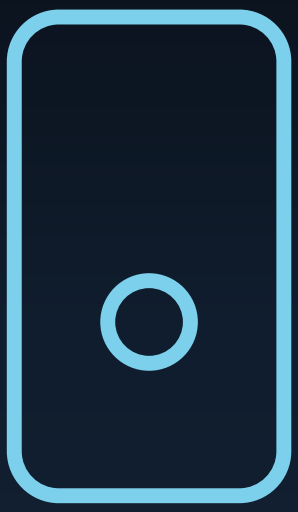


riskband™

ARIES

RiskBand API Programming Guide
Version .31 (current with 22309)

November 2021

Published November 2021.

Contents

Preface **7**

Chapter 1 Introduction to Using the RiskBand API **9**
Calling RESTful Calls Directly 9
Calling the Java CLI 13
Calling the RiskBand Java SDK 15

Chapter 2 More About the RiskBand API **17**
About Authenticating with the RiskBand ARIES Manager 17
About LoginByUser 17
Getting the passwordHash 19
Allowing the Java CLI to Authenticate for You 19
Authenticating with the RiskBand Java SDK 20

Chapter 3 More About Direct RESTful Calls **21**
About cURL Examples 21
About JSON Payload Lengths 21
About Modify Commands 22
Example of ModifyUser Using cURL 23

Chapter 4 More About the RiskBand Java CLI **27**
Downloading and Installing the Java CLI 28
Using the Java CLI with PowerShell 29
About CLI Command Line Help 31
Breaking Commands into Multiple Lines 33
Special Features of the CLI 34
About User Name and Password Authentication 34
About the responseField display 35
Unary Options in the CLI 36
About the `-debug` Flag 37
About the -prettyJson flag 39
About the -verbose flag 40
About Modify Commands 41
Example of ModifyUser using the CLI 42

Chapter 5 More About the RiskBand Java SDK **45**
Downloading and Installing the Java SDK 46
About JavaDocs for the RiskBand Java SDK 47
About Modify Commands 47
Example of ModifyUser using the Java SDK 48
Full Listing of ModifyUserExample.java 50
ModifyUserExample.java Output 52

Chapter 6 Example: Creating a Geofence using a Web Browser 53
Calling LoginByUser 54
Calling GetOrganizationDivisions 55
Calling CreateGeoFence 56
Calling GetGeoFences 57

Chapter 7 Example: Creating a Geofence using cURL 59
Calling LoginByUser 60
Calling GetOrganizationDivisions 61
Calling CreateGeoFence 63
Calling GetGeoFences 64

Chapter 8 Example: Creating a Geofence using PowerShell 67
Calling LoginByUser 67
Calling GetOrganizationDivisions 68
Calling CreateGeoFence 70
Calling GetGeoFences 71
Full Listing of CreateGeoFence PS Script 73
Full Listing of CreateGeoFence PS Script Output 76

Chapter 9 Example: Creating a Geofence Using the Java CLI 81
Calling LoginByUser 81
Calling GetOrganizationDivisions 82
Calling CreateGeoFence 83
Calling GetGeoFences 84

Chapter 10 Example: Creating a Geofence using the Java SDK 85
Calling RiskBandClientImpl 85
Calling GetOrganizationDivisions 86
Calling CreateGeoFence 86
Calling GetGeoFences 87
Full Listing of CreateGeoFenceExample.java 89
Full Listing CreateGeoFenceExample.java Output 91

Chapter 11 Receiving Notifications from the RiskBand ARIES Manager 93
Configuring Your Server for RiskBand Notifications 93
About Notification Payloads 94
EmergencyRegisteredNotification Payload 94
NewEmergencyArtifactNotification Payload 99
EmergencyClosedNotification Payload 99
DeviceAssignedNotification Payload 100
DeviceUnassignedNotification Payload 105
DeviceCalledHomeNotification Payload 105
DeviceOfflineNotification Payload 107

Chapter 12 Working with Emergencies 109
Calling GetEmergencies 109
GetEmergencies Response Payload 110
Calling GetEmergencies from a Browser 110
Calling GetEmergencies using cURL 111
Calling GetEmergencies using the Java CLI 111
Calling GetEmergencies from PowerShell 112
Calling GetEmergencies using the Java SDK 114
Receiving Emergency Notifications 115
Getting Emergency Contact Information 116
Getting Emergency GPS Coordinates 117
Getting Emergency Artifact Objects 121
Downloading Emergency Artifacts 122
Downloading Emergency Artifacts with cURL 122
Downloading Emergency Artifacts with the Java CLI 122
Closing an Emergency 124
Adding Additional Emergency Notes 124
Getting Emergency Notes 125

Chapter 13 RiskBand API Syntax Examples 129
ActivateManufactureInProgressDevice 129
CloseEmergencyByEmergencyResponder 129
CloseEmergencyByUser 130
CreateActionMessage 130
CreateAdministratorPermission 130
CreateDeviceModel 131
CreateDeviceSecurityPolicy 131
CreateEmergencyNote 132
CreateEmergencyResponsePolicy 132
CreateGeoFence 134
CreateOrganizationPermissions 134
CreateUser 134
DeleteUser 134
DownloadEmergencyArtifact 134
DownloadGuiBuild 134
GetAdministratorPermissions 135
GetDevices 135
GetEmergencies 137
GetEmergencyArtifacts 138
GetEmergencyArtifactsCount 138
GetEmergencyContactInformation 138
GetEmergencyContactInformations 139
GetEmergencyNotes 139
GetGeoFences 140
GetGpsPositions 140
GetOrganization 140
GetOrganizations 140

GetOrganizationDivisions 141
GetOrganizationPermissions 141
GetUser 141
GetUsers 142
LoginByUser 143
ModifyUser 143
Ping 143
RegisterDevice 144
ReprogramDeviceAuthorization 145
ResetUserPassword 146
RiskBandClientImpl 146
UploadDeviceFirmwareBuild 146
UploadGuiBuild 146

Chapter 14 Utilities 147
Require All Users to Change Password Next Login 148
Full Listing of ModifyAllUsers_MustChangePassword.ps1 Utility 149
ModifyAllUsers_MustChangePassword.ps1 Output 150
Create User that is an Organizational Administrator 151
Full Listing of createOrgAmind.bat File 152
createOrgAdminUser.bat Output 153
Converting UNIX Time with Perl 153

Chapter 15 Contacting RiskBand 155

Preface

RiskBand periodically releases revisions of its software and hardware. Some functions described in this document might not be supported by all versions of the software or hardware.

1 Introduction to Using the RiskBand API

The RiskBand provides a RESTful API for interacting with the RiskBand ARIES Manager. Each API call sends a JSON formatted request over HTTPS to the RiskBand ARIES Manager, and, if successful, the results are returned in JSON format.

There are three ways to use the RiskBand API:

- Calling RESTful Calls Directly on page 9
- Calling the Java CLI on page 13
- Calling the RiskBand Java SDK on page 15

Calling RESTful Calls Directly

Calling a RiskBand API directly requires you to create a URL that includes the address of the RiskBand ARIES Manager, the name of the API, and any required or desired options.

There are a variety of ways to make these HTTPS calls, and nearly all programming languages provide mechanisms that support this. By way of illustration, here are some examples of calling the `LoginByUser` API as one long URL. The URL used for these examples is

```
https://dev3.riskband.ws/?request=LoginByUser&name=acme%5c%5capier&passwordHas
h=vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
```

Figure 1: LoginByUser using Browser Address Bar

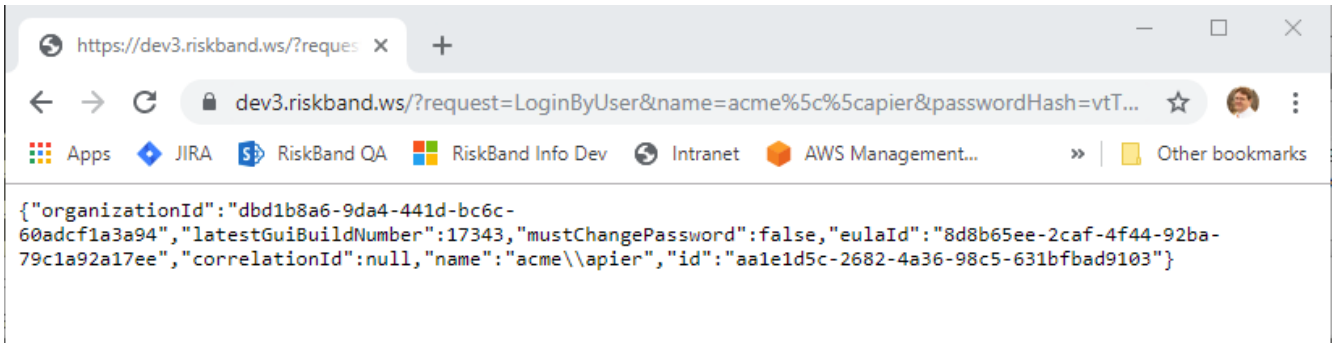


Figure 2: LoginByUser using Invoke-WebRequest from Powershell

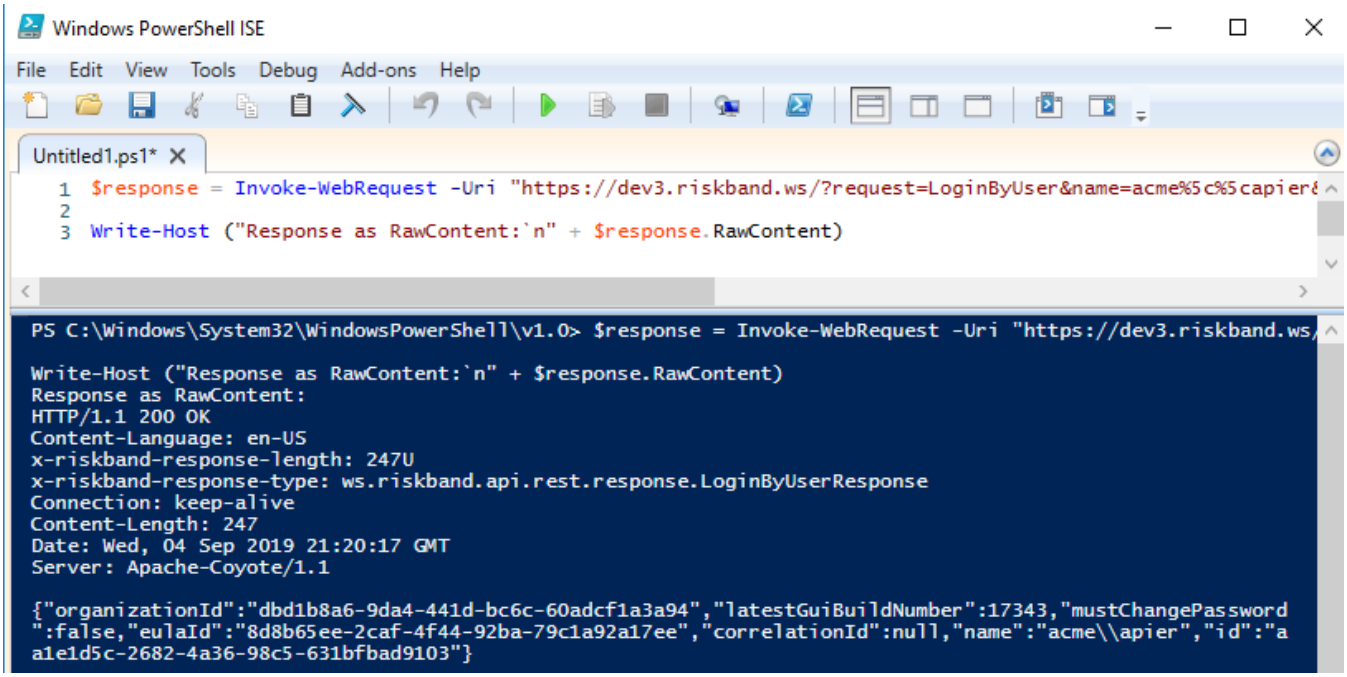


Figure 3: LoginByUser using cURL from Linux

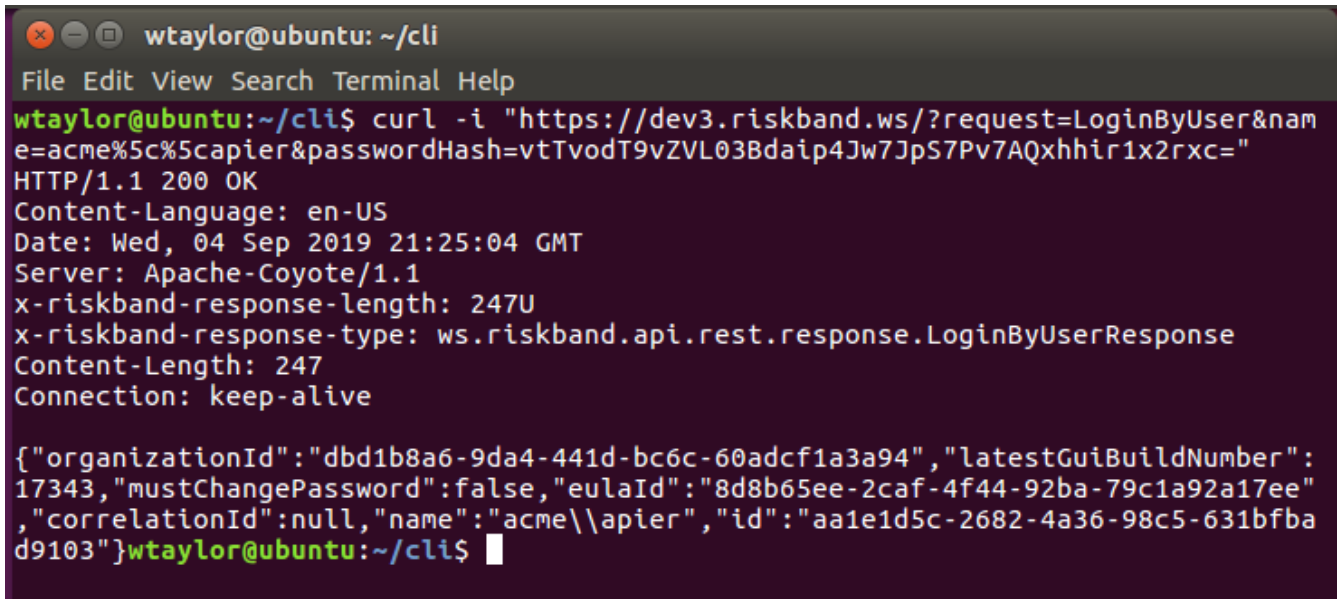
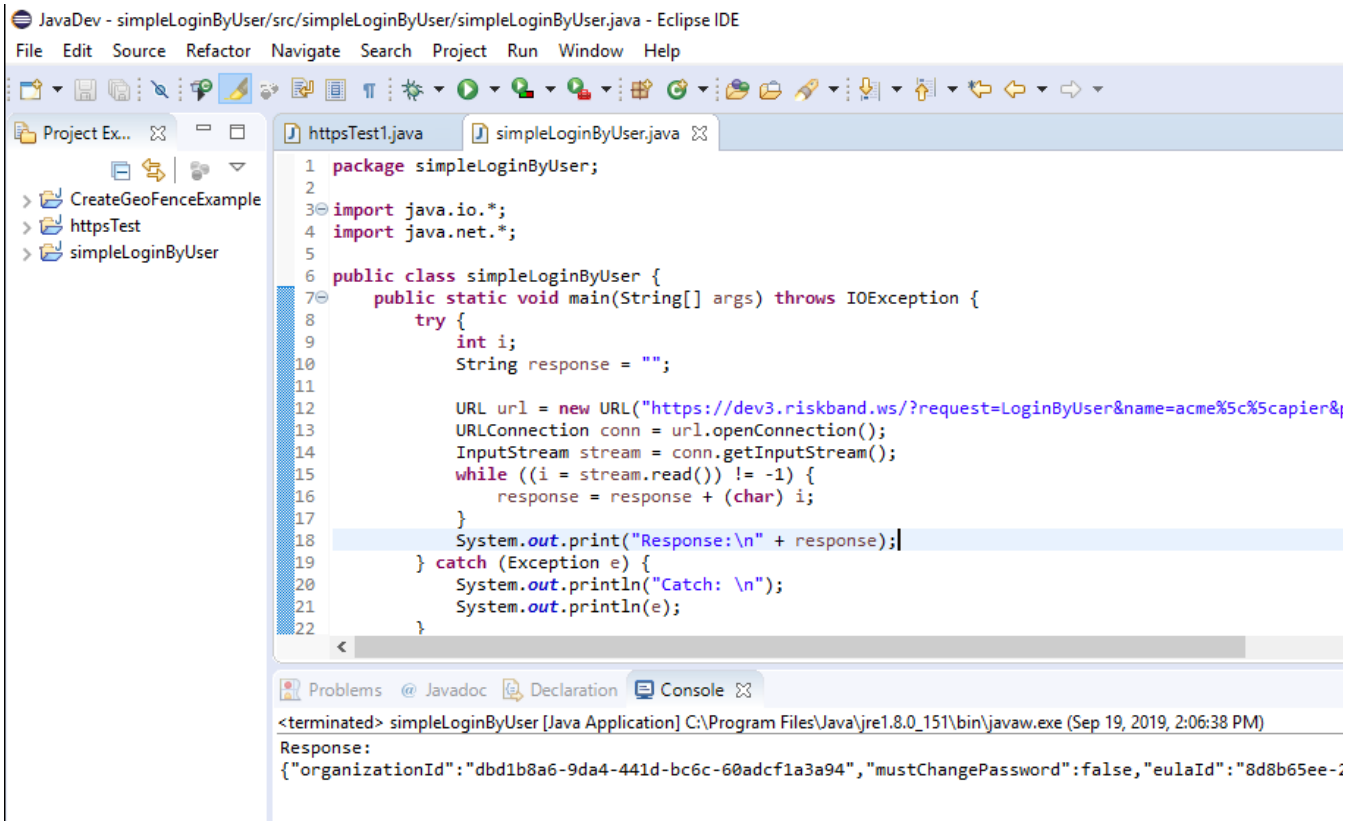


Figure 4: LoginByUser using URLConnection Class in Java



While these calls are syntacticly very simple, it can be difficult to visualize the JSON structure that is basis of the request. Consequently, to make it easier to see the "request payload," many API examples are provided using a "PUT" approach. That is, the JSON structure of the API request is defined separately in the script or in a file, and then the data or file is referenced in the call.

```
curl -iv -X PUT -H "Content-Length: 279" -H "Content-Type: application/json" -d @_loginByUser_json.json https://dev3.riskband.ws?LoginByUser=279U
```

Length JSON Data | JSON Data in File | URL RiskBand Server | API Name | Length JSON Data

```
Invoke-RestMethod -Uri https://dev3.riskband.ws?LoginByUser=297U -ContentType "application/json" -Method PUT -Body $json
```

URL RiskBand Server | API Name | Length JSON Data | JSON Data in Array

By way of illustration, here are examples of calling the `LoginByUser` API in cURL and Powershell using the PUT approach. In the cURL example the JSON structure is in a file called `_loginByUser_acmeApier_279.json`

