



Power IQ

WS-API Programming Guide Release 1.5.0 and 1.5.1

Copyright © 2009 Raritan, Inc.

PIQAPI-0B-v1.5.0-1.5.1-E

February 2010

255-80-6102-00

This document contains proprietary information that is protected by copyright. All rights reserved. No part of this document may be photocopied, reproduced, or translated into another language without express prior written consent of Raritan, Inc.

© Copyright 2010 Raritan, Inc., CommandCenter®, Dominion®, Paragon® and the Raritan company logo are trademarks or registered trademarks of Raritan, Inc. All rights reserved. Java® is a registered trademark of Sun Microsystems, Inc. Internet Explorer® is a registered trademark of Microsoft Corporation. Netscape® and Netscape Navigator® are registered trademarks of Netscape Communication Corporation. All other trademarks or registered trademarks are the property of their respective holders.

FCC Information

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a commercial installation. This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential environment may cause harmful interference.

VCCI Information (Japan)

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づくクラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

Raritan is not responsible for damage to this product resulting from accident, disaster, misuse, abuse, non-Raritan modification of the product, or other events outside of Raritan's reasonable control or not arising under normal operating conditions.



Contents

Chapter 1 Introduction	5
<hr/>	
Chapter 2 Use Cases	6
Use case 1: Asset management system/data synchronization	6
Use case 2: Power control	6
<hr/>	
Chapter 3 Technical Description Summary	7
<hr/>	
Chapter 4 Functional Description	8
Fields in Power IQ.....	8
PDUs	8
Device.....	9
Device Group.....	9
Rack.....	10
Row.....	10
Aisle	10
Room	11
Floor.....	11
Data Center	11
List of operations in the API.....	12
Requests and commands.....	12
<hr/>	
Chapter 5 Technical description	14
Getting started	14
Enable the WS API	14
Port Requirement.....	14
WSDL Files	14
Security	15
Authorization	16
Synchronous and Asynchronous Method Calls	16
Synchronous Example.....	17
Asynchronous Example, Successful Job	17
Asynchronous Example, Job with a Problem	19

Contents

Internationalization.....	19
Chapter 6 Power Control Examples	20
Step 1: Current state of device	20
Step 2: Switch outlet off	22
Step 3: Check status of job.....	23
Chapter 7 FindEDMentities Examples	25
Returning Multiple Entities Using the IN Predicate	26
Using the LIKE Predicate.....	27
Finding Exact Matches Using the EQUAL Predicate.....	28
Creating Queries With Multiple Conditions.....	29
Appendix A References	31
Index	33

Chapter 1 Introduction

This document describes the Power IQ web services API. This API is bi-directional: external applications can access and update the information in Power IQ, and request power control actions.

In short, the document describes:

- Example use cases of what can be done with this API
- A technical description of the API
- A functional description of the API including a list of API calls, and an overview of the relevant objects and their fields

The word "PDU" in this document means "rack PDU". The concept "EDM entity" includes PDUs, Devices, Racks, Data Centers etc. EDM is an abbreviation of Enterprise Data Model.

Power IQ can be used:

- as a stand-alone application (without integration with an external application)
- integrated with an external application using the API described in this document
- a combination of both, because changes made in Power IQ will show up in the external asset management application and vice versa

Chapter 2 Use Cases

Here are some example use cases of the Power IQ web services API.

- An asset management system is used to manage/maintain inventory information and Power IQ is kept in sync (data synchronization)
- Developers want the ability to remotely switch multiple outlets on or off (power control)

In This Chapter

Use case 1: Asset management system/data synchronization	6
Use case 2: Power control	6

Use case 1: Asset management system/data synchronization

A customer has an asset management system that contains information about floors and racks, such as: 'this floor has these racks', 'this rack has this name and this location', and 'these Servers are located in this rack'. The customer would like Power IQ's understanding of the world to be based on the information in the asset management system. If a Rack gets added to this system, or a PDU removed, Power IQ should be updated automatically.

Using the API, the customer can write a component to be added to the asset management system that leverages the API of Power IQ to inform Power IQ of certain events ("a new Rack was created, here are the details" or "PDU 192.168.0.1 was just deleted").

Use case 2: Power control

Each developer and tester is responsible for a rack. They go home, and then realize they forgot to switch off their rack. They login to an in-house custom application that utilizes the Power IQ web services API to switch off the power for the rack or one more devices attached to the rack.

Note: Users could also log in to the Power IQ web application remotely to control power.

Chapter 3 Technical Description Summary

This is a web services API that uses SOAP over HTTPS

Creating/updating/deleting Devices, PDU, Racks etc can be done through this interface (initiated by the Client application). See **Functional Description** (on page 8).

