": 9.5
            },
            "ceilingTemperature": {
                "instant": 25.08
            },
            "motion": {
```

```
                    "instant": 1340729008
                }
            }
        }
    ],
    "fixture": [
        {
            "serialNum": "sn1",
            "name": "Office Fixture #1",
            "type": "LUMINAIRE",
            "sensorStats": {
                "power": {
                    "instant": 4.5
                },
                "temperature": {
                    "instant": 25.5
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        },
        {
            "serialNum": "sn2",
            "name": "Conference Room Fixture #1",
            "type": "LUMINAIRE",
            "sensorStats": {
                "power": {
                    "instant": 5.5
                },
                "temperature": {
                    "instant": 26.5
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        },
        {
            "serialNum": "sn3",
            "name": "Conference Room Fixture #2",
            "type": "LUMINAIRE",
            "sensorStats": {
                "power": {
                    "instant": 5
                },
                "temperature": {
                    "instant": 26
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        },
```

```
        {
            "serialNum": "sn4",
            "name": "Office Fixture #1",
            "type": "HYBRID",
            "sensorStats": {
                "power": {
                    "instant": 10
                },
                "ceilingTemperature": {
                    "instant": 20
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        },
        {
            "serialNum": "sn5",
            "name": "Unassigned #1",
            "type": "FIXTURELESS ",
            "sensorStats": {
                "power": {
                    "instant": 10
                },
                "ceilingTemperature": {
                    "instant": 20
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        },
        {
            "serialNum": "sn6",
            "name": "Unassigned #2",
            "type": "FIXTURELESS ",
            "sensorStats": {
                "power": {
                    "instant": 9.5
                },
                "ceilingTemperature": {
                    "instant": 24
                },
                "motion": {
                    "instant": 1340729008
                }
            }
        }
    ]
}
```


1.8.1 Example: Get Entire Data Model

This example retrieves the entire Data Model.

With the REST API, you make an HTTP GET request to the following URL below. The response is the <DataModelExample> represented above.

REST API request to get entire Data Model

```
http[s]://<DirectorName>/rApi
```

With the Unified API, when you request the Data Model, you receive the response below. The rValue for the requestData is {}, which specifies that you are requesting a get of that object in its entirety. Specifying the rValue for any attribute as either null, [], or {} indicates that is the attribute being requested. If the attribute is an object ({}) or an array ([]), the full object or set of objects is returned. responseData is a special case, but these semantics apply to all attributes of the <DataModelExample>.

API request to get entire Data Model

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "get",
    "requestData": {}
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "getResponse",
    "responseData": <DataModelExample>
}
```

1.8.2 Example: Get All Data for a Specific Location

This example retrieves a specific location by its ID number (Office, which has ID 100) and all the data within that location.

Note: The REST API also supports retrieving a location by name. See the example in 1.7.9.

REST API request for entire sub-tree for a specific location

```
http[s]://<DirectorName>/rApi/location/100
```

REST API response


```
{
    "id": 100,
    "name": "Office",
    "childFixture": [
        "/fixture/sn1"
    ],
    "sceneControl": {
        "activeSceneName": "",
        "scene": [
            {
                "name": "Scene 100",
                "order": 1
            },
            {
                "name": "Scene 50",
                "order": 3
            },
            {
                "name": "Scene 75",
                "order": 2
            }
        ]
    },
    "sensorStats": {
        "power": {
            "instant": 4.5
        },
        "ceilingTemperature": {
            "instant": 25.5
        },
        "motion": {
            "instant": 1340729008
        }
    }
}
```


Using the Unified API, when you specify an rValue of the key attribute ("id") of an object in an array attribute ("location") and no other attributes, the full object that matches the specified key-attribute value (1) is returned. If no matching object is found, an empty responseData object is returned ("responseData" : {}).