Figure 5: LoginByUser using cURL in Linux

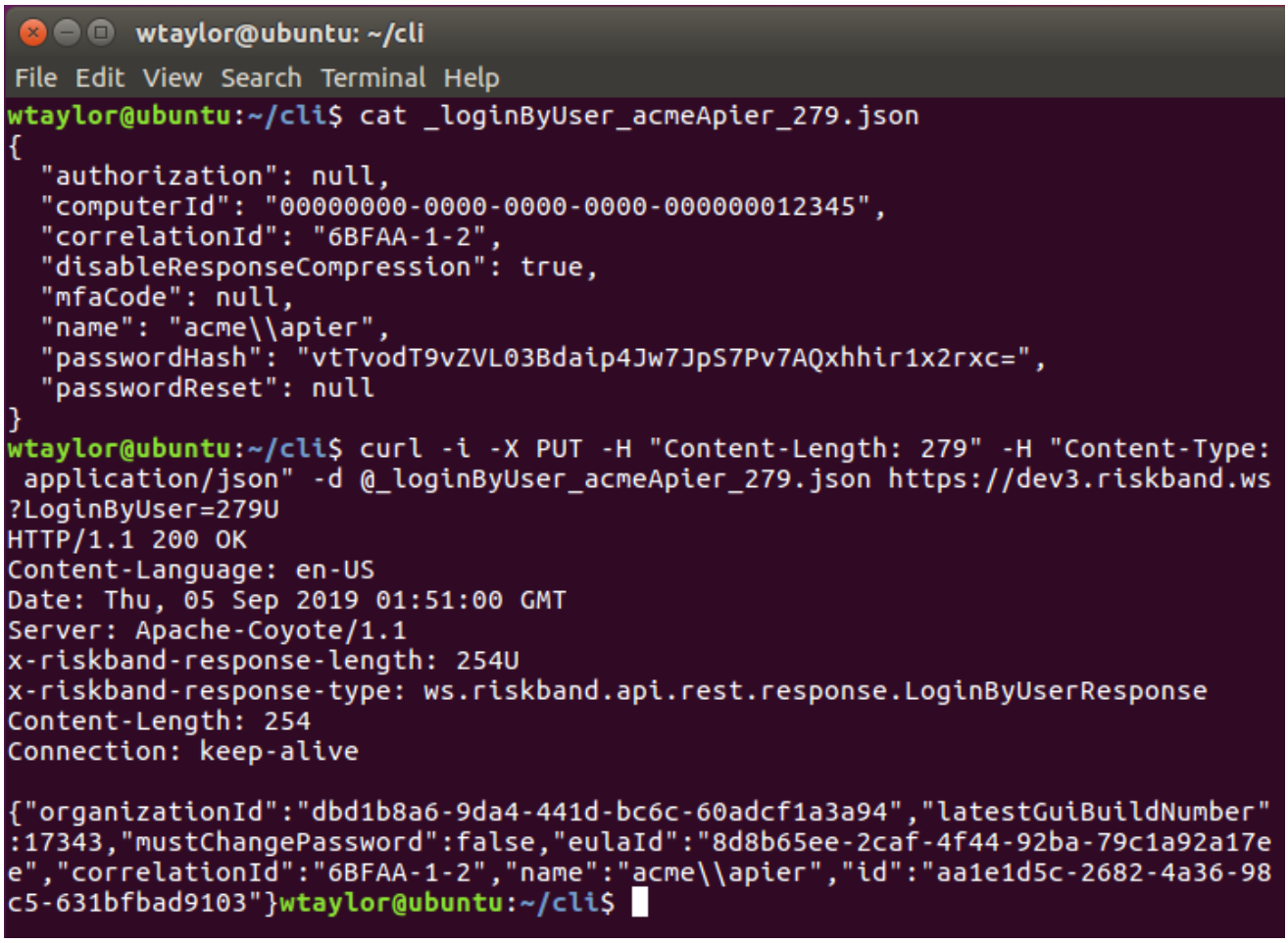
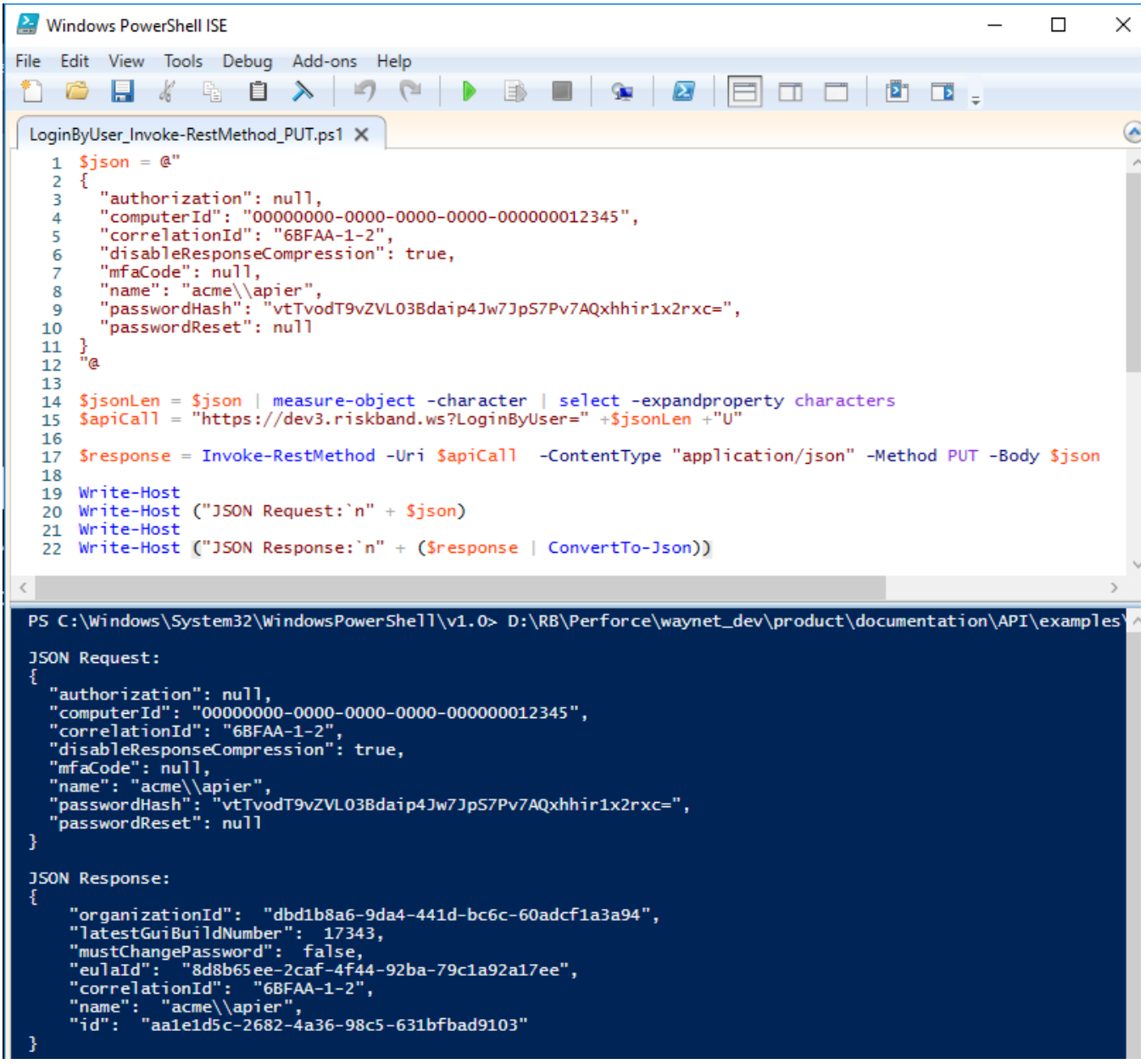


Figure 6: LoginByUser using PowerShell and Invoke-RestMethod



In this document, most of the examples for making direct API calls are shown using the Put approach.

Calling the Java CLI

The RiskBand Java CLI puts a "wrapper" around the RiskBand API calls to simplify the mechanics of calling the APIs. With the CLI, two executable files, `runRbCli.bat` and `runRbCLI.sh`, collect the API name and the associated options and then pass these to the `RiskBand-Java_CLI-•••••.jar` file. This file then makes the API call, converting the command line arguments to a JSON request structure and sending it to the server using HTTPS. An example of how the command line arguments are provided in Windows and LInux is shown here:

Figure 7: LoginByUser using CLI in Windows

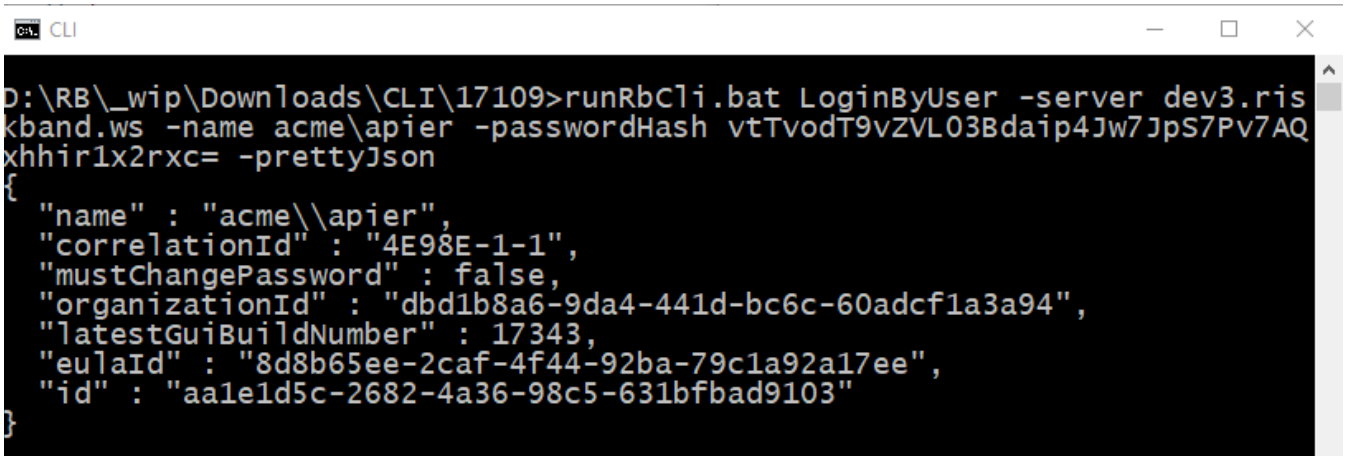
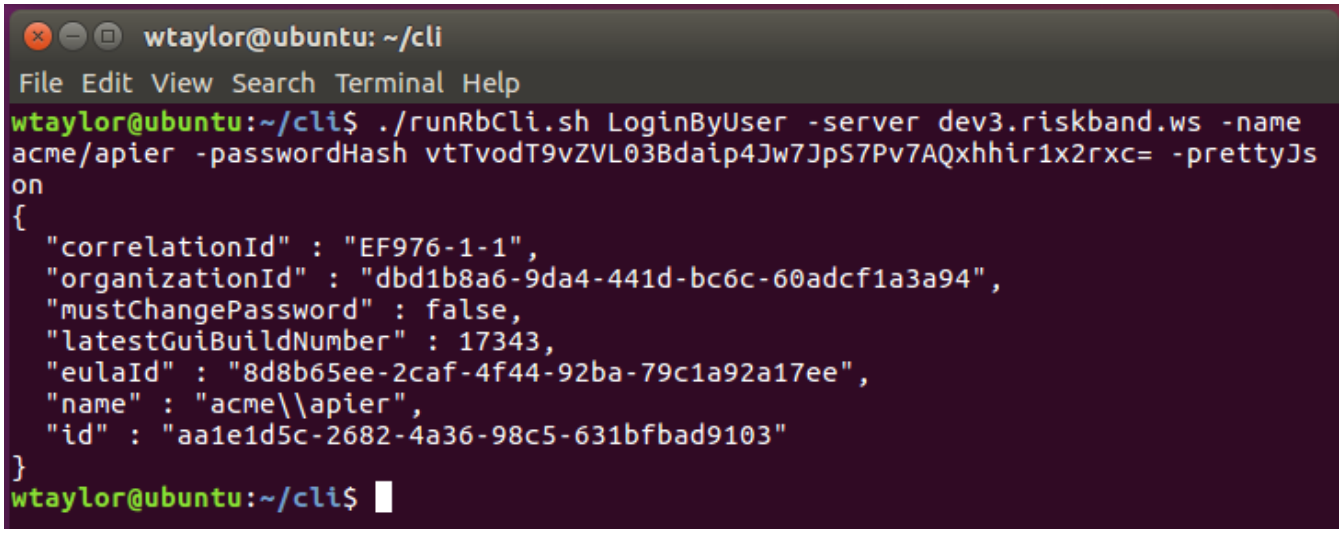


Figure 8: LoginByUser using CLI in LInux

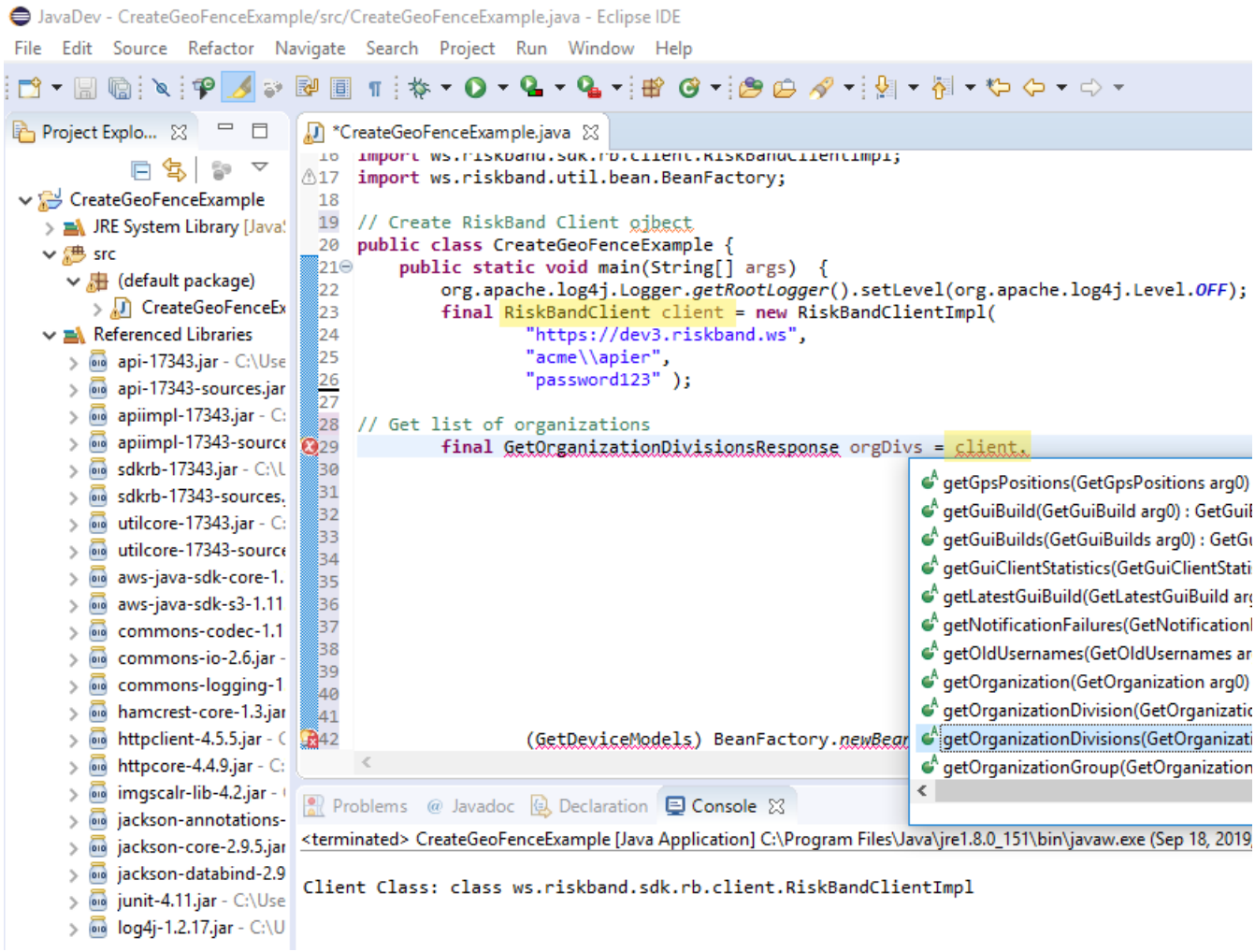


Note: The significant difference between the two examples is the use of backslash (Windows) or forward slash (Linux) when separating the user name from the organizational prefix: acme\apier vs. acme/apier. Most of the examples in this document will be given in the Windows format, so if you try an example on Linux and it fails, the first thing to check is the direction of slashes in your input.

Using the CLI can be an instructional way to learn about, or experiment with, the RiskBand API because the CLI provides command line help about what options are required for any given API. Additionally the `-debug` flag can provide considerable information about how the JSON request structure should be formatted. (see About the `-debug` Flag on page 37). For this reason the majority of API examples in this document are provided using the JAVA CLI.

Calling the RiskBand Java SDK

Using the RiskBand Java SDK eliminates much of the overhead of creating request strings and processing responses. Using the SDK, you can create a a *RiskBandClient* object which contains methods for accessing all of the RiskBand APIs.



Additionally, the SDK provides you with full compile-time type safety—or in other words, it ensures that you will get the correct response type for every request type.

2 More About the RiskBand API

This section describes some common features and aspects of the RiskBand API.

About Authenticating with the RiskBand ARIES Manager

Every API call must contain valid credentials in order to connect with the RiskBand ARIES Manager. The API accepts three different types of credentials:

- User — when a user is created in the RiskBand ARIES Manager, the user is assigned a password. The user can use this password to log in to the RiskBand ARIES Manager ARIES manager and, depending on the permissions granted, interact with system. The user can also use these credentials to authenticate an API call.
- Device — when a device is manufactured, it is programmed with device credentials that allow it to communicate with the RiskBand ARIES Manager. At regular intervals these credentials are rotated to protect the device against cloning or otherwise being individually compromised. All API calls made from RiskBand devices use device credentials.
- Emergency — Whenever an emergency is triggered, emergency provider credentials are created which may be used by the provider to view images taken by the device, to look up details about the user who triggered it, and to view other information relevant to that specific emergency. These credentials are always valid while the emergency is active; however, they are invalidated according to policy after the emergency has been closed. (The length of time emergency credentials are valid is determined by the field in the emergency response policy called *Provider Access After Closed*.)

In this document most of the examples show authentication using USER credentials.

About LoginByUser

When authenticating to the RiskBand ARIES Manager as type USER, your credentials consist of the UUID of your user object and a password hash of your password. As most people don't know their UUID, you can use `LoginByUser` to map your user name to its UUID.