The API also supports Power Control. See **Functional Description** (on page 8).

The API uses WSDL and XML schema files to define the objects in the web services API. The WSDL will define data types, required fields, lengths of fields etc.

One cannot do firmware upgrade or bulk configuration of PDUs through the API

Administrative settings of Power IQ (such as its IP address, security settings etc) cannot be changed through the API. This includes user groups/role management.

Authentication is done using WS-Security. See **Getting Started** (on page 14).

Roles and permissions will be respected. This way, it is possible to define users on Power IQ that will be allowed to use the API, and fine grained permissions control will be available (the same permissions are available in the user interface).

Roles and permissions defined in Power IQ are also applicable to the Power IQ web services API. If a user is given permission to only control the power of one rack, that user will only be able to control the power of that one rack, whether through the Power IQ web interface, or using the Power IQ web services API. If the third-party application integrating with Power IQ needs less restrictive permissions, you can create a user in Power IQ with a broader set of roles. All Power IQ web services could then be executed with this user.

Chapter 4 Functional Description

In This Chapter

Fields in Power IQ8
List of operations in the API.....12

Fields in Power IQ

There are 2 types of fields in Power IQ, static and dynamic. The static fields remain within Power IQ (this includes fields like the name of a rack, the customer name of a device, etc). This is the majority of fields. The second type of field is dynamic, in the sense that those fields are retrieved from the PDU and/or used to update the fields on the PDU itself.

The sections below give an overview of the available fields for each object. Not all fields are required, and not all objects are required for Power IQ to function optimally. For example, information about "Rooms", "Floors" etc are optional and can be skipped.

PDU's

Field	Access type	Required?
ID	Read-only (key)	(generated)
IP address	Read-only (cannot be changed after initial creation)	Yes
Proxy identifier	Read-only (cannot be changed after initial creation)	No
Manufacturer	Read-only (provided by PDU)	N/a (provided by PDU)
Model	Read-only (provided by PDU)	N/a (provided by PDU)
Serial number	Read-only (provided by PDU)	N/a (provided by PDU)
Firmware version	Read-only (provided by PDU)	N/a (provided by PDU)
Rated voltage	Read-only (provided by PDU)	N/a (provided by PDU)
Rated current	Read-only (provided by PDU)	N/a (provided by PDU)
Rated VA	Read-only (provided by PDU)	N/a (provided by PDU)
Contact person name	Read only (provided by PDU)	No
PDU name	Read only (provided by PDU)	No
Location	Read only (provided by PDU)	No
SNMP version	Currently not available.	Yes

Field	Access type	Required?
SNMP community string/user name/other credentials	Currently not available.	Yes
Status (OK, Error, Bad Password etc)	Currently not available.	N/a (provided by PDU)
Single phase or Three phase	Read-only (provided by PDU)	N/a (provided by PDU)
Belongs to (which data center, floor, room, aisle, row or rack this PDU belongs to)	Read/write	No
List of outlets with the name and power state of each outlet	Read-only (power control is available. It's a separate API call though).	N/a (provided by PDU)

Device

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
Customer	Read/write	No
Device Type	Read/write	No
Power rating (Watts/VA)	Read/write	No
Decommissioned?	Read/write	No
Device groups	Read/write	No
Outlets connected to (the outlets where this device is drawing its power from)	Read/write	No
Custom field 1	Read/write	No
Custom field 2	Read/write	No
External key	Read/write	No
Rack this device belongs to	Read/write	Yes

Device Group

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes

Field	Access type	Required?
Members of the device group	Read/write	No

Rack

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
Location (GetSpaceId)	Read/write	No
External key	Read/write	No
Row, Aisle, Room, Floor or Data Center this Rack belongs to	Read/write	Yes

Row

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
External key	Read/write	No
Aisle, Room, Floor or Data Center this Row belongs to	Read/write	Yes

Aisle

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
External key	Read/write	No
Room, Floor or Data Center this Aisle belongs to	Read/write	Yes

Room

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
External key	Read/write	No
Floor or Data Center this Room belongs to	Read/write	Yes

Floor

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
External key	Read/write	No
Data Center this Floor belongs to	Read/write	Yes

Data Center

Field	Access type	Required?
ID	Read-only (key)	(generated)
Name	Read/write	Yes
Company name	Read/write	No
Contact name	Read/write	No
Phone	Read/write	No
Email	Read/write	No
City	Read/write	No
State/province	Read/write	No
Country	Read/write	No
Peak rate (\$/kWh)	Read/write	No
Off-peak rate (\$/kWh)	Read/write	No
Peak begin	Read/write	No
Peak end	Read/write	No

Field	Access type	Required?
CO2 factor	Read/write	No
Cooling factor	Read/write	No
Custom field 1	Read/write	No
Custom field 2	Read/write	No
External key	Read/write	No

List of operations in the API

Requests and commands

This is a list with API calls that will be available through the Web services API.