Unified API request for entire sub-tree for a specific location

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "get",
    "requestData": {
        "location": [
            {
                "id": 100
            }
        ]
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "getResponse",
    "responseData": {
        "location": [
            {
                "id": 100,
                "name": "Office",
                "childFixture": [
                    "/fixture/sn1"
                ],
                "sceneControl": {
                    "activeSceneName": "",
                    "scene": [
                        {
                            "name": "Scene 100",
                            "order": 1
                        },
                        {
                            "name": "Scene 50",
                            "order": 3
                        },
                        {
                            "name": "Scene 75",
                            "order": 2
                        }
                    ]
                },
                "sensorStats": {
                    "power": {
                        "instant": 4.5
                    },
                    "ceilingTemperature": {
```

```
                        "instant": 25.5
                    },
                    "motion": {
                        "instant": 1340729008
                    }
                }
            }
        ]
    }
}
```

1.8.3 Example: Get IDs Only for Locations

This example retrieves only the IDs of all locations. The main difference between this example and the previous example of requesting a sub-tree for a specific location is whether the rValue for the key attribute ("name") is specified or null. When the rValue is null, it returns only the key-attribute value of all the objects in the array.
Note: The REST API does not support this request.

Unified API request to get IDs only for locations

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "get",
    "requestData": {
        "location": [
            {
                "id": null
            }
        ]
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "getResponse",
    "responseData": {
        "location": [
            {
                "id": 0
            },
            {
```

```
                "id": 11
            },
            {
                "id": 100
            },
            {
                "id": 101
            },
            {
                "id": 102
            }
        ]
    }
}
```

1.8.4 Example: Get a Subset of Data for a Specific Location

This example shows how to retrieve a specific subset of data for a location. The request gets the power and ceiling temperature only for the Conference Room, which has an ID of 101.

Because the REST API does not support limiting data within an object to a subset of attributes, you must make an HTTP GET request for the entire contents to the URL.

REST API request for data for a specific location

```
http[s]://<DirectorName>/rApi/location/101/sensorStats
```

REST API response

```
{
    "power": {
        "instant": 10.5
    },
    "ceilingTemperature": {
        "instant": 26.25
    },
    "motion": {
        "instant": 1340729008
    }
}
```

You can request a subset of data with the Unified API.

Unified API request for a subset of data for a specific location

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "get",
    "requestData": {
        "location": [
            {
                "id": 101,
                "sensorStats": {
                    "power": {
                        "instant": null
                    },
                    "ceilingTemperature": {
                        "instant": null
                    }
                }
            }
        ]
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "getResponse",
    "responseData": {
        "location": [
            {
                "id": 101,
                "sensorStats": {
                    "power": {
                        "instant": 10.5
                    },
                    "ceilingTemperature": {
                        "instant": 26.25
                    }
                }
            }
        ]
    }
}
```

1.8.5 Example: Get a Subset of Data for All Fixtures

This example retrieves only the power and temperature for all fixtures.

To request all elements in an array, the key attribute is unspecified. When elements of an array are retrieved, the key-attribute value is always included. The key attribute identifies each object within an array. Only the data for the elements whose rValue is null are retrieved.

Similar to the previous example, the REST API does not support this request.

Unified API request to get a subset of data for all fixtures

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "get",
    "requestData": {
        "fixture": [
            {
                "sensorStats": {
                    "power": {
                        "instant": null
                    },
                    "temperature": {
                        "instant": null
                    }
                }
            }
        ]
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "getResponse",
    "responseData": {
        "fixture": [
            {
                "serialNum": "sn1",
                "sensorStats": {
                    "power": {
                        "instant": 4.5
                    },
                    "temperature": {
                        "instant": 25.5
                    }
                }
            },
            {
                "serialNum": "sn2",
                "sensorStats": {
```

```
                    "power": {
                        "instant": 5.5
                    },
                    "temperature": {
                        "instant": 26.5
                    }
                }
            },
            {
                "serialNum": "sn3",
                "sensorStats": {
                    "power": {
                        "instant": 5
                    },
                    "temperature": {
                        "instant": 26
                    }
                }
            },
            {
                "serialNum": "sn4",
                "sensorStats": {
                    "power": {
                        "instant": 10
                    },
                    "ceilingTemperature": {
                        "instant": 20
                    }
                }
            },
            {
                "serialNum": "sn5",
                "sensorStats": {
                    "power": {
                        "instant": 10
                    },
                    "ceilingTemperature": {
                        "instant": 20
                    }
                }
            },
            {
                "serialNum": "sn6",
                "sensorStats": {
                    "power": {
                        "instant": 9.5
                    },
                    "ceilingTemperature": {
                        "instant": 24
                    }
                }
            }
        ]
    }
```

```
}
```