For example, if you wanted to authenticate an API call as the user `acme\apier`, you would call `LoginByUser` with the string "`acme\apier`" and a hash of the password. (For more information about how to get the passwordHash see Getting the passwordHash on page 19.) Here's what the JSON data would look like for that call:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  }
```

The results of a successful call will contain the UUID for the user `acme\apier` in the “id” field:

```
{
  "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber": 17343,
  "mustChangePassword": false,
  "eulaId": "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "correlationId": "6BFAA-1-2",
  "name": "acme\\apier",
  "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

Subsequent API calls can then use an authorization sub-section, or authorization object, that looks like this:

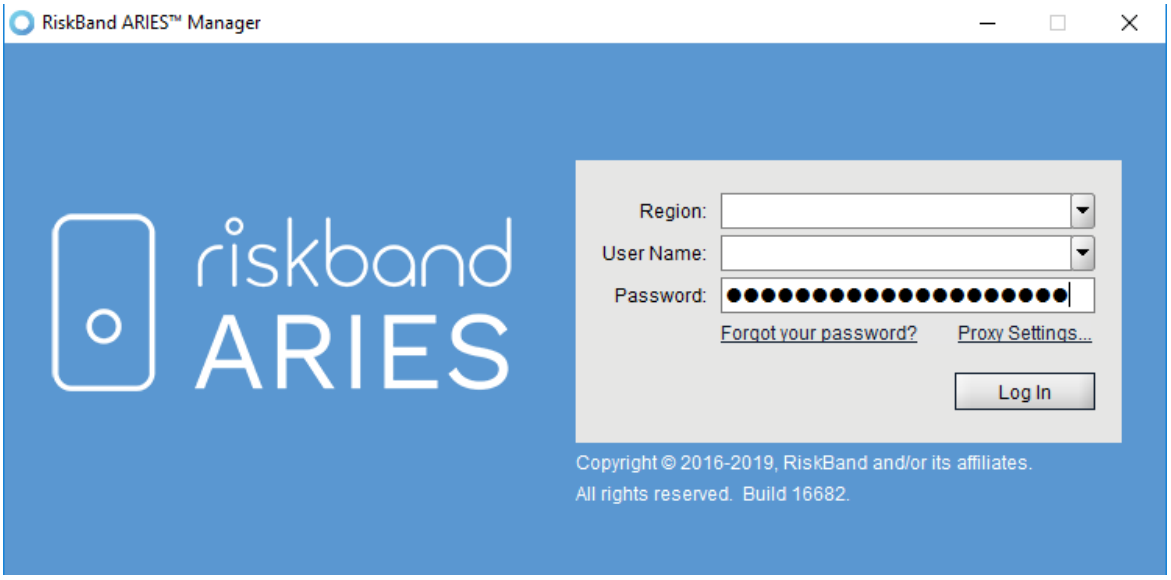
```
 "authorization": {
     "computerId": "00000000-0000-0000-0000-000000012345",
     "entityType": "USER",
     "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
     "mfaCode": null,
     "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
   }
```

In a larger context, the same authorization sub-section might look like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [
    {
      "property": "organizationId",
      "value": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "operator": "EQUALS"
    }
  ],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
```

Getting the passwordHash

You can get the hash for a user password using the RiskBand ARIES Manager client software. On the login screen, enter the password you want to hash. Then press **Ctrl-Alt-H**. This will put the `passwordHash` for that password in the paste buffer of your computer.



For example, the `passwordHash` for the password "password123" is vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=

Note: The JSON data for every API call is transmitted to the RiskBand ARIES Manager using HTTPS so the password hash is encrypted. Additionally, in logs and in console output the value for `passwordHash` is displayed as `<concealed>`.

Allowing the Java CLI to Authenticate for You

A convenient feature of the RiskBand Java CLI is that you can authenticate CLI calls with a user name and password. Behind the scenes, the CLI will get the UUID for your user and will convert the password to a hash for you.

For example, you can make the `GetOrganizationDivisions` call with user name and password:

```
runRbCli.bat GetOrganizationDivisions -server dev3.riskband.ws -user acme/apier
-pwd password123 -prettyJson
```

However, you can also specify the CLI command using the UUID and passwordHash:

```
runRbCli.bat GetOrganizationDivisions -server dev3.riskband.ws -authId
1d4524e7-d81e-4729-a670-0aca0c39cbd1 -authKey
AG0AF2yIutIiy/AFwn72C3i1Qj9GUXW2Nu16WgHIc0Q= -authType USER -prettyJson
```

CLI command examples use a mix of *user/pwd* and *authId/authKey* authentication.

Authenticating with the RiskBand Java SDK

With the RiskBand Java SDK you provide the endpoint, user name, and password when you create the RiskBand Client object. For example, the code to implement a client object looks like this:

```
final RiskBandClient client = new RiskBandClientImpl(
        "https://dev3.riskband.ws",
        "acme\\apier",
        "password123" );
```

The user credentials are stored as attributes in the client object, and, the SDK does the work of including them in request structures of sent by the Java SDK.

3 More About Direct RESTful Calls

The RiskBand server, or RiskBand ARIES Manager, provides RESTful services and resources that available over HTTPS. This allows you to make calls to the server directly without needing any intermediary libraries or frameworks.

About cURL Examples

Most of the cURL examples provided will use the following format:

```
curl -iv -X PUT -H "Content-Length: <JSONPayloadLength>" -H "Content-Type:
application/json" -d @<fileContainingJSONPayload>
https://<endPoint>?<APICall>=<JSONPayloadLength>U > <serverResponseFile>
```

where

- *JSONPayloadLength* — is the number of characters in the JSON payload.
- *fileContainingJSONPayload* — is the name of a file that contains the JSON payload.
- *endPoint* — is the URL of the RiskBand ARIES Manager
- *APICall* — is the name of the API being called
- *serverResponseFile* — is the name of a file that server's response will be redirected to.

An example cURL command, then, could look something like this:

```
curl -iv -X PUT -H "Content-Length: 279" -H "Content-Type: application/json" -d
@_loginByUser_acmeApier_279.json https://dev3.riskband.ws?LoginByUser=279U
```

About JSON Payload Lengths

If you are using the "PUT" approach to making API calls, where the JSON request structure is in a separate array or file, then you will need to supply the character length of the JSON request. In this document, JSON request structures are specified in this format:

```
{
  "authorization": null,
  "computerId": "00000000-0000-0000-0000-000000012345",
  "correlationId": "6BFAA-1-2",
  "disableResponseCompression": true,
  "mfaCode": null,
  "name": "acme\\apier",
  "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
  "passwordReset": null
}
```

For the purposes of the RiskBand API, the character length request structure above is 279 characters.

If you are specifying an array in Windows PowerShell or other programming language, there are functions available to calculate the length of a request. If, however, you make API calls by referencing the request structure in a file, the method for determining the

structure length is more complicated. Additionally, depending on the amount of white space and the kind of white space that is used in formatting the JSON structure, it can be a little tricky to determine the correct length.

The PowerShell measure-object cmdlet yields consistently accurate results, and the website http://www.charactercountonline.com also provide character counts that the RiskBand API accepts.

About Modify Commands

Generally, modify commands in the RiskBand API work in two ways:

- Like a create command — where you specify all the attributes you want for an existing object, and the object is essentially re-created—but keeps the original UUID. If you do not specify any *propertiesToModify* in the request structure, or you set it to NULL, only the attributes explicitly specified in the call will be set. That is, everything not specified will be set to NULL.
- Like a selective modify command — where you specify just the optional attributes you want to modify using the *propertiesToModify* flag

RiskBand recommends using the *propertiesToModify* flag when modifying objects.

In the following examples, the ModifyUser API is used to modify just two attributes for user acme\cchesteron:

- citizenship
- all clear passphrase

Example of ModifyUser Using cURL

Make the GetUser call to see the current JSON structure of acme\cchesteron user object and to get the UUID. The JSON request structure for GetUser looks like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "mfaCode": null,
    "entityType": "USER",
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
  },
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filter": {
    "property": "name",
    "value": "acme\\cchesteron"
  }
}
```

You can use cURL to make the GetUser call that references the request structure using a command like this:

```
curl -iv -X PUT -H "Content-Length: 384" -H "Content-Type: application/json" -d
@_GetUser_384.json  https://dev3.riskband.ws?GetUser=384U
```

The formatted output of the call looks like this:

```
HTTP/1.1 200 OK
Content-Language: en-US
Date: Thu, 05 Sep 2019 00:52:32 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 1151U
x-riskband-response-type: ws.riskband.api.rest.response.GetUserResponse
Content-Length: 1151
Connection: keep-alive

{
  "shortUserName": null,
  "incompleteRequiredCourses": [ ],
  "eulaUrl": "https://dev3.riskband.ws/legal.pdf",
  "eulaId": "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "correlationId": "RANDOM-0-002",
  "result": {
    "metadata": null,
    "groupId": null,
    "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
    "dateRegistered": 1567117762409,
    "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "phoneNumber": "606-606-6060",
    "primaryPhotoId": null,
    "emailAddress": null,
    "firstName": "Cogsworth",
    "lastName": "Chesterton",
    "mfaTimeoutInMins": null,
    "mustChangePassword": true,
    "gender": null,
    "dateOfBirth": null,
    "lifeSavingInformation": "Asthma",
    "phoneVerified": true,
    "dateLastActivity": null,
    "geosFusionUserId": null,
    "mfaCurrentCode": 0,
    "mfaCurrentDate": 0,
    "passphraseAllClear": "flotsam",
    "passphraseDuress": null,
    "citizenship": null,
    "restrictAccessToKnownComputers": false,
    "restrictAccessToIps": null,
    "emailVerified": true,
    "passwordResetEmailCode": null,
    "passwordResetPhoneCode": null,
    "passwordResetExpirationDate": null,
    "passwordHash": "<concealed>",
    "googleApiKey": null,
    "emergencyContactInformationId": null,
    "suspended": false,
    "description": null,
    "name": "acme\\cchesteron",
    "id": "f4f386f1-fa69-479b-a663-b48576fb5410"
  }
}
```

Then make the ModifyUser call to change citizenship from null to USA and to change the all clear passphrase from “flotsam” to “jetsam.” Any attributes that are not required and are not changed should be left out of the request structure:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "mfaCode": null,
    "entityType": "USER",
    "passwordHash": "epssEzyyw9RYHu0uGT2zzurQr0mL6RqQWz3anOYIHbk=",
    "id": "dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6"
  },
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "name": "acme\\cchesterton",
  "lastName": "Chesterton",
  "firstName": "Cogsworth",
  "id": "f4f386f1-fa69-479b-a663-b48576fb5410",
  "propertiesToModify": [
    "citizenship",
    "passphraseAllClear"
  ],
  "citizenship": "USA",
  "passphraseAllClear": "jetsam"
}
```

You can use cURL to make the ModifyUser call that references the request structure using a command like this:

```
curl -iv -X PUT -H "Content-Length: 571" -H "Content-Type: application/json" -d
@_ModifyUser_571.json  https://dev3.riskband.ws?ModifyUser=571U
```

The formatted output of the call looks like this:

```
HTTP/1.1 200 OK
Content-Language: en-US
Date: Thu, 05 Sep 2019 00:57:18 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 1087U
x-riskband-response-type: ws.riskband.api.rest.response.GetUserResponse
Content-Length: 1087
Connection: keep-alive

{
  "shortUserName": null,
  "incompleteRequiredCourses": null,
  "eulaUrl": null,
  "eulaId": null,
  "correlationId": "RANDOM-0-002",
  "result": {
    "metadata": null,
    "groupId": null,
    "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
    "dateRegistered": 1567117762409,
    "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "phoneNumber": "606-606-6060",
    "primaryPhotoId": null,
    "emailAddress": null,
    "firstName": "Cogsworth",
    "lastName": "Chesterton",
    "mfaTimeoutInMins": null,
    "mustChangePassword": true,
    "gender": null,
    "dateOfBirth": null,
    "lifeSavingInformation": "Asthma",
    "phoneVerified": true,
    "dateLastActivity": null,
    "geosFusionUserId": null,
    "mfaCurrentCode": 0,
    "mfaCurrentDate": 0,
    "passphraseAllClear": "jetsam",
    "passphraseDuress": null,
    "citizenship": "USA",
    "restrictAccessToKnownComputers": false,
    "restrictAccessToIps": null,
    "emailVerified": true,
    "passwordResetEmailCode": null,
    "passwordResetPhoneCode": null,
    "passwordResetExpirationDate": null,
    "passwordHash": "<concealed>",
    "googleApiKey": null,
    "emergencyContactInformationId": null,
    "suspended": false,
    "description": null,
    "name": "acme\\cchesteron",
    "id": "f4f386f1-fa69-479b-a663-b48576fb5410"
  }
}
```

4 More About the RiskBand Java CLI

The RiskBand Java CLI is easiest way to use RiskBand APIs. The syntax for specifying parameters for a call is somewhat simplified, and, if an error occurs, command line help displays the correct syntax for the command as well as possible reasons for the error. Additionally, if you use the *-prettyJson* or *-debug* options with a working command, it can provide insight into structure of JSON payloads that can be applied to creating direct RESTful calls or programmatic Java SDK calls.

In order to use the Java CLI your computer will need to have the following software installed:

- A Java runtime environment (JRE) version 1.7 or later
- The contents of the `RiskBand-Java-CLI-•••••.zip` file which contains `RiskBand-Java_CLI-•••••.jar`.

Downloading and Installing the Java CLI

To download and install the RiskBand Java CLI:

1. Download `RiskBand-Java-CLI-•••••.zip` from https://downloads.riskband.com/index.html.
2. Place the file in the desired directory or folder and unzip the file.

The zip file also contains the two executable files, `runRbCli.bat` and `runRbCLI.sh`, which collect the command line arguments and execute the `java` command. Both files expect `RiskBand-Java_CLI-•••••.jar` to be located in the current directory.

For example, this is how running the CLI might look in a Windows command shell:

```
> dir
 Volume in drive D is DATA_BLUE
 Volume Serial Number is EC2B-7B86

 Directory of D:\RB\Downloads\CLI\16682

08/01/2019  03:00 PM    <DIR>          .
08/01/2019  03:00 PM    <DIR>          ..
07/16/2019  09:09 AM             6,426 EXAMPLES.txt
07/16/2019  09:09 AM               607 LICENSE.txt
07/16/2019  09:09 AM             5,714 README.txt
07/16/2019  09:09 AM         8,224,005 RiskBand-Java-CLI-16682.jar
07/16/2019  09:09 AM               463 runRbCli.bat
07/16/2019  09:09 AM               306 runRbCli.sh
               6 File(s)      8,237,521 bytes
               2 Dir(s)  415,681,667,072 bytes free

D:\RB\Downloads\CLI\16682> runRbCli.bat LoginByUser -server dev3.riskband.ws
-name acme\apier -passwordHash vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
-prettyJson
{
  "name" : "acme\\apier",
  "correlationId" : "22700-1-1",
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "mustChangePassword" : false,
  "latestGuiBuildNumber" : 17343,
  "eulaId" : "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
```

This is an example of how it might look on UNIX/Linux:

```
wtaylor@ubuntu:~/cli$ ls -al
total 15192
drwxrwxr-x  2 wtaylor wtaylor    4096 Sep  4 19:21 .
drwxr-xr-x 21 wtaylor wtaylor    4096 Sep  4 19:21 ..
-r--r--r--  1 wtaylor wtaylor    6426 Jul 16 09:09 EXAMPLES.txt
-r--r--r--  1 wtaylor wtaylor     607 Jul 16 09:09 LICENSE.txt
-r--r--r--  1 wtaylor wtaylor    5714 Jul 16 09:09 README.txt
-r-xr-xr-x  1 wtaylor wtaylor 8224005 Jul 16 09:09 RiskBand-Java-CLI-16682.jar
-rw-rw-r--  1 wtaylor wtaylor 7293959 Aug 20 14:53 RiskBand-Java-CLI-16682.zip
-r-xr-xr-x  1 wtaylor wtaylor     463 Jul 16 09:09 runRbCli.bat
-r-xr-xr-x  1 wtaylor wtaylor     306 Jul 16 09:09 runRbCli.sh

wtaylor@ubuntu:~/cli$ ./runRbCli.sh LoginByUser -server dev3.riskband.ws -name
acme/apier -passwordHash vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
-prettyJson
{
  "correlationId" : "18EFB-1-1",
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "mustChangePassword" : false,
  "latestGuiBuildNumber" : 17343,
  "eulaId" : "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "name" : "acme\\apier",
  "id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

Using the Java CLI with PowerShell

Using the Java CLI with PowerShell is probably the best way to experiment with and to learn how to use RiskBand APIs. This is because

- The Java CLI offers command line help about which options are required and which are optional
- The *-debug* option available for CLI commands shows the JSON request that is being sent to the server
- PowerShell offers a convenient way to convert JSON results to an addressable array.

For example, in a PowerShell script, the output from the command comes back as unformatted JSON. But it is relatively easy to display the output in a formatted way using the Format-List or the ConvertTo-JSON cmdlets.. Additionally, using the ConvertFrom-Json cmdlet, the JSON structure can be converted into an array whose elements can be easily addressed and saved to variables.

```
$response = .\runRbCli.bat LoginByUser -server dev3.riskband.ws -name acme\apier
-passwordHash vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -prettyJson
Write-Host ("`n Response as unformatted JSON:`n`n " + $response)

Write-Host ("`n Response as formatted JSON:")
$response | Format-List -Property *

# Convert JSON response to an array
$responseJSONobject = $response | ConvertFrom-Json
Write-Host ("`n Response converted from JSON to an array :`n`n " +
$responseJSONobject)

Write-Host ("`n `$responseJSONobject.id = " + $responseJSONobject.id)
Write-Host (" `$responseJSONobject.organizationId = " +
$responseJSONobject.organizationId)
```

The output of the script looks like this:

```
PS > ./LoginByUser_calling_runclibat_and_savingResponse.ps1

 Response as unformatted JSON:

 {   "name" : "acme\\apier",   "correlationId" : "99564-1-1",   "organizationId" :
"dbd1b8a6-9da4-441d-bc
6c-60adcf1a3a94",   "mustChangePassword" : false,   "latestGuiBuildNumber" :
17343,   "eulaId" : "8d8b65e
e-2caf-4f44-92ba-79c1a92a17ee",   "id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103" }

 Response as formatted JSON:
{
  "name" : "acme\\apier",
  "correlationId" : "99564-1-1",
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "mustChangePassword" : false,
  "latestGuiBuildNumber" : 17343,
  "eulaId" : "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}

 Response converted from JSON to an array :

 @{name=acme\apier; correlationId=99564-1-1;
organizationId=dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94; mustCha
ngePassword=False; latestGuiBuildNumber=17343;
eulaId=8d8b65ee-2caf-4f44-92ba-79c1a92a17ee; id=aa1e1d5c-2
682-4a36-98c5-631bfbad9103}

 $responseJSONobject.id = aa1e1d5c-2682-4a36-98c5-631bfbad9103
 $responseJSONobject.organizationId = dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94
```

For another example of how this can be implemented see the utility description Require All Users to Change Password Next Login on page 148.

About CLI Command Line Help

If you enter either `runRbCli.bat` or `runRbCLI.sh` without any parameters a list of all the supported CLI commands displays:

```
$ ./runRbCli.sh

Usage:

AcceptEula -server <String> [-user <String>] [-pwd <String>] -eulaId <UUID>
-userId <UUID> [-id <UUID>] [-disableResponseCompression <boolean>]
[-authComputerId <UUID>] [-authId <UUID>] [-authKey <String>] [-authMfaCode
<Integer>] [-authType <USER|DEVICE|EMERGENCY>] [-correlationId <String>]
[-debug] [-prettyJson] [-responseField <String>] [-verbose]

AssignDevice -server <String> [-user <String>] [-pwd <String>] -id <UUID> -userId
<UUID> [-correlationId <String>] [-disableResponseCompression <boolean>]
[-authComputerId <UUID>] [-authId <UUID>] [-authKey <String>] [-authMfaCode
<Integer>] [-authType <USER|DEVICE|EMERGENCY>] [-debug] [-prettyJson]
[-responseField <String>] [-verbose]
. . .
```

If you enter the `runRBCli` command and include just the API name, then command line help specific to that call displays:

```
> runRbCli.bat GetOrganizationDivisions

Usage: GetOrganizationDivisions -server <String> [-user <String>] [-pwd
<String>] [-any <boolean>] [-computeResultCount <boolean>] [-filters-operator
<EQUALS|MATCHES|GREATER_THAN|LESS_THAN|DOES_NOT_EQUAL|DOES_NOT_MATCH>]
[-filters-property <String>] [-filters-value <String>] [-limit <int>] [-offset
<int>] [-sortOrder-ascending <boolean>] [-sortOrder-property <String>]
[-correlationId <String>] [-disableResponseCompression <boolean>]
[-authComputerId <UUID>] [-authId <UUID>] [-authKey <String>] [-authMfaCode
<Integer>] [-authType <USER|DEVICE|EMERGENCY>] [-debug] [-prettyJson]
[-responseField <String>] [-verbose]


server:  Required.  The DNS name of the server endpoint to communicate with.

user:  Optional.  When authenticating by user name and password, this is the
user's login name.

pwd:  Optional.  When authenticating by user name and password, this is the user's
password.

any:  Optional.  A value must be specified of type boolean.
DefaultBooleanValue{value=false}.  See API documentation at
docs.riskband.com/sdk/api/ws/riskband/api/rest/frmwrk/request/GetBeansRestRequ
est.html#isAny-- for more details.

computeResultCount:  Optional.  A value must be specified of type boolean.
DefaultBooleanValue{value=false}.  See API documentation at
docs.riskband.com/sdk/api/ws/riskband/api/rest/frmwrk/request/GetBeansRestRequ
est.html#isComputeResultCount-- for more details.
...
```

In the descriptions of the individual parameters, the *Required* or *Optional* designations can be very helpful in determining how to successfully make the call.