The table documents the following:

- **WSDL Operation:** The name of the WSDL operation. This is the "name" attribute of the tag <wsdl:operation> in Power IQ's WSDL. It uniquely identifies the operation.
- **Description:** Description of what this operation provides.
- **Synchronous:** Whether or not this is a synchronous call.

WSDL Operation	Description	Synchronous
createEdmEntity	Create an EDM entity	Yes
updateEdmEntity	Update an EDM entity	Yes
deleteEdmEntity	Delete an EDM entity	Yes
getEdmEntity	Get (view) an EDM entity	Yes
findEdmEntities	Advanced searching for EDM entities	Yes
(not supported)	Create a Device Group	Yes
(not supported)	Update a Device Group	Yes
(not supported)	Deleting a Device Group	Yes
(not supported)	Get (view) a Device Group	Yes
(not supported)	List Device Groups	Yes
outletPowerControl	Power Control for one or multiple outlets. Switch outlets on/off.	No

WSDL Operation	Description	Synchronous
deviceGroupPowerControl	Power Control for one Device Group. Switch a group of devices on/off (note: a device group needs to be created before this call can be used)	No
edmEntityPowerControl	Power control for an edm entity rack, row, device, etc. This does not include a PDU. To turn off outlets directly on a given PDU use outletPowerControl.	No

Chapter 5 Technical description

In This Chapter

Getting started	14
Security	15
Authorization	16
Synchronous and Asynchronous Method Calls	16
Internationalization	19

Getting started

Enable the WS API

To get started with the WS API, the API needs to be enabled through the Settings Tab. It is disabled by default.

► **To enable the Web API:**

1. In the Settings tab, click Web API in the Security and Encryption section.
2. Select the Enable Web API checkbox, then click Save.

Port Requirement

The API is available over HTTPS only (port 443).

WSDL Files

There are 3 separate WSDL files. They can be found on your Power IQ instance. The files are not accessible unless WS API is enabled. You may have to "View Page Source" after the page has loaded.

Assuming that poweriq.example.com is your Power IQ's hostname:

<https://poweriq.example.com/api/PowerControlService/?WSDL> – operations related to Power Control

<https://poweriq.example.com/api/EdmService/?WSDL> – operations related to creating, searching, viewing, updating, and removing EDM entities (PDUs, racks, datacenters)

<https://poweriq.example.com/api/JobService/?WSDL> – operations necessary for asynchronous operations (jobs)

Security

All users that can authenticate in the web UI will also be able to authenticate themselves through web services. The web services API uses the WS-Security UserNameToken profile.

Each web service request requires the UserNameToken. During each request the UserNameToken is used to authenticate the user. There is no need to call an explicit 'login' method for example.