1.8.6 Example: Specify JSONP Callback on a Get Request

This example shows how to specify a JSONP callback. This functionality is supported only with the REST API.

The request is identical to the example of getting a subset of data for a specific location, except that it requests a callback named myCallback.

REST API request using a JSONP callback

```
http[s]://<DirectorName>/rApi/location/101/sensorStats?jsonpCallback=myCallback
```

REST API response

```
myCallback(
    {
        "request": "/location/101/sensorStats?jsonpCallback=myCallback",
        "response": {
            "power": {
                "instant": 10.5
            },
            "ceilingTemperature": {
                "instant": 26.25
            },
            "motion": {
                "instant": 1340729008
            }
        }
    }
)
```

1.8.7 Example: Set the activeValue for an External Policy Trigger

This example sets a policy trigger's activeValue. On a set, the response does not include a responseData if the set completes successfully.

Note: The REST API does not support this request.

Unified API request to set the activeValue for an external policy trigger

```
{
```

```
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "set",
    "requestData": {
        "location": [
            {
                "id": 0,
                "policyTrigger": [
                    {
                        "name": "Example Trigger 1",
                        "activeValue": 1
                    }
                ]
            }
        ]
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "setResponse"
}
```

1.8.8 Example: Set the activeSceneName for a Location Scene

This example sets the active scene name of a location scene control object. On a set, the response does not include a responseData if the set completes successfully.

Note: The REST API does not support this request.

Unified API request to set the activeValue for a location scene

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "requestType": "set",
    "requestData": {
        "location": [
            {
                "id": 100,
                "sceneControl": {
                    "activeSceneName": "Scene 75"
                }
            }
        ]
```

```
    }
}
```

Unified API response

```
{
    "protocolVersion": "1",
    "schemaVersion": "1.3.0",
    "responseType": "setResponse"
}
```

1.8.9 Example: Search Locations by Name

This example searches for locations with the name Conference Room and requests all data for any matching locations. In this example, only one matching location is found.

Note: The Unified API does not support this request.

REST API search for location by name

```
http[s]://<DirectorName>/rApi/location/$name:Conference%20Room
```

REST API response

```
[
    {
        "request": "/location/101",
        "response": {
            "id": 101,
            "name": "Conference Room",
            "childFixture": [
                "/fixture/sn2",
                "/fixture/sn3"
            ],
            "sensorStats": {
                "power": {
                    "instant": 10.5
                },
                "ceilingTemperature": {
                    "instant": 26.25
                },
                "motion": {
                    "instant": 1340729008
                }
            }
```

```
        }
    }
]
```

1.8.10 Example: Search Locations by Name for Specific Data

This example searches for locations with the name Office and requests only the instant motion data within the matching locations. Two search response objects are included in the response JSON array because two locations in the Data Model have the name Office. Only the data requested is returned.

Note: The Unified API does not support this request.

REST API search for specific data for locations named Office

```
http[s]://<DirectorName>/rApi/location/$name:Office/sensorStats/motion/instant
```

REST API response

```
[
    {
        "request": "/location/100",
        "response": 1340729008
    },
    {
        "request": "/location/102",
        "response": 1340729008
    }
]
```

Redwood Systems | 3839 Spinnaker Court | Fremont, CA 94538-6537 | **T** 510 270 5360 | **F** 510 270 5364 | **www.redwoodsystems.com**

091112