If, however, you make a call incorrectly the console output for the command can sometimes identify the problem. For example, if I make the DeleteGeoFence call and don't supply the UUID of the geofence, the results could look something like this:

```
> runRbCli.bat DeleteGeoFence -server dev3.riskband.ws -authId
1d4524e7-d81e-4729-a670-0aca0c39cbd1 -authKey
zMe2uPKVoT9V5uTj7UnZ1jKkwTara75+szKunvvJ/fk= -authType USER -id

Usage: DeleteGeoFence -server <String> [-user <String>] [-pwd <String>] -id
<UUID> [-disableResponseCompression <boolean>] [-authComputerId <UUID>] [-authId
<UUID>] [-authKey <String>] [-authMfaCode <Integer>] [-authType
<USER|DEVICE|EMERGENCY>] [-correlationId <String>] [-debug] [-prettyJson]
[-responseField <String>] [-verbose]


server:  Required.  The DNS name of the server endpoint to communicate with.

user:  Optional.  When authenticating by user name and password, this is the
user's login name.

pwd:  Optional.  When authenticating by user name and password, this is the user's
password.

id:  Required.  A value must be specified of type UUID.
References{value=interface ws.riskband.api.dao.organization.GeoFence}.  See API
documentation at
docs.riskband.com/sdk/api/ws/riskband/api/rest/request/DeleteGeoFence.html#get
Id-- for more details.
.
.
.
.

**********************************************************************
*  An exception occurred attempting to execute the specified command.  *
**********************************************************************

IllegalArgumentException: Malformed syntax: "-id" is NOT a unary option, so a
value must be specified for it (but no value was specified for it).
```

Breaking Commands into Multiple Lines

For readability on the command line, you can break up a long line of options using the `\` or `^` characters.

On Windows, you can press Enter after typing the ^ character, and the command shell will prompt you for more input that will be combined into a single command::

```
> runRbCli.bat CreateGeoFence ^
More? -server     dev3.riskband.ws ^
More? -authId     1d4524e7-d81e-4729-a670-0aca0c39cbd1 ^
More? -authKey    zMe2uPKVoT9V5uTj7UnZ1jKkwTara75+szKunvvJ/fk= ^
More? -authType   USER ^
More? -divisionId 7325fdfa-13ab-49cc-b78f-5198a04a1779 ^
More? -lat1       39.749065 ^
More? -lat2       39.746049 ^
More? -lat3       39.751864 ^
More? -lat4       39.75504 ^
More? -lon1       "-105.217103" ^
More? -lon2       "-105.223367" ^
More? -lon3       "-105.233152" ^
More? -lon4       "-105.224258" ^
More? -name       "Colorado School of Mines" ^
More? -priority       500 ^
More? -maxAltitudeInMeters  1828 ^
More? -minAltitudeInMeters  1524

{"correlationId":"6EA48-1-1","result":{"maxAltitudeInMeters":1828,"minAltitude
InMeters":1524,"lat2":39.746049,"lat1":39.749065,"lon1":-105.217103,"lon2":-10
5.223367,"lat3":39.751864,"lon4":-105.224258,"lon3":-105.233152,"lat4":39.7550
4,"divisionId":"7325fdfa-13ab-49cc-b78f-5198a04a1779","name":"Colorado School
of Mines","priority":500,"id":"0bd1650c-0b05-4fd4-8253-6215706f3819"}}
```

On Linux, you can press Enter after typing the \ character, and the bash shell will allow you to input more options on the same line:

```
./runRbCli.sh CreateGeoFence \
-server     dev3.riskband.ws \
-authId     1d4524e7-d81e-4729-a670-0aca0c39cbd1 \
-authKey    zMe2uPKVoT9V5uTj7UnZ1jKkwTara75+szKunvvJ/fk= \
-authType   USER \
-divisionId 7325fdfa-13ab-49cc-b78f-5198a04a1779 \
-lat1       39.749065 \
-lat2       39.746049 \
-lat3       39.751864 \
-lat4       39.75504 \
-lon1       "-105.217103" \
-lon2       "-105.223367" \
-lon3       "-105.233152" \
-lon4       "-105.224258" \
-name       "Colorado School of Mines" \
-priority       500 \
-maxAltitudeInMeters  1828 \
-minAltitudeInMeters  1524
```

Special Features of the CLI

In addition to providing access to all the RiskBand APIs, the CLI also provides the following special features:

- User name and password authentication
- Specific responseField display

About User Name and Password Authentication

When calling RiskBand APIs directly, you authenticate by providing the UUID of your user object and a hash of your password. This same mechanism also works with the Java CLI. For example, if the UUID for user ggreen were `5b0d51f8-fd73-4e94-80c3-4d6751cbd893`, and the password hash were `vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=`, then a CLI call to get users would look like this:

```
> runRbCli.bat GetUsers -server dev3.riskband.ws -authId
5b0d51f8-fd73-4e94-80c3-4d6751cbd893 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER
-prettyJson | findstr "name"
    "name" : "acme\\user3",
    "name" : "acme\\orvillewright786543",
    "name" : "acme\\user5",
    "name" : "acme\\ggreen",
    "name" : "acme\\eeclles",
    "name" : "acme\\pparker",
    "name" : "acme\\egray",
    "name" : "acme\\williston.smythe.rumfelt",
    "name" : "acme\\ddanielson",
    "name" : "acme\\cchesteron",
    "name" : "acme\\rsmith",
    "name" : "acme\\axel.andersen",
    "name" : "acme\\wwhite",
    "name" : "acme\\apier",
```

Most of examples of CLI commands in this document are provided in this style to provide consistency with other mechanisms for calling the APIs.

However, a convenient feature of the CLI is that you can also provide credentials in a “human readable” format. For example, if you wanted to call GetUsers as user ggreen with a password of *password123*, you could use this command:

```
W:\17109>runRbCli.bat GetUsers -server dev3.riskband.ws -user acme\apier -pwd
password123 -prettyJson  | findstr "name"
    "name" : "acme\\user3",
    "name" : "acme\\orvillewright786543",
    "name" : "acme\\user5",
    "name" : "acme\\ggreen",
    "name" : "acme\\eeclles",
    "name" : "acme\\pparker",
    "name" : "acme\\egray",
    "name" : "acme\\williston.smythe.rumfelt",
    "name" : "acme\\ddanielson",
    "name" : "acme\\cchesteron",
    "name" : "acme\\rsmith",
    "name" : "acme\\axel.andersen",
    "name" : "acme\\wwhite",
    "name" : "acme\\apier",
```

Behind the scenes the CLI converts the “human readable” user name and password to a UUID and password hash.

About the responseField display

Part of the work of using the RiskBand CLI is manipulating the data that comes back as the response. When the response is long, it can be difficult at times to identify, or parse out, the information you want. The *-responseField* option allows you to see just a single value in the output.

For example, if you wanted to get information about the Acme organization, you could make a call like this:

```
> runRbCli.bat GetOrganization -user acme\ggreen -pwd password123 -server
dev3.riskband.ws -filter-property name -filter-value acme -prettyJson
{
  "result" : {
    "name" : "Acme",
    "description" : null,
    "dateRegistered" : 1566233520906,
    "securityPolicyId" : "46de61fa-0079-4120-859a-c269a812ae32",
    "firmwarePolicyId" : "f8df340b-4909-439a-9d12-b27f8b74ca99",
    "googleApiKey" : "AIzaSyAHNhOJbywWKnPp7aMGuZUbU4dEcDWiEFI",
    "emergencyContactInformationId" : null,
    "emergencyResponsePolicyId" : "caee66b0-5f92-4f4f-99ca-cdb1ad4f3f5c",
    "geosFusionNotificationSecretKey" : null,
    "unassignedPowerManagementPolicyId" : "af282211-2802-4243-99d4-b597dd5f92ef",
    "assignedPowerManagementPolicyId" : "b209fbd3-eb5e-4e24-aef7-677dfe42a5fe",
    "maxEmergencyRetentionInDays" : 1095,
    "eulaAcceptanceRequired" : true,
    "geosFusionServerEndPoint" : null,
    "geosFusionServerSecretKey" : null,
    "defaultPassphraseDuress" : "adrift",
    "passwordSelfRecoveryPolicy" : "ALLOW_VIA_MULTI_FACTOR",
    "minEmergencyRetentionInDays" : null,
    "defaultPassphraseAllClear" : "awash",
    "geosFusionTenantId" : null,
    "userNameDomain" : "acme\\",
    "smsInviteMessage" : "Welcome to Acme: {USERNAME} {PASSWORD} {EMAIL} {ID}",
    "emailInviteMessage" : "Welcome to Acme.\n\n{EMAILSUFFIX}",
    "passwordPolicy" : "WEAK",
    "shortName" : "acme",
    "id" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94"
  },
  "correlationId" : "05811-1-1"
}
```

But if all you wanted to see was the email invite message being used by the organization, you could specify *result-emailInviteMessage* as the responseField value. Note that the *result-* prefix is required because emailInviteMessage is part of the result object.

```
> runRbCli.bat GetOrganization -user acme\ggreen -pwd password123 -server
dev3.riskband.ws -filter-property name -filter-value acme
-responseField result-emailInviteMessage

Welcome to Acme.

{EMAILSUFFIX}
```

If, for example, you want to see just the correlationId, you would not use the *result-* prefix because that ID is not part of the result object.

```
> runRbCli.bat GetOrganization -user acme\ggreen -pwd password123 -server
dev3.riskband.ws -filter-property name -filter-value acme
-responseField correlationId

1D6A8-1-1
```

For CLI commands that return multiple objects, the value specified by responseField in the first object returned is displayed. For example, the call GetUsers displays multiple user objects. Specifying name as the responseField displays only the first object.

Note: You will want to inspect the format of the non-filtered API response to see the prefix needed for the -responseField option. For example, for an API like GetOrganization, that returns only one object, you will need to use the *result-* prefix. For an API that returns multiple objects, like GetUsers, you will need to use the *results-* prefix.

```
> runRbCli.bat GetUsers -server dev3.riskband.ws -user acme\apier -pwd password123
-prettyJson  | findstr "name"

    "name" : "acme\\bbrown",
    "name" : "acme\\user4",
    "name" : "acme\\7",
    "name" : "acme\\bblack",
    "name" : "acme\\user3",
    "name" : "acme\\username",
    "name" : "acme\\orvillewright786543",
    "name" : "acme\\user5",
    "name" : "acme\\ggreen",
    "name" : "acme\\eeclles",
    "name" : "acme\\username6",
    "name" : "acme\\user2",
    "name" : "acme\\pparker",
    "name" : "acme\\username4",
    "name" : "acme\\egray",
    "name" : "acme\\username1",
    "name" : "acme\\williston.smythe.rumfelt",
    "name" : "acme\\ddanielson",
    "name" : "acme\\username2",
    "name" : "acme\\cchesteron",
    "name" : "acme\\rsmith",
    "name" : "acme\\username3",
    "name" : "acme\\axel.andersen",
    "name" : "acme\\wwhite",
    "name" : "acme\\apier",
    "name" : "acme\\username5",
    "name" : "acme\\jdoe",
    "name" : "acme\\user1",

> runRbCli.bat GetUsers -server dev3.riskband.ws -user acme\apier -pwd password123
-responseField results-name

acme\bbrown
```

Unary Options in the CLI

Along with the parameters required to make an API call, the CLI provides the following options, or flags, that can be used on all CLI calls:

- -debug
- -prettyJson
- -verbose

About the `-debug` Flag

The `-debug` flag provides considerable information about what is being sent and received during a CLI call. This flag can be especially useful for determining the correct JSON format for API request. An example of how a CLI command might look using the *-debug* option is shown below. The JSON data being sent to the server is highlighted in blue.

```
W:\17109>runRbCli.bat LoginByUser -server dev3.riskband.ws -name acme\apier
-passwordHash vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -prettyJson -debug

INFO Aug 20 15:13:03,387 [main] | JVM initialized.  Runtime information is as
follows...
  User Name: wtaylor
  User Language: en
  User Timezone: America/Denver
  Operating System: Windows 10
.
.
.
INFO Aug 20 15:13:03,542 [main] | 5 command line options parsed:
  debug
  name | acme\apier
  passwordHash | zMe2uPKVoT9V5uTj7UnZ1jKkwTara75+szKunvvJ/fk=
  prettyJson
  server | dev3.riskband.ws
  (Main.<init>:176)
.
.
.
INFO Aug 20 15:13:03,693 [main] | RBClient@1-ReqExec making request
LoginByUser#B95A9-1-1 upon https://dev3.riskband.ws...:
{
  "name" : "acme\\apier",
  "computerId" : null,
  "mfaCode" : null,
  "passwordHash" : "<concealed>",
  "correlationId" : "B95A9-1-1",
  "authorization" : null,
  "passwordReset" : null,
  "disableResponseCompression" : false
}  (RiskBandClientSupportImpl$RequestExecutor.executeRequestInternal:367)
INFO Aug 20 15:13:03,693 [main] | RBClient@1-ReqExec making request
LoginByUser#B95A9-1-1 upon https://dev3.riskband.ws......
(RiskBandClientSupportImpl.logToBothWireAndStdLogs:633)
.
.
.
LoginByUser#B95A9-1-1 [842 ms].
(RiskBandClientSupportImpl$RequestExecutor.run:293)
INFO Aug 20 15:13:04,397 [main] | RBClient@1-ReqExec got response 200 for
LoginByUser#B95A9-1-1 [842 ms]:
{
  "name" : "acme\\apier",
  "correlationId" : "B95A9-1-1",
  "mustChangePassword" : false,
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber" : 17109,
  "eulaId" : null,
  "id" : "dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6"
}  (RiskBandClientSupportImpl$RequestExecutor.run:296)
INFO Aug 20 15:13:04,398 [main] | Executed LoginByUserImpl in 851 ms.
(CommandExecutor.execute:63)
{
  "name" : "acme\\apier",
  "correlationId" : "B95A9-1-1",
  "mustChangePassword" : false,
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber" : 17109,
  "eulaId" : null,
  "id" : "dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6"
}
```

About the -prettyJson flag

RiskBand APIs return unformatted JSON structures, and the CLI commands display the response as a single line. In some cases, especially when the response is large, it can be hard to see the structure of the response, and specifying *-prettyJson* can clarify the content.

Additionally, the *-prettyJson* flag can facilitate the use of filters to limit how much text is displayed. For example, if you use a filter like findstr without *-prettyJson*, then the entire response is returned because the response is all one line. However, with *-prettyJson* the results can be filtered in more useful way.

```
> runRbCli.bat GetOrganizationDivisions -server dev3.riskband.ws -user acme/apier
-pwd passowrd123 | findstr name

{"results":[{"name":"Div3","description":null,"securityPolicyId":"67291a77-dd7f-4
8c3-8edc-6566c80c253d","firmwarePolicyId":null,"organizationId":"dbd1b8a6-9da4-44
1d-bc6c-60adcf1a3a94","emergencyContactInformationId":null,"emergencyResponsePoli
cyId":null,"allowGroupsToUseDivisionDevices":true,"assignedPowerManagementPolicyI
d":null,"unassignedPowerManagementPolicyId":"742d435b-e827-493b-bde7-468380381477
","allowSelfAssignment":false,"shortName":"divis-3","id":"3354ef17-8935-4174-bde3
-789a8a4f5d23"},{"name":"Div2","description":null,"securityPolicyId":null,"firmwa
rePolicyId":null,"organizationId":"dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94","emergen
cyContactInformationId":null,"emergencyResponsePolicyId":null,"allowGroupsToUseDi
visionDevices":false,"assignedPowerManagementPolicyId":null,"unassignedPowerManag
ementPolicyId":null,"allowSelfAssignment":false,"shortName":"div2","id":"5aad84aa
-3496-467d-ba36-aa2efbf6b815"},{"name":"Div1","description":null,"securityPolicyI
d":null,"firmwarePolicyId":null,"organizationId":"dbd1b8a6-9da4-441d-bc6c-60adcf1
a3a94","emergencyContactInformationId":null,"emergencyResponsePolicyId":null,"all
owGroupsToUseDivisionDevices":true,"assignedPowerManagementPolicyId":null,"unassi
gnedPowerManagementPolicyId":null,"allowSelfAssignment":false,"shortName":"div1",
"id":"96f89df3-647e-482c-99b6-74a702fdd283"}],"correlationId":"A69EF-1-1","result
Count":null}



> runRbCli.bat GetOrganizationDivisions -server dev3.riskband.ws -user acme/apier
-pwd password123 -prettyJson | findstr name

    "name" : "Div3",
    "name" : "Div2",
    "name" : "Div1",
```

About the -verbose flag

If the results of a successful API call are confusing or unexpected, you can add the *-verbose* flag to get some direction on understanding the response. The *-verbose* flag will display teh name of the response definition that is being returned. You can then refer to this definition in the JavaDocs documentation.

```
> runRbCli.bat GetOrganizationDivision -server dev3.riskband.ws -user acme\apier
-pwd password123 -filter-property name -filter-value Div1 -prettyJson -verbose

##############################
ws.riskband.api.rest.response.GetOrganizationDivisionResponse
##############################

{
  "result" : {
    "name" : "Div1",
    "description" : null,
    "securityPolicyId" : null,
    "firmwarePolicyId" : null,
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "emergencyContactInformationId" : null,
    "emergencyResponsePolicyId" : null,
    "shortName" : "div1",
    "assignedPowerManagementPolicyId" : null,
    "unassignedPowerManagementPolicyId" : null,
    "allowGroupsToUseDivisionDevices" : true,
    "allowSelfAssignment" : false,
    "id" : "96f89df3-647e-482c-99b6-74a702fdd283"
  },
  "correlationId" : "BF59F-1-1"
}
```

Information about the API response for this command can be found at

https://docs.riskband.com/sdk/api/ws/riskband/api/rest/response/GetOrganizationDivisionResponse.html

```
> runRbCli.bat LoginByUser -server dev3.riskband.ws -name root -passwordHash
epssEzyyw9RYHu0uGT2zzurQr0mL6RqQWz3anOYIHbk= -prettyJson -verbose

##############################
ws.riskband.api.rest.response.LoginByUserResponse
##############################

{
  "name" : "root",
  "correlationId" : "16BA4-1-1",
  "organizationId" : null,
  "mustChangePassword" : false,
  "latestGuiBuildNumber" : 17109,
  "eulaId" : null,
  "id" : "024cc9f1-1975-4083-af83-6a00e76ed942"
}
```

Information about the API response for this command can be found at

https://docs.riskband.com/sdk/api/ws/riskband/api/rest/response/LoginByUserResponse.html

About Modify Commands

Generally, modify commands in the RiskBand API work in two ways:

- Like a create command — where you specify all the attributes you want for an existing object, and the object is essentially re-created—but keeps the original UUID. If you do not specify the *propertiesToModify* flag on the command line, or you set it to NULL, only the attributes explicitly specified in the call will be set. That is, everything not specified will be set to NULL.
- Like a selective modify command — where you specify just the optional attributes you want to modify using the *propertiesToModify* flag

RiskBand recommends using the *propertiesToModify* flag when modifying objects.