► **Example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://schemas.xmlsoap.org/ws/2002/07/secext"
xmlns:q1="http://www.raritan.com/poweriq/integration/types"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <q0:Security>
      <UsernameToken>
        <Username>admin</Username>
        <Password>raritan</Password>
      </UsernameToken>
    </q0:Security>
  </soapenv:Header>
  <soapenv:Body>
    <q1:outletPowerControl>
      <q1:command>
        <q1:outlet>
          <q1:outletId>12</q1:outletId>
          <q1:pduId>12</q1:pduId>
        </q1:outlet>
        <q1:state>On</q1:state>
      </q1:command>
    </q1:outletPowerControl>
  </soapenv:Body>
</soapenv:Envelope>
```

Authorization

Permissions to act on objects and perform actions will be based on the current scheme used in Power IQ. Administrators will setup users, groups, and roles within the administration page. This setup will work the same in the UI as it does in a similar web service call.

▶ **Example:**

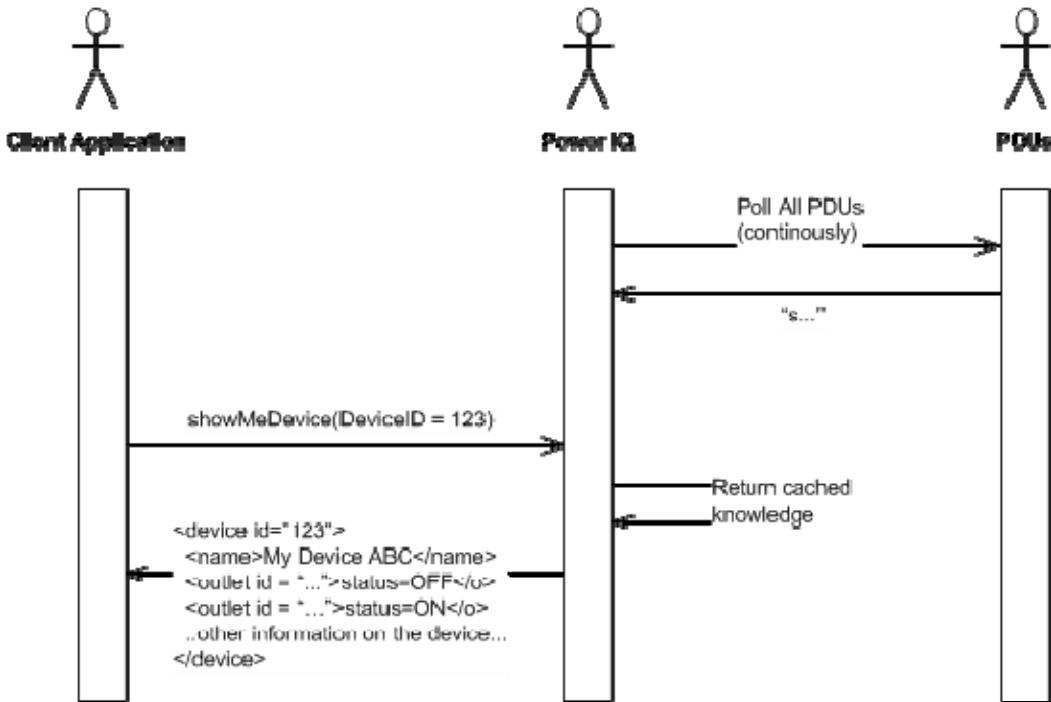
- User "Sammy" is given the Power Control role on rack1 by a Power IQ administrator through the web user interface.
- When Sammy invokes a web service to control the power in rack1, Power IQ will use his user, group, and role information to determine if he is allowed to control the power in rack1. In this case, Sammy will be allowed to control the power of rack1.

Synchronous and Asynchronous Method Calls

Asynchronous calls are limited to power control operations. As a general rule, anything that involves Power IQ communicating with PDUs is set up asynchronously. The WS API will return a Job ID that can be used in future calls to get the status of the request.

In those cases, the calling application needs to verify after the fact what happened. A job will either be ACTIVE or COMPLETE. If a job is COMPLETE, a flag 'hasErrors' will indicate if there were any errors while executing this job. The details of the errors are available through the GetJobMessages method.