In the following examples, the ModifyUser API is used to modify just two attributes for user acme\cchesteron:

- citizenship
- all clear passphrase

Example of ModifyUser using the CLI

Make the GetUser call to see the current JSON structure of acme\cchesteron user object and to get the UUID:

```
> runRbCli.bat GetUser -server dev3.riskband.ws -authId
dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -filter-property name
-filter-value acme\cchesteron -prettyJson
{
  "result" : {
    "name" : "acme\\cchesteron",
    "passwordHash" : "<concealed>",
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "dateLastActivity" : null,
    "geosFusionUserId" : null,
    "mfaCurrentCode" : 0,
    "mfaCurrentDate" : 0,
    "passwordResetExpirationDate" : null,
    "description" : null,
    "emailAddress" : null,
    "suspended" : false,
    "firstName" : "Cogsworth",
    "lastName" : "Chesterton",
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "groupId" : null,
    "metadata" : null,
    "phoneNumber" : null,
    "restrictAccessToKnownComputers" : false,
    "passwordResetEmailCode" : null,
    "passwordResetPhoneCode" : null,
    "emergencyContactInformationId" : "e7a2a6c3-c037-43bb-b10d-132dd0e52820",
    "mfaTimeoutInMins" : null,
    "mustChangePassword" : true,
    "dateRegistered" : 1566234071699,
    "passphraseDuress" : null,
    "lifeSavingInformation" : null,
    "primaryPhotoId" : null,
    "restrictAccessToIps" : null,
    "passphraseAllClear" : "flotsam",
    "phoneVerified" : true,
    "dateOfBirth" : null,
    "googleApiKey" : null,
    "citizenship" : null,
    "gender" : null,
    "emailVerified" : true,
    "id" : "b65cda2e-0926-4b36-84e8-f5b1eacbeb13"
  },
  "correlationId" : "01BAC-1-1",
  "incompleteRequiredCourses" : [ ],
  "shortUserName" : null,
  "eulaId" : "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "eulaUrl" : "https://dev3.riskband.ws/legal.pdf"
}
```

Then make the ModifyUser call to change citizenship from null to USA and to change the all clear passphrase from flotsam to jetsam.

```
C:\Users\waynet\Documents\_wip\Downloads\CLI\17109>runRbCli.bat ModifyUser
-server dev3.riskband.ws -authId dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -name acme\cchesterton
-firstName Cogsworth -lastName Chesterton -id b65cda2e-0926-4b36-84e8-f5b1eacbeb13
-citizenship "USA" -passphraseAllClear "jetsam" -propertiesToModify
{citizenship}{passphraseAllClear}  -prettyJson
{
  "result" : {
    "name" : "acme\\cchesteron",
    "passwordHash" : "<concealed>",
    "restrictAccessToKnownComputers" : false,
    "lifeSavingInformation" : null,
    "passphraseAllClear" : "jetsam",
    "passphraseDuress" : null,
    "restrictAccessToIps" : null,
    "mustChangePassword" : true,
    "mfaTimeoutInMins" : null,
    "emergencyContactInformationId" : "e7a2a6c3-c037-43bb-b10d-132dd0e52820",
    "primaryPhotoId" : null,
    "suspended" : false,
    "description" : null,
    "phoneVerified" : true,
    "emailVerified" : true,
    "dateOfBirth" : null,
    "gender" : null,
    "citizenship" : "USA",
    "metadata" : null,
    "groupId" : null,
    "lastName" : "Chesterton",
    "firstName" : "Cogsworth",
    "emailAddress" : null,
    "phoneNumber" : null,
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "googleApiKey" : null,
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "dateLastActivity" : null,
    "geosFusionUserId" : null,
    "mfaCurrentCode" : 0,
    "mfaCurrentDate" : 0,
    "passwordResetExpirationDate" : null,
    "dateRegistered" : 1566234071699,
    "passwordResetEmailCode" : null,
    "passwordResetPhoneCode" : null,
    "id" : "b65cda2e-0926-4b36-84e8-f5b1eacbeb13"
  },
  "correlationId" : "1DCD5-1-1",
  "incompleteRequiredCourses" : [ ],
  "eulaUrl" : null,
  "eulaId" : null,
  "shortUserName" : null
}
```

5 More About the RiskBand Java SDK

The RiskBand Java SDK provides methods for making API calls to the RiskBand ARIES Manager. It eliminates the need to create JSON request structures directly and simplifies the processing of the JSON responses that come back from the server. It also provides better security than calling the APIs directly, and it provides full compile-time type safety. That is, the SDK ensures that you get bacl the correct response type for every request type.

In order to use the Java SDK you will need to have the following software installed on your computer:

- A Java Development Kit (JDK) version 1.7 or later
- The jar files contained in `RiskBand-Java-SDK-••••••.zip` file

The example code in this document was created using Eclipse, but it is not required.

Downloading and Installing the Java SDK

To download and install the RiskBand Java SDK:

1. Download `RiskBand-Java-SDK-•••••.zip` from https://downloads.riskband.com/index.html.
2. Unzip the downloaded file and place the contents in the desired directory or folder.

   The zip file contains both RiskBand-created jar files as well as third-party jar files. While not required, RiskBand recommends placing the files in the `lib` directory of your project or programming environment.
3. Add the directory or directories containing the jar files to the CLASSPATH of your programming environment. Or, add the directory or directories to the build path of your IDE.

Your Java programs should now be able to resolve and link any RiskBand API classes that are imported into your Java programs.

About JavaDocs for the RiskBand Java SDK

The JavaDocs for the SDK can be found at https://docs.riskband.com/sdk/api/ws/riskband/api/rest/request/package-summary.html

JavaDocs are updated on the server with every release of the RiskBand ARIES Manager application.

About Modify Commands

Generally, modify commands in the RiskBand API work in two ways:

- Like a create command — where you specify all the attributes you want for an existing object, and the object is essentially re-created but keeps the original UUID. If you leave the *propertiesToModify* string blank or set it to NULL, only the attributes explicitly specified in the call will be set. That is, everything not specified will be set to NULL.
- Like a selective modify command — where you specify just the optional attributes you want to modify using the *propertiesToModify* string array.

RiskBand recommends populating the *propertiesToModify* string array when modifying objects.

In the following examples, the *ModifyUser* API is used to modify just two attributes for user acme\cchesteron:

- citizenship
- all clear passphrase

Example of ModifyUser using the Java SDK

The first thing you need to do is instantiate a RiskBand client object. Provide credentials for the user you want to log in as.

```
    final RiskBandClient client = new RiskBandClientImpl(
          "https://dev3.riskband.ws",
            "acme\\apier",
            "password123" );
```

Then, in order to modify a user, you will need the UUID of the user being modified as well as the user name, first name, and last name of the user. These can be obtained by calling either *GetUsers* or *GetUser*. With *GetUsers* you get all the users, and you can pull the information out for the just the user you're interested in. With *GetUser* you provide a filter that specifies the name of the user whose information you are interested in and the call returns information for just that user.

Examples of both calls are provided in the sample code below.

```
// Get users
        System.out.println("Getting User in Organization");
        final GetUsersResponse orgUsers = client.getUsers(
            (GetUsers) BeanFactory.newBean(GetUsers.class)
            .setComputeResultCount(true));

// Get user. Create a filter for specific user
        final GetBeanFilter filter = BeanFactory.newBean(GetBeanFilter.class)
            .setProperty(NameObservable.NAME)
            .setValue("acme\\cchesteron");

        final GetUserResponse oneUser = client.getUser(
            (GetUser) BeanFactory.newBean(GetUser.class)
            .setFilter((filter)));
        UUID userId = oneUser.getResult().getId();
```

After successfully getting the user information, you can make the *ModifyUser* call. In the code sample below, the value for citizenship is changed from null to USA, and the value of the all clear passphrase is changed from flotsam to jetsam.

```
// Modify User
        String[] listOfPropertiesToModify = {
            "citizenship",
            "passphraseAllClear"
        };

        client.modifyUser(
            (ModifyUser) BeanFactory.newBean(ModifyUser.class)
            .setName(oneUser.getResult().getName())
            .setFirstName(oneUser.getResult().getFirstName())
            .setLastName(oneUser.getResult().getLastName())
            .setCitizenship("USA")
            .setPassphraseAllClear("jetsam")
            .setPropertiesToModify(listOfPropertiesToModify)
            .setId(userId));
```

Full Listing of ModifyUserExample.java

```
/*******************************************************************************
 *
 * Copyright C 2019, Whereable Technologies LLC and/or its affiliates.
 * All rights reserved.
 *
 ******************************************************************************/

import java.util.UUID;

import ws.riskband.api.rest.request.GetUser;
import ws.riskband.api.rest.request.GetUsers;
import ws.riskband.api.rest.request.ModifyUser;

import ws.riskband.api.rest.response.GetUserResponse;
import ws.riskband.api.rest.response.GetUsersResponse;

import ws.riskband.api.rest.frmwrk.request.GetBeanFilter;

import ws.riskband.api.dao.shared.NameObservable;

import ws.riskband.sdk.rb.client.RiskBandClient;
import ws.riskband.sdk.rb.client.RiskBandClientImpl;
import ws.riskband.util.bean.BeanFactory;

public class ModifyUserExample {
     public static void main(String[] args) {

org.apache.log4j.Logger.getRootLogger().setLevel(org.apache.log4j.Level.OFF);

// Instantiate instance of RiskBandClient
         final RiskBandClient client = new RiskBandClientImpl(
                 "https://dev3.riskband.ws",
                 "acme\\apier",
                 "password123" );

// Get users
            System.out.println("Getting User in Organization");
            final GetUsersResponse orgUsers = client.getUsers(
                    (GetUsers) BeanFactory.newBean(GetUsers.class)
                    .setComputeResultCount(true));
           System.out.print("Users Count: " + orgUsers.getResultCount() + "\n   "
);
            for ( int i = 0; i < orgUsers.getResultCount(); ++i ) {
              System.out.print("(" + orgUsers.getResults()[i].getId() +")" );
             System.out.print(": " + orgUsers.getResults()[i].getName() + "\n   "
);
            }

// Get user
            System.out.print("\nGetting a specific user by name \n   ");

            // Create filter for specific user
           final GetBeanFilter filter = BeanFactory.newBean( GetBeanFilter.class )
                    .setProperty( NameObservable.NAME )
                    .setValue( "acme\\cchesteron" );

            final GetUserResponse oneUser = client.getUser(
                    (GetUser) BeanFactory.newBean(GetUser.class)
                    .setFilter((filter)) );
            UUID userId = oneUser.getResult().getId();

            System.out.print(oneUser.getResult().getName() );
            System.out.print("  (" + userId + ")\n" );
            System.out.print("   Citizenship: ");
```

```
            System.out.print(oneUser.getResult().getCitizenship() );
            System.out.print("\n   All clear passphrase: ");
            System.out.print(oneUser.getResult().getPassphraseAllClear() );

// Modify User
            String[] listOfPropertiesToModify = {"citizenship",
                                                 "passphraseAllClear"};

            client.modifyUser(
                    (ModifyUser) BeanFactory.newBean(ModifyUser.class)
                    .setName( oneUser.getResult().getName() )
                    .setFirstName( oneUser.getResult().getFirstName() )
                    .setLastName( oneUser.getResult().getLastName() )
                    .setCitizenship("USA")
                    .setPassphraseAllClear("jetsam")
                    .setPropertiesToModify(listOfPropertiesToModify)
                    .setId(userId) );

// Get user again to see modification
            System.out.print("\nVerifying modification \n   ");
            final GetUserResponse modUser = client.getUser(
                    (GetUser) BeanFactory.newBean(GetUser.class)
                    .setFilter((filter)) );

            System.out.print(modUser.getResult().getName() );
            System.out.print("  (" + modUser.getResult().getId() + ")\n   " );
          System.out.print("Citizenship: " + modUser.getResult().getCitizenship()
);
            System.out.print("\n   All clear passphrase: ");
            System.out.print(modUser.getResult().getPassphraseAllClear() );

    }
}
```

ModifyUserExample.java Output

```
Getting User in Organization
Users Count: 28
   (15424399-9b60-45b6-82c4-1896bd9abc36): acme\egray
   (205f1063-a799-4fc5-80f6-12079a1eb398): acme\username2
   (34c9eae7-5672-462b-a322-acf6a4363056): acme\pparker
   (36f5de75-25af-4a1c-b634-afcf98eca5c3): acme\ddanielson
   (38b0b401-ba97-440d-8e80-e15cd978ce4e): acme\user5
   (3defb1d0-5978-4e8b-bbcc-175cb9a8782a): acme\username5
   (46f3fed5-0351-48cd-b787-8838b25561f5): acme\orvillewright786543
   (4dd19219-8c16-4625-89df-b4c578304035): acme\bblack
   (53281193-9b06-4845-9bd2-1d0675a59fe5): acme\eeclles
   (566774a4-e65e-45f6-ae88-954e0e948ec7): acme\ggreen
   (639f5170-8ba7-4d40-aed5-04d185a420f2): acme\rsmith
   (68f6b37a-eb0b-4631-b8e6-9c0cbede7dfd): acme\user2
   (6ab96eb1-1f02-47a3-b398-0f0915546be1): acme\username4
   (6fbdf3c6-c21d-4dab-a072-fd617f943de8): acme\user1
   (7cdbd27b-0b94-46b0-b933-1f16c4fdacb0): acme\axel.andersen
   (94aa6bf0-0937-4291-9097-8af1b5708131): acme\wwhite
   (981e73e0-1f43-4608-a916-4a5fd14c280c): acme\user3
   (aa1e1d5c-2682-4a36-98c5-631bfbad9103): acme\apier
   (b258f2c1-4d6e-4221-b1ff-5642ad851077): acme\username6
   (bc7a33c7-b161-456f-8f69-b858c278365f): acme\user4
   (c447fa82-4032-4f1d-9f81-5dbc1cfeea7b): acme\williston.smythe.rumfelt
   (db005343-7f26-4a9a-99d0-20fa8b968491): acme\username3
   (dbd3a6d1-58b0-40e3-b2af-2fbc989b91e6): acme\admin
   (df24aa80-515e-4e27-986a-3ba35f626bcb): acme\username1
   (eb8d581f-b717-4ddb-8091-e59e144237bd): acme\bbrown
   (f31cc831-4bc8-45f1-8ef6-b20f862fcb45): acme\cchesteron
   (f4caf14a-169f-4c74-8299-0d6f8ad75867): acme\username
   (fca757c9-e936-4608-8fb6-ee62197cd141): acme\jdoe

Getting a specific user by name
   acme\cchesteron  (f31cc831-4bc8-45f1-8ef6-b20f862fcb45)
   Citizenship: null   All clear passphrase: flotsam
Verifying modification
   acme\cchesteron  (f31cc831-4bc8-45f1-8ef6-b20f862fcb45)
   Citizenship: USA
   All clear passphrase: jetsam
```

6 Example: Creating a Geofence using a Web Browser

Generally, issuing API commands from the web browser is most useful for single commands. Using the web browser to accomplish tasks that require multiple steps can be awkward to enter and difficult to maintain. However, in the interest of illustrating this method of calling the API, this section provides an example of how to use the RiskBand ARIES Manager RESTful API from a web browser to create a geofence.

Before running these examples, the following configuration needs to be performed on the RiskBand ARIES Manager:

- Organization Created — in this example the organization is Acme
- User Created in Organization — in this example, the user is Apier
- Division Created in Organization — because Geofences can only be created inside divisions.
- User Apier has “Manage GeoFence” permissions
- No multi-factor authentication is enabled for the user

To create a geofence using the RiskBand ARIES Manager API, you will need to make the following calls:

- Calling LoginByUser
- Calling GetOrganizationDivisions
- Calling CreateGeoFence
- Calling GetGeoFences

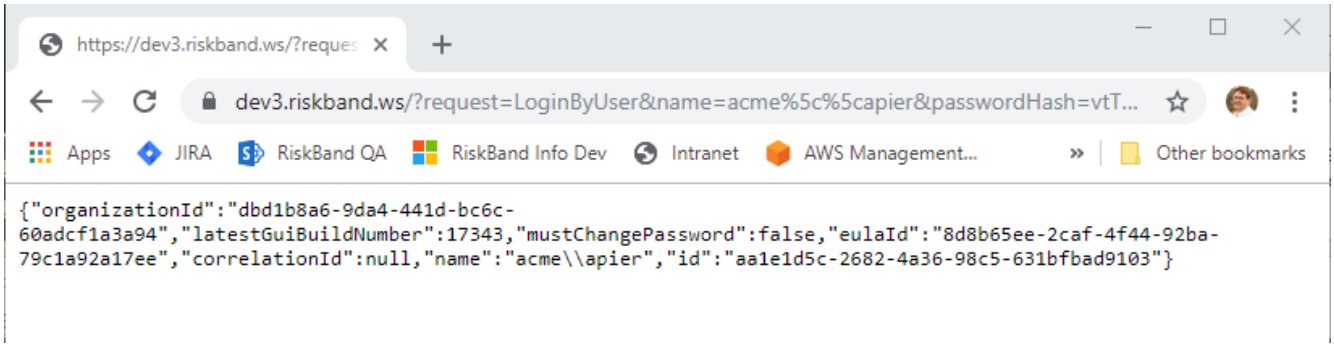
Note: Authenticating an API requires a hash of the user password, and if the password hash contains special characters, you will need to change those characters to URL percent encoding. For example, a password hash of `3tYGOAeUHge5QTkDcUJBGH5WJESIuONyVav9I0XkZlQ=` would be entered in the address bar of the browser as `3tYGOAeUHge5QTkDcUJBGH5WJESIuONyVav9I0XkZlQ%3D`.

Calling LoginByUser

Make the LoginByUser call by entering the following command in the address bar of your browser:

```
https://dev3.riskband.ws/?request=LoginByUser&name=acme%5c%5capier&passwordHas
h=vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
```

Figure 9: LoginByUser using Browser Address Bar



The formatted response from the server looks like this:

```
{
  "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber": 17343,
  "mustChangePassword": false,
  "eulaId": "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "correlationId": null,
  "name": "acme\\apier",
  "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

The values you will need from the response to make the GetOrganizationDivisions call are

- the UUID of the user you will want to authenticate with
- the UUID of the organization where the division resides. In this example, the assumption is the division is in the same organization as the user that you authenticate with.:

```
"organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94"
"id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
```

Calling GetOrganizationDivisions

Make the GetOrganizationDivisions by entering the following command in the address bar of your browser:

```
https://dev3.riskband.ws/?request=GetOrganizationDivisions&authorization-entit
yType=USER&authorization-id=aa1e1d5c-2682-4a36-98c5-631bfbad9103&authorization
-passwordHash=vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
```

The formatted response from the server looks like this:


```
{
  "correlationId": null,
  "resultCount": null,
  "results": [
    {
      "shortName": "divis-3",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": "67291a77-dd7f-48c3-8edc-6566c80c253d",
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": "742d435b-e827-493b-bde7-468380...",
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div3",
      "id": "3354ef17-8935-4174-bde3-789a8a4f5d23"
    },
    {
      "shortName": "div2",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": false,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div2",
      "id": "5aad84aa-3496-467d-ba36-aa2efbf6b815"
    },
    {
      "shortName": "div1",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div1",
      "id": "96f89df3-647e-482c-99b6-74a702fdd283"
    }
  ]
}
```

The value you will need for the CreateGeoFence call is the UUID of the division where you want to create the geofence. In this case it will be the UUID of “Div1”:

```
"name": "Div1",
"id": "96f89df3-647e-482c-99b6-74a702fdd283"
```

Calling CreateGeoFence

Make the CreateGeoFence by entering the following command in the address bar of your browser:

```
https://dev3.riskband.ws/?request=CreateGeoFence&authorization-entityType=USER
&authorization-id=aa1e1d5c-2682-4a36-98c5-631bfbad9103&authorization-passwordH
ash=vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=&divisionId=96f89df3-647e-482c
-99b6-74a702fdd283&name=Colorado%20School%20of%20Mines&priority=501&minAltitud
eInMeters=1524&maxAltitudeInMeters=1828&lat1=39.749065&lat2=39.746049&lat3=39.
751864&lat4=39.75504&lon1=%2D105.217103&lon2=%2D105.223367&lon3=%2D105.233152&
lon4=%2D105.224258
```

The formatted response from the server looks like this:

```
{
  "correlationId": null,
  "result": {
    "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
    "minAltitudeInMeters": 1524,
    "maxAltitudeInMeters": 1828,
    "lat1": 39.749065,
    "lon1": -105.217103,
    "lat2": 39.746049,
    "lon2": -105.223367,
    "lat3": 39.751864,
    "lon3": -105.233152,
    "lat4": 39.75504,
    "lon4": -105.224258,
    "name": "Colorado School of Mines",
    "priority": 501,
    "id": "e29b7738-55f8-4fb7-9e8d-4c272508a44e"
  }
}
```

Calling GetGeoFences

You now can verify that the geofence exists in the system, by calling GetGeoFences. Make the GetGeoFences by entering the following command in the address bar of your browser:

```
https://dev3.riskband.ws/?request=GetGeoFences&authorization-entityType=USER&a
uthorization-id=1d4524e7-d81e-4729-a670-0aca0c39cbd1&authorization-passwordHas
h=zMe2uPKVoT9V5uTj7UnZ1jKkwTara75+szKunvvJ/fk=
```

The formatted response form the server looks like this:


```
{
  "correlationId": null,
  "resultCount": null,
  "results": [
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.254,
      "lon1": -85.9,
      "lat2": 30.214,
      "lon2": -85.9,
      "lat3": 30.214,
      "lon3": -85.8,
      "lat4": 30.254,
      "lon4": -85.8,
      "name": "Botheration Bayou",
      "priority": 999,
      "id": "5d2fb718-9553-481b-b8be-11ecc82d0262"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.214,
      "lon1": -85.9,
      "lat2": 30.174,
      "lon2": -85.9,
      "lat3": 30.174,
      "lon3": -85.8,
      "lat4": 30.214,
      "lon4": -85.8,
      "name": "Panama City Beach - Florida",
      "priority": 1000,
      "id": "61919772-dbff-4c4c-9d64-c9ab8d1247f7"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": 1524,
      "maxAltitudeInMeters": 1828,
      "lat1": 39.749065,
      "lon1": -105.217103,
      "lat2": 39.746049,
      "lon2": -105.223367,
      "lat3": 39.751864,
      "lon3": -105.233152,
      "lat4": 39.75504,
      "lon4": -105.224258,
      "name": "Colorado School of Mines",
      "priority": 501,
      "id": "e29b7738-55f8-4fb7-9e8d-4c272508a44e"
    }
  ]
}
```

7 Example: Creating a Geofence using cURL

This section provides an example of how to call the RiskBand ARIES Manager RESTful API with cURL to create a geofence. The examples are shown with the JSON request contained in a file.

Before running these examples, the following configuration needs to be performed on the RiskBand ARIES Manager:

- Organization Created — in this example the organization is Acme
- User Created in Organization — in this example, the user is Apier
- Division Created in Organization — because Geofences can only be created inside divisions.
- User Apier has “Manage GeoFence” permissions
- No multi-factor authentication is enabled for the user

To create a geofence using the RiskBand ARIES Manager API, you will need to make the following calls:

- LoginByUser
- GetOrganizationDivisions
- CreateGeoFence
- GetGeoFences

Calling LoginByUser

Make the LoginByUser call using a cURL command similar to this:

```
curl -iv -X PUT -H "Content-Length: 279" -H "Content-Type: application/json" -d
@_loginByUser_acmeApier_279.json https://dev3.riskband.ws?LoginByUser=279U
```

The `_loginByUserRequest_acmeApier_279.json` file that is sent to the server looks like this:

```
{
  "authorization": null,
  "computerId": "00000000-0000-0000-0000-000000012345",
  "correlationId": "6BFAA-1-2",
  "disableResponseCompression": true,
  "mfaCode": null,
  "name": "acme\\apier",
  "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
  "passwordReset": null
}
```

If the call completes successfully, the response will look something like this (response formatting added)

```
HTTP/1.1 200 OK
Content-Language: en-US
Date: Thu, 05 Sep 2019 15:20:10 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 254U
x-riskband-response-type: ws.riskband.api.rest.response.LoginByUserResponse
Content-Length: 254
Connection: keep-alive

{
  "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber": 17343,
  "mustChangePassword": false,
  "eulaId": "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "correlationId": "6BFAA-1-2",
  "name": "acme\\apier",
  "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

The values you will need from the response to make the GetOrganizationDivisions call are

- the UUID of the user you will want to authenticate with

  ```
  "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
  ```

- the UUID of the organization where the division resides. In this example, the assumption is that the division is in the same organization as the user that you authenticated with

  ```
  "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94"
  ```

Calling GetOrganizationDivisions

Make the GetOrganizationDivisions call using a cURL command similar to this:

```
curl -iv -X PUT -H "Content-Length: 551" -H "Content-Type: application/json" -d
@_GetOrganizationDivisions_551.json
https://dev3.riskband.ws?GetOrganizationDivisions=551U
```

The `GetOrganizationDivisions_551.json` file that is sent to the server looks like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [
    {
      "property": "organizationId",
      "value": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "operator": "EQUALS"
    }
  ],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
```

If the call completes successfully, the response will look something like this (response formatting added):

```
HTTP/1.1 200 OK
Content-Language: en-US
Date: Thu, 05 Sep 2019 00:34:49 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 1389U
x-riskband-response-type:
ws.riskband.api.rest.response.GetOrganizationDivisionsResponse
Content-Length: 1389
Connection: keep-alive

{
  "correlationId": "RANDOM-0-002",
  "resultCount": 3,
  "results": [
    {
      "shortName": "divis-3",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": "67291a77-dd7f-48c3-8edc-6566c80c253d",
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
     "unassignedPowerManagementPolicyId": "742d435b-e827-493b-bde7-4683803...",
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div3",
      "id": "3354ef17-8935-4174-bde3-789a8a4f5d23"
    },
    {
      "shortName": "div2",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": false,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div2",
      "id": "5aad84aa-3496-467d-ba36-aa2efbf6b815"
    },
    {
      "shortName": "div1",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div1",
      "id": "96f89df3-647e-482c-99b6-74a702fdd283"
    }
  ]
}
```

The value you will need for the CreateGeoFence call is the UUID of the division where you want to create the geofence. In this case it will be the UUID of “Div1”:

```
"name": "Div1",
"id": "96f89df3-647e-482c-99b6-74a702fdd283"
```

Calling CreateGeoFence

Make the CreateGeoFence call using a cURL command similar to this:

```
curl -iv -X PUT -H "Content-Length: 671" -H "Content-Type: application/json" -d
@_CreateGeoFence_671.json  https://dev3.riskband.ws?CreateGeoFence=671U
```

The `_CreateGeoFence_671.json` file that is sent to the server looks like this:

```
{
   "authorization":{
      "computerId":"00000000-0000-0000-0000-000000012345",
      "entityType":"USER",
      "id":"aa1e1d5c-2682-4a36-98c5-631bfbad9103",
      "mfaCode":null,
      "passwordHash":"vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
   },
   "correlationId":"RANDOM-0-002",
   "disableResponseCompression":true,
   "divisionId":"96f89df3-647e-482c-99b6-74a702fdd283",
   "id":null,
   "lat1":39.749065,
   "lat2":39.746049,
   "lat3":39.751864,
   "lat4":39.75504,
   "lon1":-105.217103,
   "lon2":-105.223367,
   "lon3":-105.233152,
   "lon4":-105.224258,
   "maxAltitudeInMeters":1828,
   "minAltitudeInMeters":1524,
   "name":"Colorado School of Mines",
   "priority":500
}
```

If the call completes successfully, the response will look something like this (response formatting added):

```
HTTP/1.1 201 Created
Content-Language: en-US
Date: Thu, 05 Sep 2019 18:41:43 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 385U
x-riskband-response-type: ws.riskband.api.rest.response.GetGeoFenceResponse
Content-Length: 385
Connection: keep-alive

{
  "correlationId": "RANDOM-0-002",
  "result": {
    "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
    "minAltitudeInMeters": 1524,
    "maxAltitudeInMeters": 1828,
    "lat1": 39.749065,
    "lon1": -105.217103,
    "lat2": 39.746049,
    "lon2": -105.223367,
    "lat3": 39.751864,
    "lon3": -105.233152,
    "lat4": 39.75504,
    "lon4": -105.224258,
    "name": "Colorado School of Mines",
    "priority": 500,
    "id": "cef0b309-2822-4f1f-b6fe-27d294dfaafa"
  }
}
```

Calling GetGeoFences

You can verify that the geofence now exists in the system, by calling GetGeoFences. The call can be made using a cURL command similar to this:

```
curl -iv -X PUT -H "Content-Length: 425" -H "Content-Type: application/json" -d
@_GetGeoFences_425.json  https://dev3.riskband.ws?GetGeoFences=425U
```

The `_GetGeoFences_425.json` file that is sent to the server looks like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
```

If the call completes successfully, the response will look something like this (response formatting added):

```
HTTP/1.1 200 OK
Content-Language: en-US
Date: Thu, 05 Sep 2019 18:43:29 GMT
Server: Apache-Coyote/1.1
x-riskband-response-length: 1019U
x-riskband-response-type: ws.riskband.api.rest.response.GetGeoFencesResponse
Content-Length: 1019
Connection: keep-alive

{
  "correlationId": "RANDOM-0-002",
  "resultCount": 3,
  "results": [
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.254,
      "lon1": -85.9,
      "lat2": 30.214,
      "lon2": -85.9,
      "lat3": 30.214,
      "lon3": -85.8,
      "lat4": 30.254,
      "lon4": -85.8,
      "name": "Botheration Bayou",
      "priority": 999,
      "id": "5d2fb718-9553-481b-b8be-11ecc82d0262"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.214,
      "lon1": -85.9,
      "lat2": 30.174,
      "lon2": -85.9,
      "lat3": 30.174,
      "lon3": -85.8,
      "lat4": 30.214,
      "lon4": -85.8,
      "name": "Panama City Beach - Florida",
      "priority": 1000,
      "id": "61919772-dbff-4c4c-9d64-c9ab8d1247f7"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": 1524,
      "maxAltitudeInMeters": 1828,
      "lat1": 39.749065,
      "lon1": -105.217103,
      "lat2": 39.746049,
      "lon2": -105.223367,
      "lat3": 39.751864,
      "lon3": -105.233152,
      "lat4": 39.75504,
      "lon4": -105.224258,
      "name": "Colorado School of Mines",
      "priority": 500,
      "id": "cef0b309-2822-4f1f-b6fe-27d294dfaafa"
    }
  ]
}
```

8 Example: Creating a Geofence using PowerShell

This section provides an example of how to use the Invoke-RestMethod in Windows PowerShell to create a geofence.

Before running these examples, the following configuration needs to be performed on the RiskBand ARIES Manager:

- Organization Created — in this example the organization is Acme
- User Created in Organization — in this example, the user is Apier
- Division Created in Organization — because Geofences can only be created inside divisions.
- User Apier has “Manage GeoFence” permissions
- No multi-factor authentication is enabled for the user

To create a geofence using the RiskBand ARIES Manager API, you will need to make the following calls:

- LoginByUser
- GetOrganizationDivisions
- CreateGeoFence
- GetGeoFences

The PowerShell script used in this example is a single script; however, the sections for each call will be broken out and described separately below.

Calling LoginByUser

The JSON request for the LoginByUser call is created using an array in PowerShell:

```
$json = @"
{
  "authorization": null,
  "computerId": "00000000-0000-0000-0000-000000012345",
  "correlationId": "6BFAA-1-2",
  "disableResponseCompression": true,
  "mfaCode": null,
  "name": "acme\\apier",
  "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
  "passwordReset": null
}
"@
```

The length of the JSON request is then calculated and the HTTPS request is created. Then the Invoke-RestMethod is called with the HTTPS request.

```
$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?LoginByUser=" +$jsonLen +"U"
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json
```

If the call completes successfully, the response will look something like this (response formatting added)

```
{
    "organizationId":  "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "latestGuiBuildNumber":  17343,
    "mustChangePassword":  false,
    "eulaId":  "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
    "correlationId":  "6BFAA-1-2",
    "name":  "acme\\apier",
    "id":  "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

The values you will need from the response to make the GetOrganizationDivisions call are

- the UUID of the user you will want to authenticate with

  ```
  "id":  "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
  ```

- the UUID of the organization where the division resides. In this example, the assumption is that the division is in the same organization as the user that you authenticated with

  ```
  "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94"
  ```

These values are saved in variables:

```
$id = $response.id
$orgID = $response.organizationId
```

Calling GetOrganizationDivisions

The GetOrganizationDivisions request is created using values obtain from the previous call:

```
$json = @"
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [
    {
      "property": "organizationId",
      "value": "$orgID",
      "operator": "EQUALS"
    }
  ],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetOrganizationDivisions=" +$jsonLen +"U"
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json
```

The response comes back as an array with nested arrays.


```
{
  "correlationId": "RANDOM-0-002",
  "resultCount": 3,
  "results": [
    {
      "shortName": "divis-3",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": "67291a77-dd7f-48c3-8edc-6566c80c253d",
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": "742d435493b-bde...",
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div3",
      "id": "3354ef17-8935-4174-bde3-789a8a4f5d23"
    },
    {
      "shortName": "div2",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": false,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div2",
      "id": "5aad84aa-3496-467d-ba36-aa2efbf6b815"
    },
    {
      "shortName": "div1",
      "organizationId": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "securityPolicyId": null,
      "firmwarePolicyId": null,
      "allowGroupsToUseDivisionDevices": true,
      "allowSelfAssignment": false,
      "emergencyResponsePolicyId": null,
      "assignedPowerManagementPolicyId": null,
      "unassignedPowerManagementPolicyId": null,
      "emergencyContactInformationId": null,
      "description": null,
      "name": "Div1",
      "id": "96f89df3-647e-482c-99b6-74a702fdd283"
    }
  ]
}
```


When programmatically processing a nested array it is very useful to have the resultcount, which is the number of nested arrays returned. However, for simplicity, this example does process the arrays; it just uses the values in the second nested array. As nested array numbering starts with zero, the second array, div2, is identified with a [1].

```
$nestedArrayCount = $response.resultcount
$idFromSecondArray = $response.results.id[1]
```

The division UUID you will need for the CreateGeoFence call is

```
5aad84aa-3496-467d-ba36-aa2efbf6b815
```

Calling CreateGeoFence

The CreateGeoFence request is created using values obtain from the previous calls

```
$json = @"
{
   "authorization":{
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
    "correlationId":"RANDOM-0-002",
    "disableResponseCompression":true,
    "divisionId":"$idFromSecondArray",
    "id":null,
    "lat1":39.749065,
    "lat2":39.746049,
    "lat3":39.751864,
    "lat4":39.75504,
    "lon1":-105.217103,
    "lon2":-105.223367,
    "lon3":-105.233152,
    "lon4":-105.224258,
    "maxAltitudeInMeters":1828,
    "minAltitudeInMeters":1524,
    "name":"Colorado School of Mines",
    "priority":500
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?CreateGeoFence=" +$jsonLen +"U"
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json
```

If the call completes successfully, the response will look something like this (response formatting added):

```
{
  "correlationId": "RANDOM-0-002",
  "result": {
    "divisionId": "5aad84aa-3496-467d-ba36-aa2efbf6b815",
    "minAltitudeInMeters": 1524,
    "maxAltitudeInMeters": 1828,
    "lat1": 39.749065,
    "lon1": -105.217103,
    "lat2": 39.746049,
    "lon2": -105.223367,
    "lat3": 39.751864,
    "lon3": -105.233152,
    "lat4": 39.75504,
    "lon4": -105.224258,
    "name": "Colorado School of Mines",
    "priority": 500,
    "id": "279aa3ab-b6fc-45b0-8348-ff93976e91f5"
  }
}
```

Calling GetGeoFences

You can verify that the geofence now exists in the system, by calling GetGeoFences. The request is created using values obtain from the previous calls:

```
$json = @"
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetGeoFences=" +$jsonLen +"U"
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json
```

If the call completes successfully, the response will look something like this (response formatting added):


```
{
  "correlationId": "RANDOM-0-002",
  "resultCount": 3,
  "results": [
    {
      "divisionId": "5aad84aa-3496-467d-ba36-aa2efbf6b815",
      "minAltitudeInMeters": 1524,
      "maxAltitudeInMeters": 1828,
      "lat1": 39.749065,
      "lon1": -105.217103,
      "lat2": 39.746049,
      "lon2": -105.223367,
      "lat3": 39.751864,
      "lon3": -105.233152,
      "lat4": 39.75504,
      "lon4": -105.224258,
      "name": "Colorado School of Mines",
      "priority": 500,
      "id": "279aa3ab-b6fc-45b0-8348-ff93976e91f5"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.254,
      "lon1": -85.9,
      "lat2": 30.214,
      "lon2": -85.9,
      "lat3": 30.214,
      "lon3": -85.8,
      "lat4": 30.254,
      "lon4": -85.8,
      "name": "Botheration Bayou",
      "priority": 999,
      "id": "5d2fb718-9553-481b-b8be-11ecc82d0262"
    },
    {
      "divisionId": "96f89df3-647e-482c-99b6-74a702fdd283",
      "minAltitudeInMeters": -100,
      "maxAltitudeInMeters": 1000,
      "lat1": 30.214,
      "lon1": -85.9,
      "lat2": 30.174,
      "lon2": -85.9,
      "lat3": 30.174,
      "lon3": -85.8,
      "lat4": 30.214,
      "lon4": -85.8,
      "name": "Panama City Beach - Florida",
      "priority": 1000,
      "id": "61919772-dbff-4c4c-9d64-c9ab8d1247f7"
    }
  ]
}
```

Full Listing of CreateGeoFence PS Script

```
# LoginByUser
# As necessary, change valudes of "name" and "passwordHash"
$json = @"
{
  "authorization": null,
  "computerId": "00000000-0000-0000-0000-000000012345",
  "correlationId": "6BFAA-1-2",
  "disableResponseCompression": true,
  "mfaCode": null,
  "name": "acme\\apier",
  "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
  "passwordReset": null
}
"@

# Change the end point as necessary (e.g. perhaps use "https://us.riskband.com)
$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?LoginByUser=" +$jsonLen +"U"

Write-Host ("`n ***  Calling LoginByUser  ***")
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json

# Save off variables for upcoming calls
$id = $response.id
$orgID = $response.organizationId

Write-Host ("`n JSON Request:`n" + $json)
Write-Host ("`n JSON Response:`n" + ($response | ConvertTo-Json))
Write-Host ("`n OrganizationId where creating GeoFence: " + $orgID)

# GetOrganizationDivisions
$json = @"
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [
    {
      "property": "organizationId",
      "value": "$orgID",
      "operator": "EQUALS"
    }
  ],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetOrganizationDivisions=" +$jsonLen +"U"

Write-Host ("`n`n ***  Calling GetOrganizationDivisions  ***")
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json
```

```
# The response comes back as nested arrays. The value of resultcount can help with
progmatically indexing into the array
# In this case we know we want the value of "id" from the (zero-based) 2nd item
(i.e. [1] )
$nestedArrayCount = $response.resultcount
$idFromSecondArray = $response.results.id[1]


Write-Host ("`n JSON Request:`n" + $json)
Write-Host ("`n JSON Response:`n" + ($response | ConvertTo-Json))
Write-Host ("`n Number of Divisions (i.e. resultcount): " + $nestedArrayCount)
Write-Host (" Division ID where creating GeoFence:  " + $idFromSecondArray)



# CreateGeoFence
$json = @"
{
   "authorization":{
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
   "correlationId":"RANDOM-0-002",
   "disableResponseCompression":true,
   "divisionId":"$idFromSecondArray",
   "id":null,
   "lat1":39.749065,
   "lat2":39.746049,
   "lat3":39.751864,
   "lat4":39.75504,
   "lon1":-105.217103,
   "lon2":-105.223367,
   "lon3":-105.233152,
   "lon4":-105.224258,
   "maxAltitudeInMeters":1828,
   "minAltitudeInMeters":1524,
   "name":"Colorado School of Mines",
   "priority":500
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?CreateGeoFence=" +$jsonLen +"U"

Write-Host ("`n`n ***  Calling CreateGeoFence  ***")
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json

# We're saving the ID of the newly created GeoFence in case we need to modify it
$newGeoFenceId = $response.result.id

Write-Host ("`n JSON Request:`n" + $json)
Write-Host ("`n JSON Response:`n" + ($response | ConvertTo-Json))
Write-Host (" id of new GeoFence:  " + $newGeoFenceId)
```

```
# GetGeoFences
$json = @"
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "$id",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetGeoFences=" +$jsonLen +"U"