Synchronous Example



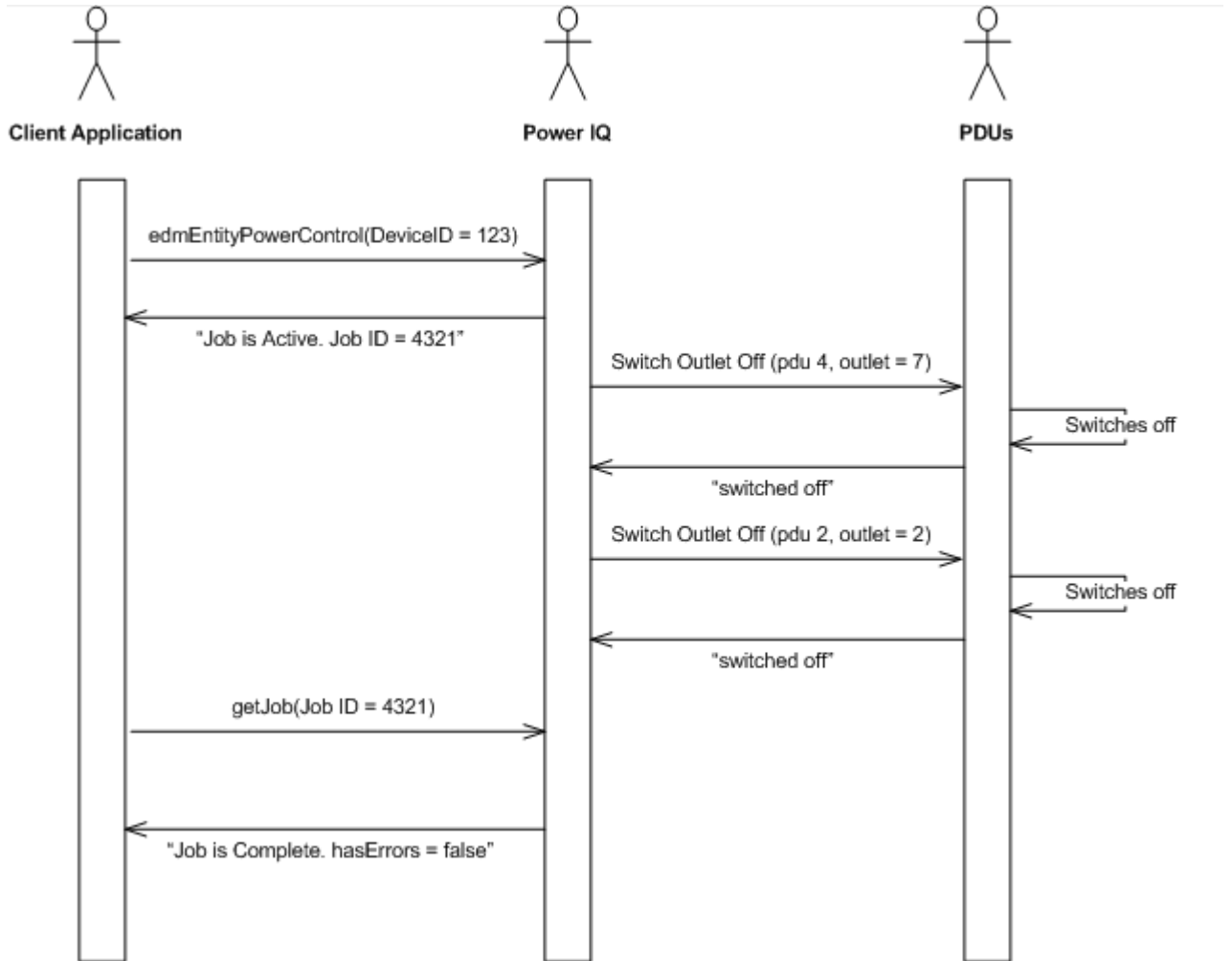
Asynchronous Example, Successful Job

Request: Power control: Switch Device 123 off

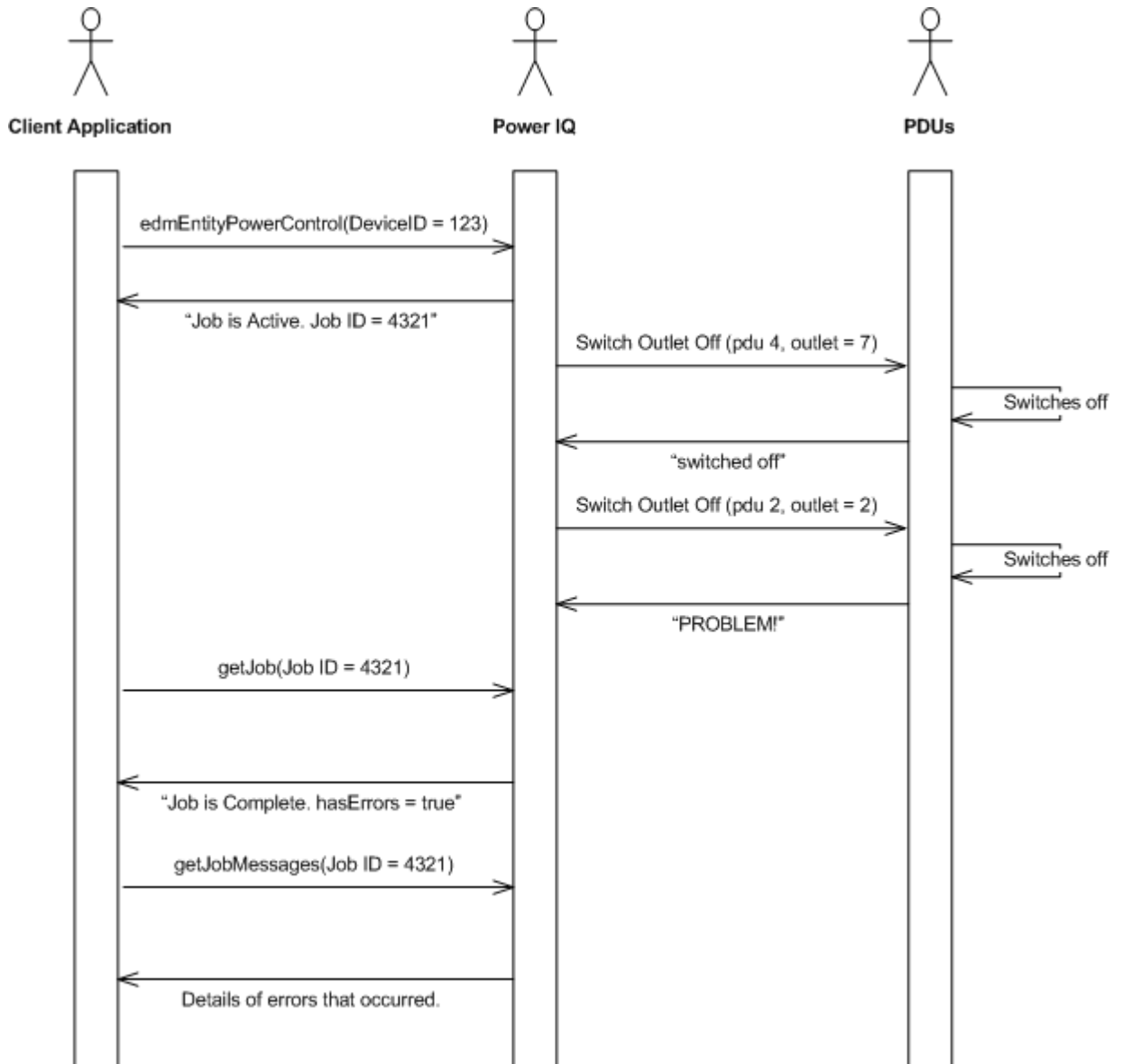
Reponse: Your Job ID is 4321

Request: What's the status of job 4321?

Response: Job 4321 has been completed.



Asynchronous Example, Job with a Problem



Internationalization

The API is not internationalized. Texts returned will be in English. Time stamps will be in the United States Eastern Standard timezone.

Chapter 6 Power Control Examples

If you are writing an application that allows a user to do power control operations through Power IQ's API, here is an example of how this can be implemented. This shows both synchronous and asynchronous commands.

► **Use case:**

An example application needs to display the power status of a device and give the user the option to turn the device on or off. The example application stores the external key and uses it to identify the device.

► **Steps:**

1. Ask Power IQ what the current power state of the device is (on or off) in order to display this to the user
2. Switch the device off using Power IQ
3. Verify if the "switch off" command was successful.

In This Chapter

Step 1: Current state of device	20
Step 2: Switch outlet off	22
Step 3: Check status of job	23

Step 1: Current state of device

The example application knows that the external key is "device-0044".

The first call gets the current power state of the device using the findEdmEntities method.

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:findEdmEntities xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:type>Device</typens:type>
  <typens:query>
    <typens:ordering>
      <typens:order direction="ascending" field="name"/>
    </typens:ordering>
    <typens:conditions>
      <typens:condition field="externalKey" value="device-0044"/>
    </typens:conditions>
    <typens:paging limit="25" offset="0"/>
  </typens:query>
</typens:findEdmEntities>
```