Write-Host ("`n`n ***  Calling GetGeoFences  ***")
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json

Write-Host ("`n JSON Request:`n" + $json)
Write-Host ("`n JSON Response:`n" + ($response | ConvertTo-Json))
Write-Host ("`n Number of GeoFences (i.e. resultcount): " + $nestedArrayCount)
```

Full Listing of CreateGeoFence PS Script Output

```
PS > CreateGeoFence_Invoke-RestMethod.ps1

 ***  Calling LoginByUser  ***

 JSON Request:
{
  "authorization": null,
  "computerId": "00000000-0000-0000-0000-000000012345",
  "correlationId": "6BFAA-1-2",
  "disableResponseCompression": true,
  "mfaCode": null,
  "name": "acme\\apier",
  "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
  "passwordReset": null
}

 JSON Response:
{
    "organizationId":  "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "latestGuiBuildNumber":  17343,
    "mustChangePassword":  false,
    "eulaId":  "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
    "correlationId":  "6BFAA-1-2",
    "name":  "acme\\apier",
    "id":  "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}

 OrganizationId where creating GeoFence: dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94


 ***  Calling GetOrganizationDivisions  ***

 JSON Request:
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [
    {
      "property": "organizationId",
      "value": "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
      "operator": "EQUALS"
    }
  ],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
```

```
 JSON Response:
{
    "correlationId":  "RANDOM-0-002",
    "resultCount":  3,
    "results":  [
                    {
                        "shortName":  "divis-3",
                        "organizationId":  "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
                      "securityPolicyId":  "67291a77-dd7f-48c3-8edc-6566c80c253d",
                        "firmwarePolicyId":  null,
                        "allowGroupsToUseDivisionDevices":  true,
                        "allowSelfAssignment":  false,
                        "emergencyResponsePolicyId":  null,
                        "assignedPowerManagementPolicyId":  null,
                        "unassignedPowerManagementPolicyId":
"742d435b-e827-493b-bde7-468380381477",
                        "emergencyContactInformationId":  null,
                        "description":  null,
                        "name":  "Div3",
                        "id":  "3354ef17-8935-4174-bde3-789a8a4f5d23"
                    },
                    {
                        "shortName":  "div2",
                        "organizationId":  "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
                        "securityPolicyId":  null,
                        "firmwarePolicyId":  null,
                        "allowGroupsToUseDivisionDevices":  false,
                        "allowSelfAssignment":  false,
                        "emergencyResponsePolicyId":  null,
                        "assignedPowerManagementPolicyId":  null,
                        "unassignedPowerManagementPolicyId":  null,
                        "emergencyContactInformationId":  null,
                        "description":  null,
                        "name":  "Div2",
                        "id":  "5aad84aa-3496-467d-ba36-aa2efbf6b815"
                    },
                    {
                        "shortName":  "div1",
                        "organizationId":  "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
                        "securityPolicyId":  null,
                        "firmwarePolicyId":  null,
                        "allowGroupsToUseDivisionDevices":  true,
                        "allowSelfAssignment":  false,
                        "emergencyResponsePolicyId":  null,
                        "assignedPowerManagementPolicyId":  null,
                        "unassignedPowerManagementPolicyId":  null,
                        "emergencyContactInformationId":  null,
                        "description":  null,
                        "name":  "Div1",
                        "id":  "96f89df3-647e-482c-99b6-74a702fdd283"
                    }
                ]
}

 Number of Divisions (i.e. resultcount): 3
 Division ID where creating GeoFence:  5aad84aa-3496-467d-ba36-aa2efbf6b815
```

```
 ***  Calling CreateGeoFence  ***

 JSON Request:
{
   "authorization":{
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
   "correlationId":"RANDOM-0-002",
   "disableResponseCompression":true,
   "divisionId":"5aad84aa-3496-467d-ba36-aa2efbf6b815",
   "id":null,
   "lat1":39.749065,
   "lat2":39.746049,
   "lat3":39.751864,
   "lat4":39.75504,
   "lon1":-105.217103,
   "lon2":-105.223367,
   "lon3":-105.233152,
   "lon4":-105.224258,
   "maxAltitudeInMeters":1828,
   "minAltitudeInMeters":1524,
   "name":"Colorado School of Mines",
   "priority":500
}

 JSON Response:
{
    "correlationId":  "RANDOM-0-002",
    "result":  {
                    "divisionId":  "5aad84aa-3496-467d-ba36-aa2efbf6b815",
                    "minAltitudeInMeters":  1524,
                    "maxAltitudeInMeters":  1828,
                    "lat1":  39.749065,
                    "lon1":  -105.217103,
                    "lat2":  39.746049,
                    "lon2":  -105.223367,
                    "lat3":  39.751864,
                    "lon3":  -105.233152,
                    "lat4":  39.75504,
                    "lon4":  -105.224258,
                    "name":  "Colorado School of Mines",
                    "priority":  500,
                    "id":  "279aa3ab-b6fc-45b0-8348-ff93976e91f5"
               }
}
 id of new GeoFence:  279aa3ab-b6fc-45b0-8348-ff93976e91f5
```

```
 ***  Calling GetGeoFences  ***

 JSON Request:
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "aa1e1d5c-2682-4a36-98c5-631bfbad9103",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": [],
  "limit": 100000,
  "offset": 0,
  "sortOrder": null
}
```

JSON Response:


```
{
    "correlationId":  "RANDOM-0-002",
    "resultCount":  3,
    "results":  [
                    {
                        "divisionId":  "5aad84aa-3496-467d-ba36-aa2efbf6b815",
                        "minAltitudeInMeters":  1524,
                        "maxAltitudeInMeters":  1828,
                        "lat1":  39.749065,
                        "lon1":  -105.217103,
                        "lat2":  39.746049,
                        "lon2":  -105.223367,
                        "lat3":  39.751864,
                        "lon3":  -105.233152,
                        "lat4":  39.75504,
                        "lon4":  -105.224258,
                        "name":  "Colorado School of Mines",
                        "priority":  500,
                        "id":  "279aa3ab-b6fc-45b0-8348-ff93976e91f5"
                    },
                    {
                        "divisionId":  "96f89df3-647e-482c-99b6-74a702fdd283",
                        "minAltitudeInMeters":  -100,
                        "maxAltitudeInMeters":  1000,
                        "lat1":  30.254,
                        "lon1":  -85.9,
                        "lat2":  30.214,
                        "lon2":  -85.9,
                        "lat3":  30.214,
                        "lon3":  -85.8,
                        "lat4":  30.254,
                        "lon4":  -85.8,
                        "name":  "Botheration Bayou",
                        "priority":  999,
                        "id":  "5d2fb718-9553-481b-b8be-11ecc82d0262"
                    },
                    {
                        "divisionId":  "96f89df3-647e-482c-99b6-74a702fdd283",
                        "minAltitudeInMeters":  -100,
                        "maxAltitudeInMeters":  1000,
                        "lat1":  30.214,
                        "lon1":  -85.9,
                        "lat2":  30.174,
                        "lon2":  -85.9,
                        "lat3":  30.174,
                        "lon3":  -85.8,
                        "lat4":  30.214,
                        "lon4":  -85.8,
                        "name":  "Panama City Beach - Florida",
                        "priority":  1000,
                        "id":  "61919772-dbff-4c4c-9d64-c9ab8d1247f7"
                    }
                ]
}
```


Number of GeoFences (i.e. resultcount): 3

9 Example: Creating a Geofence Using the Java CLI

This section provides an example of how to call the RiskBand ARIES Manager RESTful API with the RiskBand Java CLI to create a geofence. Before running this example, the following configuration needs to be performed on the RiskBand ARIES Manager:

- Organization Created — in this example the organization is Acme
- User Created in Organization — in this example, the user is Apier
- Division Created in Organization — because Geofences can only be created inside divisions.
- User Apier has “Manage GeoFence” permissions
- No multi-factor authentication is enabled for the user

To create a GeoFence using the RiskBand ARIES Manager API, you will need to make the following calls:

- LoginByUser (optional)
- GetOrganizationDivisions
- CreateGeoFence
- GetGeoFences

Calling LoginByUser

Make the LoginByUser call using a CLI command similar to this:

```
> runRbCli.bat LoginByUser -server dev3.riskband.ws -name acme\apier -passwordHash
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -prettyJson

{
  "name" : "acme\\apier",
  "correlationId" : "7762F-1-1",
  "mustChangePassword" : false,
  "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
  "latestGuiBuildNumber" : 17343,
  "eulaId" : "8d8b65ee-2caf-4f44-92ba-79c1a92a17ee",
  "id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
}
```

The *-prettyJson* option formats the API response.

If you want to authenticate with the user UUID and passwordHash, you will need the UUID of the user you will want to authenticate with:

```
"id" : "aa1e1d5c-2682-4a36-98c5-631bfbad9103"
```

In most cases you can also authenticate API calls made from the CLI using user name and password (see Allowing the Java CLI to Authenticate for You on page 19). However for consistency with the other examples of creating geofences, the user UUID and passwordHash syntax will be used here.

Calling GetOrganizationDivisions

Make the GetOrganizationDivisions call using a CLI command similar to this:

Note: Using the CLI you do not need to specify the organizationId. The CLI assumes that is the same organization where the user that is authenticating resides.

```
> runRbCli.bat GetOrganizationDivisions -server dev3.riskband.ws -authId
aa1e1d5c-2682-4a36-98c5-631bfbad9103 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -prettyJson

{
  "results" : [ {
    "name" : "Div3",
    "description" : null,
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "firmwarePolicyId" : null,
    "securityPolicyId" : "67291a77-dd7f-48c3-8edc-6566c80c253d",
    "emergencyContactInformationId" : null,
    "emergencyResponsePolicyId" : null,
    "shortName" : "divis-3",
    "assignedPowerManagementPolicyId" : null,
    "allowGroupsToUseDivisionDevices" : true,
    "unassignedPowerManagementPolicyId" : "742d435b-e827-493b-bde7-4683803...",
    "allowSelfAssignment" : false,
    "id" : "3354ef17-8935-4174-bde3-789a8a4f5d23"
  }, {
    "name" : "Div2",
    "description" : null,
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "firmwarePolicyId" : null,
    "securityPolicyId" : null,
    "emergencyContactInformationId" : null,
    "emergencyResponsePolicyId" : null,
    "shortName" : "div2",
    "assignedPowerManagementPolicyId" : null,
    "allowGroupsToUseDivisionDevices" : false,
    "unassignedPowerManagementPolicyId" : null,
    "allowSelfAssignment" : false,
    "id" : "5aad84aa-3496-467d-ba36-aa2efbf6b815"
  }, {
    "name" : "Div1",
    "description" : null,
    "organizationId" : "dbd1b8a6-9da4-441d-bc6c-60adcf1a3a94",
    "firmwarePolicyId" : null,
    "securityPolicyId" : null,
    "emergencyContactInformationId" : null,
    "emergencyResponsePolicyId" : null,
    "shortName" : "div1",
    "assignedPowerManagementPolicyId" : null,
    "allowGroupsToUseDivisionDevices" : true,
    "unassignedPowerManagementPolicyId" : null,
    "allowSelfAssignment" : false,
    "id" : "96f89df3-647e-482c-99b6-74a702fdd283"
  } ],
  "correlationId" : "EBDAB-1-1",
  "resultCount" : null
}
```

The value you will need for the CreateGeoFence call is the UUID of the division where you want to create the geofence. In this case it will be the ID of “Div1”:

```
"id" : "96f89df3-647e-482c-99b6-74a702fdd283"
```

Calling CreateGeoFence

Make the CreateGeoFence call using a CLI command similar to this:

```
> runRbCli.bat CreateGeoFence -server dev3.riskband.ws -authId
aa1e1d5c-2682-4a36-98c5-631bfbad9103 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -divisionId
96f89df3-647e-482c-99b6-74a702fdd283 -lat1 39.749065 -lat2 39.746049 -lat3
39.751864 -lat4 39.75504 -lon1 "-105.217103" -lon2 "-105.223367" -lon3
"-105.233152" -lon4 "-105.224258" -name "Colorado School of Mines 1"
-maxAltitudeInMeters 1828 -minAltitudeInMeters 1524 -priority 500 -prettyJson

{
  "result" : {
    "name" : "Colorado School of Mines 1",
    "priority" : 500,
    "lat4" : 39.75504,
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "maxAltitudeInMeters" : 1828,
    "minAltitudeInMeters" : 1524,
    "lat1" : 39.749065,
    "lon1" : -105.217103,
    "lon2" : -105.223367,
    "lat3" : 39.751864,
    "lat2" : 39.746049,
    "lon3" : -105.233152,
    "lon4" : -105.224258,
    "id" : "735ef582-7ed6-492a-a448-1e38e58d02e0"
  },
  "correlationId" : "B0721-1-1"
}
```

Calling GetGeoFences

You can verify that the geofence now exists in the system, but calling GetGeoFences. The call can be made using CLIL command similar to this:

```
> runRbCli.bat GetGeoFences -server dev3.riskband.ws -authId
aa1e1d5c-2682-4a36-98c5-631bfbad9103 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -prettyJson

{
  "results" : [ {
    "name" : "Botheration Bayou",
    "priority" : 999,
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "lat1" : 30.254,
    "lon1" : -85.9,
    "lat2" : 30.214,
    "lat3" : 30.214,
    "lon3" : -85.8,
    "lon2" : -85.9,
    "lat4" : 30.254,
    "lon4" : -85.8,
    "minAltitudeInMeters" : -100,
    "maxAltitudeInMeters" : 1000,
    "id" : "5d2fb718-9553-481b-b8be-11ecc82d0262"
  }, {
    "name" : "Panama City Beach - Florida",
    "priority" : 1000,
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "lat1" : 30.214,
    "lon1" : -85.9,
    "lat2" : 30.174,
    "lat3" : 30.174,
    "lon3" : -85.8,
    "lon2" : -85.9,
    "lat4" : 30.214,
    "lon4" : -85.8,
    "minAltitudeInMeters" : -100,
    "maxAltitudeInMeters" : 1000,
    "id" : "61919772-dbff-4c4c-9d64-c9ab8d1247f7"
  }, {
    "name" : "Colorado School of Mines 1",
    "priority" : 500,
    "divisionId" : "96f89df3-647e-482c-99b6-74a702fdd283",
    "lat1" : 39.749065,
    "lon1" : -105.217103,
    "lat2" : 39.746049,
    "lat3" : 39.751864,
    "lon3" : -105.233152,
    "lon2" : -105.223367,
    "lat4" : 39.75504,
    "lon4" : -105.224258,
    "minAltitudeInMeters" : 1524,
    "maxAltitudeInMeters" : 1828,
    "id" : "735ef582-7ed6-492a-a448-1e38e58d02e0"
  } ],
  "correlationId" : "77AC2-1-1",
  "resultCount" : null
}
```

10 Example: Creating a Geofence using the Java SDK

This section provides an example of how to use the RiskBand Java SDK to create a geofence.

Before running these examples, the following configuration needs to be performed on the RiskBand ARIES Manager:

- Organization Created — in this example the organization is Acme
- User Created in Organization — in this example, the user is Apier
- Division Created in Organization — because Geofences can only be created inside divisions.
- User Apier has “Manage GeoFence” permissions
- No multi-factor authentication is enabled for the user

To create a geofence using the RiskBand ARIES Manager API, you will need to make the following calls:

- RiskBandClientImpl
- GetOrganizationDivisions
- CreateGeoFence
- GetGeoFences

The Java program used in this example is a single file; however, the sections for each call will be broken out and described separately below.

Calling RiskBandClientImpl

The first thing you need to do is create a RiskBand Client object by calling *RiskBandClientImpl*. The inputs for this call are the RiskBand ARIES Manager server you want to work with, the user name you want to authenticate with, and the password. For example, the call might look something like this:

```
final RiskBandClient client = new RiskBandClientImpl(
"https://dev3.riskband.ws",
"acme\\apier",
"password123" );
```

After creating this object, you can access all the RiskBand APIs methods belonging to this object. For example, using the code provided above, the *CreateGeoFence* call could be invoked by calling `client.createGeoFence`.

Calling GetOrganizationDivisions

As geofences need to be created inside divisions, you will need to get the UUID of the division where you want the geofence to reside. This can be accomplished by calling the *GetOrganizationDivisions*, which returns a list of all the divisions in an organization along with their names and UUIDs.

The code to accomplish this looks something like this:

```
final GetOrganizationDivisionsResponse orgDivs =
client.getOrganizationDivisions(
(GetOrganizationDivisions) BeanFactory.newBean(GetOrganizationDivisions.class)
.setComputeResultCount(true));
```

The response will come back as an array of nested arrays. In order to process the sub arrays you will want to include the `.setComputeResultCount(true)` parameter when making the call. Then, using the result count you can process the returned arrays:

```
System.out.print("Divisions Count: " + orgDivs.getResultCount() + "\n    " );
for ( int i = 0; i < orgDivs.getResultCount(); ++i ) {
   System.out.print(orgDivs.getResults()[i].getName() );
   System.out.print(": " + orgDivs.getResults()[i].getId() + "\n    " );
}
```

If the API call is successful, the console output from the call would look like this:

```
Divisions Count: 3
   Div3: 3354ef17-8935-4174-bde3-789a8a4f5d23
   Div2: 5aad84aa-3496-467d-ba36-aa2efbf6b815
   Div1: 96f89df3-647e-482c-99b6-74a702fdd283
```

This example uses the second division that was returned, so the UUID of the second division (5aad84aa-3496-467d-ba36-aa2efbf6b815) is placed in the variable `divId` for later use.