The response would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:findEdmEntitiesResponse
xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:result>
    <typens:message>Found one edm entity</typens:message>
  </typens:result>
  <typens:entities>
    <typens:device>
      <typens:id>1</typens:id>
      <typens:name>My Device Name</typens:name>
      <typens:externalKey>device-0044</typens:externalKey>
      <typens:customField1>customField1</typens:customField1>
      <typens:customField2>customField2</typens:customField2>
      <typens:customer>Some Customer</typens:customer>
      <typens:decommissioned>>false</typens:decommissioned>
      <typens:outlets>
        <typens:outlet>
          <typens:id>1</typens:id>
          <typens:outletId>1</typens:outletId>
          <typens:pduId>22</typens:pduId>
          <typens:deviceId>1</typens:deviceId>
          <typens:name>outlet1</typens:name>
          <typens:state>ON</typens:state>
        </typens:outlet>
        <typens:outlet>
          <typens:id>2</typens:id>
          <typens:outletId>2</typens:outletId>
          <typens:pduId>22</typens:pduId>
          <typens:deviceId>1</typens:deviceId>
          <typens:name>outlet2</typens:name>
          <typens:state>ON</typens:state>
        </typens:outlet>
      </typens:outlets>
    </typens:device>
  </typens:entities>
</typens:findEdmEntitiesResponse>
```

In this example the device has two outlets associated with it, each of which is on.

Step 2: Switch outlet off

The next step is for the example application to request Power IQ to switch the power of this device off. The external key is used in this example to identify the device.

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:edmEntityPowerControl
xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:commands>
    <typens:command>
      <typens:device>
        <typens:externalKey>device-0044</typens:externalKey>
      </typens:device>
      <typens:powerControl>
        <typens:state>OFF</typens:state>
      </typens:powerControl>
    </typens:command>
  </typens:commands>
</typens:edmEntityPowerControl>
```

Power IQ will immediately respond with a "Job created successfully message" and the corresponding Job ID (in this example it is 1234).

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:edmEntityPowerControlResponse
xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:result>
    <typens:message>Job created successfully</typens:message>
  </typens:result>
  <typens:job>
    <typens:id>1234</typens:id>
  </typens:job>
</typens:edmEntityPowerControlResponse>
```

Step 3: Check status of job

The last step is to verify the status of the job request:

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:getJob xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:job>
    <typens:id>1234</typens:id>
  </typens:job>
</typens:getJob>
```

In this example response, the job is still active, meaning that the power control operation has not completed yet.

```
<?xml version="1.0" encoding="UTF-8"?>
<typens:getJobResponse xmlns:typens="http://www.raritan.com/poweriq/integration/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.raritan.com/poweriq/integration/types
poweriq-integration.xsd ">
  <typens:result>
    <typens:message>Found job 1234</typens:message>
  </typens:result>
  <typens:job>
    <typens:id>1234</typens:id>
    <typens:description>Job Description</typens:description>
    <typens:endTime>Thu Oct 29 17:23:45</typens:endTime>
    <typens:startTime>Thu Oct 29 17:22:45</typens:startTime>
    <typens:status>ACTIVE</typens:status>
  </typens:job>
</typens:getJobResponse>
```

Chapter 7 FindEDMEntities Examples

The EdmService findEdmEntities method provides a searching mechanism for locating EDM entities. The following examples illustrate a number of ways in which findEdmEntities can be used. The examples are in XML, with the SOAP elements removed for brevity. You will have to map the XML elements to your xml binding framework accordingly.

In This Chapter

Returning Multiple Entities Using the IN Predicate	26
Using the LIKE Predicate	27
Finding Exact Matches Using the EQUAL Predicate	28
Creating Queries With Multiple Conditions.....	29

Returning Multiple Entities Using the IN Predicate

In this example one invocation of the findEdmEntities method is used to retrieve multiple devices by externalKey using the in predicate. Note that queries can only be scoped against one EDM entity type, in the example below, the query will execute against Device entities.

```
<q1:findEdmEntities
xmlns:q1="http://www.raritan.com/poweriq/integration/
types">
<q1:type>Device</q1:type>
<q1:query>
<q1:ordering>
<q1:order direction="ascending" field="name"/>
</q1:ordering>
<q1:conditions>
<q1:condition>
<q1:field>externalKey</q1:field>
<q1:predicate>in</q1:predicate>
<q1:values>
<q1:value>Device -- 1</q1:value>
<q1:value>Device -- 2</q1:value>
<q1:value>Device -- 3</q1:value>
</q1:values>
</q1:condition>
</q1:conditions>
</q1:query>
</q1:findEdmEntities>
```

The message above will result in the following sql query execution in Power IQ:

```
SELECT * FROM "devices" WHERE
(external_key IN (E'Device -- 1',E'Device -- 2',E'Device
-- 3'))
ORDER BY name ASC LIMIT 50 OFFSET 0
```

Using the LIKE Predicate

This example returns all Device entities whose externalKey has the name "Device" in it.

```
<q1:findEdmEntities
xmlns:q1="http://www.raritan.com/poweriq/integration/
types">
<q1:type>Device</q1:type>
<q1:query>
<q1:ordering>
<q1:order direction="ascending" field="name"/>
</q1:ordering>
<q1:conditions>
<q1:condition>
<q1:field>externalKey</q1:field>
<q1:predicate>like</q1:predicate>
<q1:values>
<q1:value>Device</q1:value>
</q1:values>
</q1:condition>
</q1:conditions>
</q1:query>
</q1:findEdmEntities>
```

The message above will result in the following sql query execution in Power IQ:

```
SELECT * FROM "devices" WHERE
(external_key LIKE E'%Device%')
ORDER BY name ASC LIMIT 50 OFFSET 0
```

Finding Exact Matches Using the EQUAL Predicate

Use the equal predicate to find exact matches.

```
<q1:findEdmEntities
xmlns:q1="http://www.raritan.com/poweriq/integration/
types">
<q1:type>Device</q1:type>
<q1:query>
<q1:ordering>
<q1:order direction="ascending" field="name"/>
</q1:ordering>
<q1:conditions>
<q1:condition>
<q1:field>externalKey</q1:field>
<q1:predicate>equal</q1:predicate>
<q1:values>
<q1:value>Device -- 1</q1:value>
</q1:values>
</q1:condition>
</q1:conditions>
</q1:query>
</q1:findEdmEntities>
```

The message above will result in the following sql query execution in Power IQ:

```
SELECT * FROM "devices" WHERE
(external_key = E'Device -- 1')
ORDER BY name ASC LIMIT 50 OFFSET 0
```

Creating Queries With Multiple Conditions

By combining multiple conditions more advanced queries can be made. Currently, each query condition is combined with an AND. At this time there is no way to override the operator used to combine query conditions.

```
<q1:findEdmEntities
xmlns:q1="http://www.raritan.com/poweriq/integration/
types">
<q1:type>Device</q1:type>
<q1:query>
<q1:ordering>
<q1:order direction="ascending" field="name"/>
</q1:ordering>
<q1:conditions>
<q1:condition>
<q1:field>customer</q1:field>
<q1:predicate>equal</q1:predicate>
<q1:values>
<q1:value>IBM</q1:value>
</q1:values>
</q1:condition>
<q1:condition>
<q1:field>deviceType</q1:field>
<q1:predicate>equal</q1:predicate>
<q1:values>
<q1:value>Oracle Server</q1:value>
</q1:values>
</q1:condition>
</q1:conditions>
</q1:query>
</q1:findEdmEntities>
```

The message above will result in the following sql query execution in Power IQ:

```
SELECT * FROM "devices" WHERE  
(customer = E'IBM' AND device_type = E'Oracle Server')  
ORDER BY name ASC LIMIT 50 OFFSET 0
```

Appendix A References

Web Services Security (WS-Security)

Version 1.0 05 05 Apr 2002. Updated 01 Mar 2004

<http://www.ibm.com/developerworks/library/specification/ws-secure/>

Web Services Security UsernameToken Profile 1.0

OASIS Standard 200401, March 2004

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

Index

A

- Aisle • 10
- Asynchronous Example, Job with a Problem • 19
- Asynchronous Example, Successful Job • 17
- Authorization • 16

C

- Creating Queries With Multiple Conditions • 29

D

- Data Center • 11
- Device • 9
- Device Group • 9

E

- Enable the WS API • 14

F

- Fields in Power IQ • 8
- FindEDMentities Examples • 25
- Finding Exact Matches Using the EQUAL Predicate • 28
- Floor • 11
- Functional Description • 7, 8

G

- Getting started • 7, 14

I

- Internationalization • 19
- Introduction • 5

L

- List of operations in the API • 12

P

- PDUs • 8
- Port Requirement • 14
- Power Control Examples • 20

R

- Rack • 10
- References • 31
- Requests and commands • 12

Returning Multiple Entities Using the IN

- Predicate • 26
- Room • 11
- Row • 10

S

- Security • 15
- Step 1
 - Current state of device • 20
- Step 2
 - Switch outlet off • 22
- Step 3
 - Check status of job • 23
- Synchronous and Asynchronous Method Calls • 16
- Synchronous Example • 17

T

- Technical description • 14
- Technical Description Summary • 7

U

- Use case 1
 - Asset management system/data synchronization • 6
- Use case 2
 - Power control • 6
- Use Cases • 6
- Using the LIKE Predicate • 27

W

- WSDL Files • 14

▶ **U.S./Canada/Latin America**

Monday - Friday
8 a.m. - 6 p.m. ET
Phone: 800-724-8090 or 732-764-8886
For CommandCenter NOC: Press 6, then Press 1
For CommandCenter Secure Gateway: Press 6, then Press 2
Fax: 732-764-8887
Email for CommandCenter NOC: tech-ccnoc@raritan.com
Email for all other products: tech@raritan.com

▶ **China**

Beijing

Monday - Friday
9 a.m. - 6 p.m. local time
Phone: +86-10-88091890

Shanghai

Monday - Friday
9 a.m. - 6 p.m. local time
Phone: +86-21-5425-2499

GuangZhou

Monday - Friday
9 a.m. - 6 p.m. local time
Phone: +86-20-8755-5561

▶ **India**

Monday - Friday
9 a.m. - 6 p.m. local time
Phone: +91-124-410-7881

▶ **Japan**

Monday - Friday
9:30 a.m. - 5:30 p.m. local time
Phone: +81-3-3523-5991
Email: support.japan@raritan.com

▶ **Europe**

Europe

Monday - Friday
8:30 a.m. - 5 p.m. GMT+1 CET
Phone: +31-10-2844040
Email: tech.europe@raritan.com

United Kingdom

Monday - Friday
8:30 a.m. to 5 p.m. GMT
Phone +44(0)20-7090-1390

France

Monday - Friday
8:30 a.m. - 5 p.m. GMT+1 CET
Phone: +33-1-47-56-20-39

Germany

Monday - Friday
8:30 a.m. - 5:30 p.m. GMT+1 CET
Phone: +49-20-17-47-98-0
Email: rg-support@raritan.com

▶ **Melbourne, Australia**

Monday - Friday
9:00 a.m. - 6 p.m. local time
Phone: +61-3-9866-6887

▶ **Taiwan**

Monday - Friday
9 a.m. - 6 p.m. GMT -5 Standard -4 Daylight
Phone: +886-2-8919-1333
Email: support.apac@raritan.com