```
UUID divId = orgDivs.getResults()[1].getId();
```

Calling CreateGeoFence

Using the UUID of the division that was put in the `divId` variable, the *CreateGeoFence* call can be made like this:

```
client.createGeoFence (
BeanFactory.newBean( CreateGeoFence.class )
.setDivisionId(divId)
.setLat1(39.749065)
.setLat2(39.746049)
.setLat3(39.751864)
.setLat4(39.75504)
.setLon1(-105.217103)
.setLon2(-105.223367)
.setLon3(-105.233152)
.setLon4(-105.224258)
.setMaxAltitudeInMeters(1828)
.setMinAltitudeInMeters(1524)
.setName( "Colorado School of Mines" )
.setPriority(500) );
}
```

Calling GetGeoFences

You can verify that the geofence now exists in the system, by calling GetGeoFences. Again this call returns nested arrays, so the call is made with `.setComputeResultCount(true)` as a parameter:

```
final GetGeoFencesResponse GF = client.getGeoFences(
(GetGeoFences) BeanFactory.newBean( GetGeoFences.class )
.setComputeResultCount(true));

System.out.print("Geofences Count: " + GF.getResultCount() + "\n   " );
for ( int i = 0; i < GF.getResultCount(); ++i ) {
System.out.print(GF.getResults()[i].getName() );
System.out.print("  (Division: "+GF.getResults()[i].getDivisionId()+")\n   " );
}
```

If the API call completes successfully, the response will look something like this:

```
Getting GeoFence
Geofences Count: 3
   Colorado School of Mines  (Division: 5aad84aa-3496-467d-ba36-aa2efbf6b815)
   Botheration Bayou  (Division: 96f89df3-647e-482c-99b6-74a702fdd283)
   Panama City Beach - Florida  (Division: 96f89df3-647e-482c-99b6-74a702fdd283)
```

Full Listing of CreateGeoFenceExample.java

```
/*************************************************************************
 *
 * Copyright C 2019, Whereable Technologies LLC and/or its affiliates.
 * All rights reserved.
 *
 *************************************************************************/

import java.util.UUID;

import ws.riskband.api.rest.request.GetOrganizationDivisions;
import ws.riskband.api.rest.request.CreateGeoFence;
import ws.riskband.api.rest.request.GetGeoFences;

import ws.riskband.api.rest.response.GetOrganizationDivisionsResponse;
import ws.riskband.api.rest.response.GetGeoFencesResponse;

import ws.riskband.sdk.rb.client.RiskBandClient;
import ws.riskband.sdk.rb.client.RiskBandClientImpl;
import ws.riskband.util.bean.BeanFactory;

// Create RiskBand Client object
public class CreateGeoFenceExample {
  public static void main(String[] args)  {

    org.apache.log4j.Logger.getRootLogger().setLevel(org.apache.log4j.Level.OFF);

    final RiskBandClient client = new RiskBandClientImpl(
        "https://dev3.riskband.ws",
                "acme\\apier",
                "password123" );

// Get list of organizations
    System.out.println("\nGetting Divisions in Organization");
        final GetOrganizationDivisionsResponse orgDivs = client.getOrganizationDivisions(
               (GetOrganizationDivisions) BeanFactory.newBean(GetOrganizationDivisions.class)
                .setComputeResultCount(true));
        System.out.print("Divisions Count: " + orgDivs.getResultCount() + "\n   " );
        for ( int i = 0; i < orgDivs.getResultCount(); ++i ) {
          System.out.print(orgDivs.getResults()[i].getName() );
          System.out.print(": " + orgDivs.getResults()[i].getId() + "\n   " );
        }
        UUID divId = orgDivs.getResults()[1].getId();


// Create Geofence
    System.out.println("\nCreating GeoFence");
        client.createGeoFence (
                BeanFactory.newBean( CreateGeoFence.class )
                .setDivisionId(divId)
                .setLat1(39.749065)
                .setLat2(39.746049)
                .setLat3(39.751864)
                .setLat4(39.75504)
                .setLon1(-105.217103)
                .setLon2(-105.223367)
                .setLon3(-105.233152)
                .setLon4(-105.224258)
                .setMaxAltitudeInMeters(1828)
                .setMinAltitudeInMeters(1524)
                .setName( "Colorado School of Mines" )
                .setPriority(500) );
```

```
// Get Geofences
      System.out.println("\nGetting GeoFence");
        final GetGeoFencesResponse GF = client.getGeoFences(
                (GetGeoFences) BeanFactory.newBean( GetGeoFences.class )
                .setComputeResultCount(true));
        System.out.print("Geofences Count: " + GF.getResultCount() + "\n   " );
        for ( int i = 0; i < GF.getResultCount(); ++i ) {
          System.out.print(GF.getResults()[i].getName() );
          System.out.print("  (Division: " + GF.getResults()[i].getDivisionId() + ")\n   " );
        }
    }
}
```

Full Listing CreateGeoFenceExample.java Output

```
Getting Divisions in Organization
Divisions Count: 3
   Div3: 3354ef17-8935-4174-bde3-789a8a4f5d23
   Div2: 5aad84aa-3496-467d-ba36-aa2efbf6b815
   Div1: 96f89df3-647e-482c-99b6-74a702fdd283

Creating GeoFence

Getting GeoFence
Geofences Count: 3
   Colorado School of Mines  (Division: 5aad84aa-3496-467d-ba36-aa2efbf6b815)
   Botheration Bayou  (Division: 96f89df3-647e-482c-99b6-74a702fdd283)
   Panama City Beach - Florida  (Division: 96f89df3-647e-482c-99b6-74a702fdd283)
```

11 Receiving Notifications from the RiskBand ARIES Manager

The RiskBand ARIES Manager can be configured to send notifications when the following events occur:

- An emergency is registered
- An emergency artifact is uploaded
- An active emergency is closed
- A device is assigned to a user
- A device is unassigned from a user
- A device reports new GPS coordinates
- A device goes offline or comes back online

The mechanism for communicating these events is through a RESTful API call that "posts" a notification type and a JSON payload to the endpoint specified whenever the configured events occur. Consequently, in order to take advantage of the notification hooks in the RiskBand ARIES Manager you will need to have a server configured that can handle RiskBand query parameters.

(For additional information on how to configure the RiskBand ARIES Manager to send these notifications, see the "About Emergency Notifications" section in the *RiskBand ARIES Manager Administration Guide*.)

Configuring Your Server for RiskBand Notifications

The RiskBand ARIES Manager uses the endpoint specified for notifications and appends the following query parameters:

- ?notificationType=EmergencyRegisteredNotification
- ?notificationType=NewEmergencyArtifactNotification
- ?notificationType=EmergencyClosedNotification
- ?notificationType=DeviceAssignedNotification
- ?notificationType=DeviceUnassignedNotificaiton
- ?notificationType=DeviceCalledHomeNotification
- ?notificationType=DeviceOfflineNotification

For example, if the RESTful API endpoint specified were *https://mysecurity.org:3000/notify,* then the notification string that the RiskBand ARIES Manager would send for an emergency would look like this:

```
https://mysecurity.org:3000/notify?notificationType=EmergencyRegisteredNotification
```

Each notification is accompanied by a payload that contains information about the emergency, device, or GPS position.

About Notification Payloads

The payloads that are sent with each *notificationType* are described below:

- EmergencyRegisteredNotification Payload on page 94
- NewEmergencyArtifactNotification Payload on page 99
- EmergencyClosedNotification Payload on page 99
- DeviceAssignedNotification Payload on page 100
- DeviceUnassignedNotification Payload on page 105
- DeviceCalledHomeNotification Payload on page 105
- DeviceOfflineNotification Payload on page 107

Note: All dates and times contained in notification payloads are in Unix epoch time.

EmergencyRegisteredNotification Payload

The data payload that is sent with an emergency registered notification contains the following embedded objects:

- `model` (device model)
- `device`
- `user`
- `latestGPS`
- `emergency`
- `organization`
- `division`
- `personalEci` (personal emergency contact information for the user triggering the emergency)
- `enterpriseEci` (enterprise emergency contact information for organizaiton, division or group where the device resides)
- `group`

Additionally, the payload contains the following two fields:

- `photosUrl` — this is the URL where you can view photos that have been uploaded to the RiskBand ARIES Manager during the emergency.
- `password`—this is a password that you can use in conjunction with the emergency ID to login to the RiskBand ARIES Manager with emergency credentials. Note, however, that you will need to convert the password into a passwordHash in order to use this for emergency credentials.

A high-level overview with JSON formatting of the payload is given below, with a + symbol providing a placeholder for the contents of the embedded object.

Here is a complete example of a `EmergencyRegisteredNotification` payload


```
{
   "notifications":[
      {
         "model":{
            "dateRegistered":1620942124395,
            "description":"None",
            "name":"RB3x",
            "id":"77539469-2c85-4070-bd9c-5cfa696022b2"
         },

"photosUrl":"https://test2.riskband.ws/?request=HtmlDownloadEmergencyPhotos&autho
rization-entityType=EMERGENCY&authorization-id=6297d822-7293-4d82-8315-f3a0721aa1
a4&authorization-passwordHash=UqI7%2FF%2BuPZghycdz6uxaPtcBJJ9s76Qd3sySQDkUQ4g%3D"
,
         "latestGps":{
            "dateRegistered":1621128726858,
            "userId":"0c96e612-c903-4bac-b4d0-8fcb0fc8cd83",
            "deviceId":"1995747f-9dd1-4562-b059-e60e1c1388dd",
            "gpsFix":"GPS_3D",
            "emergencyId":"None",
            "dateCreated":1621128726856,
            "numSatellites":3,
            "latitude":30.17787,
            "longitude":-85.80549,
            "heading":0,
            "speedInKmph":79,
            "altitudeInMeters":100,
            "horizontalUncertaintyInMeters":5,
            "verticalUncertaintyInMeters":0,
            "stale":false,
            "latestForDevice":true,
            "latestForUser":true,
            "id":"876f4032-f1a4-403d-a92f-4be45569303f"
         },
         "emergency":{
            "dateRegistered":1621128844376,
            "userId":"0c96e612-c903-4bac-b4d0-8fcb0fc8cd83",
            "passwordHash":"<concealed>",
            "deviceId":"1995747f-9dd1-4562-b059-e60e1c1388dd",
            "emergencyResponsePolicyId":"876fe1fe-8a1c-49c2-b708-c7939326ab4e",
            "triggerSource":"DEVICE_USER",
            "triggerDetails":"None",
            "dateRetentionNoticeSent":"None",
            "dateCreated":1621128844902,
            "dateClosed":"None",
            "id":"6297d822-7293-4d82-8315-f3a0721aa1a4"
         },
         "user":{
            "dateRegistered":1620943562044,
            "metadata":"None",
            "divisionId":"None",
            "phoneNumber":"None",
            "passwordHash":"<concealed>",
            "partnerId":"None",
            "passwordType":"PASSWORD",
            "groupId":"None",
            "googleApiKey":"None",
            "emergencyContactInformationId":"None",
            "passwordResetEmailCode":"None",
            "passwordResetPhoneCode":"None",
            "primaryPhotoId":"None",
            "restrictAccessToKnownComputers":false,
            "restrictAccessToIps":"None",
            "dateLastActivity":"None",
```

```
            "dateLastPasswordChange":"None",
            "expiredPasswordChangeDeadlineDate":"None",
            "mfaCurrentCode":0,
            "mfaCurrentDate":0,
            "passwordResetExpirationDate":"None",
            "organizationId":"074be768-52b3-43e5-ac6b-7d98ce668afc",
            "mustChangePassword":true,
            "underSecurityLockout":"None",
            "dateOfBirth":"None",
            "lifeSavingInformation":"None",
            "passphraseAllClear":"None",
            "passphraseDuress":"None",
            "citizenship":"None",
            "phoneVerified":true,
            "gender":"None",
            "emailAddress":"None",
            "firstName":"Axel",
            "lastName":"Andersen",
            "mfaRequired":false,
            "emailVerified":true,
            "suspended":false,
            "description":"None",
            "name":"acme\\axel.andersen",
            "id":"0c96e612-c903-4bac-b4d0-8fcb0fc8cd83"
         },
         "organization":{
            "dateRegistered":1620942439551,
            "shortName":"acme",
            "numDevices":0,
            "emergencyResponsePolicyId":"876fe1fe-8a1c-49c2-b708-c7939326ab4e",
            "minEmergencyRetentionInDays":"None",
            "maxEmergencyRetentionInDays":1095,
            "eulaAcceptanceRequired":true,
            "googleApiKey":"AIzaSyAHNhOJbywWKnPp7aMGuZUbU4dEcDWiEFI",
            "minGpsRetentionInHours":"None",
            "inProduction":true,
            "userNameDomain":"acme\\",
            "emailInviteMessage":"Welcome to Acme, Inc.: {EMAILSUFFIX}",
            "smsInviteMessage":"Welcome to Acme, Inc.: {USERNAME} {PASSWORD}
{EMAIL}",
            "defaultPassphraseAllClear":"adrift",
            "defaultPassphraseDuress":"awash",

"assignedPowerManagementPolicyId":"b4c77f2d-63c8-4f46-aa14-cebe99cbf06a",

"unassignedPowerManagementPolicyId":"424807c7-1fae-477c-a0f5-1e3fca9363d7",
            "passwordPolicy":"WEAK",
            "maxPasswordAgeInDays":"None",
            "graceDaysAfterPasswordExpiration":"None",
            "passwordSelfRecoveryPolicy":"ALLOW_VIA_MULTI_FACTOR",
            "deviceCellularPlan":"NORTH_AMERICA",
            "deviceMsisdnPolicy":"OPEN_LOOP",

"emergencyContactInformationId":"71f781f9-8562-4d31-aa76-2f852be78f7b",
            "notesForEmergencyResponse":"None",
            "securityPolicyId":"50e2e058-d1cb-4609-981c-e0c5ae162b44",
            "firmwarePolicyId":"497692f0-1c54-421b-a16f-8ef9bfa9cf33",
            "numUsers":0,
            "numActiveEmergencies":0,
            "numClosedEmergencies":0,
            "partnerAccess":"None",
            "description":"None",
            "name":"Acme, Inc.",
            "id":"074be768-52b3-43e5-ac6b-7d98ce668afc"
         },
         "division":"None",
```

```
"personalEci":"None",
"enterpriseEci":{
   "city":"None",
   "zipCode":"None",
   "country":"None",
   "secondaryPhoneNumber":"None",
   "tertiaryPhoneNumber":"None",
   "secondaryEmailAddress":"None",
   "primaryEmergencyContactName":"None",
   "primaryEmergencyContactPhone":"None",
   "primaryEmergencyContactCountry":"None",
   "primaryEmergencyContactEmail":"None",
   "secondaryEmergencyContactName":"None",
   "secondaryEmergencyContactPhone":"None",
   "secondaryEmergencyContactCountry":"None",
   "secondaryEmergencyContactEmail":"None",
   "state":"None",
   "address":"None",
   "id":"71f781f9-8562-4d31-aa76-2f852be78f7b"
},
"device":{
   "dateRegistered":1620943257058,
   "registeredWithGeosE911":false,
   "divisionId":"None",
   "cliEnabled":false,
   "forceFirmwareBuildId":"None",
   "firmwareBuildId":"e766e052-e981-4333-b6f2-1e42657a2bae",
   "locationStalenessInMins":1,
   "arrayDebugModules":"None",
   "phoneNumber":"467191003473982",
   "locationName":"None",
   "lastBatteryMv":4000,
   "lastCellSignalStrength":50,
   "lastCellSignalQuality":"None",
   "firmwareUpgradeBuildId":"None",
   "callInProgress":"false",
   "locationCertainty":"None",
   "userId":"0c96e612-c903-4bac-b4d0-8fcb0fc8cd83",
   "debugModules":"None",
   "logCallHomeRequired":"None",
   "gpsCallHomeRequired":false,
   "passwordHash":"<concealed>",
   "groupId":"None",
   "cellularCarrier":"TELIT",
   "modelId":"77539469-2c85-4070-bd9c-5cfa696022b2",
   "imei":"<concealed>",
   "uptimeInSecs":7,
   "charging":true,
   "lastBatteryLevel":60,
   "registrationCode":"None",
   "registrationPassword":"None",
   "bootloaderBuildNumber":0,
   "iccid":"<concealed>",
   "cellularCarrierId":"<concealed>",
   "manufacturer":"c0e94b3f-cb9e-49fb-a45c-497c8168ff0b",
   "organizationId":"074be768-52b3-43e5-ac6b-7d98ce668afc",
   "dateOrganizationRegistered":1621009760137,
   "dateLastHeartbeat":1621128727100,
   "lastAuthorizationRotation":1620943257058,
   "fullyManufactured":true,
   "securityPolicyId":"None",
   "powerManagementPolicyId":"None",
   "firmwarePolicyId":"None",
   "inAirplaneMode":false,
   "policyDictatedFirmwareUpgradeId":"None",
   "veryLateCallingHome":false,
```

```
            "reportedAsNeedingRecharge":false,
            "rxBytes":0,
            "txBytes":0,
            "serialNumber":"210513-10001",
            "name":"acme-3",
            "id":"1995747f-9dd1-4562-b059-e60e1c1388dd"
         },
         "group":"None",
         "password":"yK6SU6BzHqmQ"
      }
   ]
}
```

NewEmergencyArtifactNotification Payload

The NewEmergencyArtifactNotification will alert endpoints that an emergency artifact has been fully and successfully uploaded to the server. This notification is sent whether the emergency is open or closed, and whether it is uploaded by a device or by an authorized system administrator. Here is an example of the JSON object:

```
{
  'notifications': [
    {
      'dateCreated': 1636824768890,
      'dateRegistered': 1636824789078,
      'emergencyId': 'e8fdc182-1d52-4c36-885d-9184016d7e01',
      'type': 'PHOTO',
      'name': '1636824768.890.jpeg',
      'id': '8b4a8734-28cc-4cc0-984a-e6de7e2a4676'
    }
  ]
}
```

EmergencyClosedNotification Payload

When an emergency is closed, a closed emergency object is created. The UUID for the closed emergency object is contained in the `id` field of the JSON payload. This is the UUID that you will need if you want to get additional information about the closed emergency (for example, to see the notes that were entered when the emergency was closed).

Here is a complete example of an `EmergencyClosedNotification` payload:

```
{
  'notifications': [
    {
      'emergency': {
        'dateCreated': 1636824768090,
        'dateClosed': 1636824974490,
        'dateRegistered': 1636824783416,
        'userId': 'd9d3605f-399d-4821-985e-6804d8976d35',
        'passwordHash': '<concealed>',
        'deviceId': 'a5a1dc88-35f6-4405-bffa-c0b6faea5afd',
        'emergencyResponsePolicyId': 'd8890433-63cd-4546-982d-1c48b424d199',
        'triggerSource': 'DEVICE_USER',
        'triggerDetails': None,
        'dateRetentionNoticeSent': None,
        'id': 'e8fdc182-1d52-4c36-885d-9184016d7e01'
      }
    }
  ]
}
```

DeviceAssignedNotification Payload

When an device is assigned to a user in the RiskBand ARIES Manager, the payload that is sent with the device assigned notification contains embedded objects. A high-level overview of the payload is given below, with a + symbol providing a placeholder for the contents of the embedded object.

Additionally, an administrator in the RiskBand ARIES Manager can select *Force Third Party Re-Synchronization* which will send a notification payload that contains a list of all the devices in the organization that have been assigned (if they have been assigned an

emergency response policy that calls for sending device assign notifications). In these situations, the payload will contain multiple `notifications`. For example, here is a high-level example of a payload that contains three notifications:

Here is a complete example of a `DeviceAssignedNotification` payload:


```
{
   "notifications":[
      {
         "model":{
            "dateRegistered":1583254098111,
            "description":"None",
            "name":"RB3x",
            "id":"a3fb260b-7ca2-4310-8d6d-17cbaa74574e"
         },
         "device":{
            "groupId":"None",
            "registeredWithGeosE911":false,
            "manufacturer":"e513f986-4d92-4eef-9df2-d74242752f6c",
            "organizationId":"a75e866c-06b5-497b-8936-e861ef01ae49",
            "dateOrganizationRegistered":1631809207715,
            "dateLastHeartbeat":1636824470285,
            "lastAuthorizationRotation":1636824253518,
            "fullyManufactured":true,
            "securityPolicyId":"None",
            "powerManagementPolicyId":"None",
            "firmwarePolicyId":"None",
            "policyDictatedFirmwareUpgradeId":"None",
            "inAirplaneMode":false,
            "deprecatedDate":"None",
            "veryLateCallingHome":false,
            "reportedAsNeedingRecharge":false,
            "rxBytes":0,
            "txBytes":0,
            "cellularCarrier":"ATT",
            "divisionId":"36da2198-c09a-4451-be4e-e2c8ff38e141",
            "modelId":"a3fb260b-7ca2-4310-8d6d-17cbaa74574e",
            "dateRegistered":1614015922929,
            "forceFirmwareBuildId":"None",
            "deprecatedReason":"None",
            "lastCellSignalStrength":42,
            "registrationCode":"None",
            "registrationPassword":"None",
            "imei":"<concealed>",
            "locationName":"None",
            "iccid":"<concealed>",
            "cliEnabled":true,
            "debugModules":"None",
            "firmwareBuildId":"be1baa11-33f5-4181-9d95-a1f40ccd25c5",
            "bootloaderBuildNumber":19400,
            "lastCellSignalQuality":71,
            "firmwareUpgradeBuildId":"None",
            "callInProgress":"false",
            "userId":"d9d3605f-399d-4821-985e-6804d8976d35",
            "phoneNumber":"13237659172",
            "uptimeInSecs":254,
            "charging":false,
            "lastBatteryLevel":100,
            "lastBatteryMv":4054,
            "logCallHomeRequired":"None",
            "gpsCallHomeRequired":false,
            "locationStalenessInMins":"None",
            "arrayDebugModules":"None",
            "locationCertainty":"None",
            "passwordHash":"<concealed>",
            "cellularCarrierId":"<concealed>",
            "serialNumber":"200303-10001",
            "name":"div1-4",
            "id":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd"
         },
```

```
            "user":{
               "metadata":"None",
               "groupId":"None",
               "googleApiKey":"None",
               "organizationId":"a75e866c-06b5-497b-8936-e861ef01ae49",
               "divisionId":"36da2198-c09a-4451-be4e-e2c8ff38e141",
               "dateRegistered":1583254300556,
               "phoneNumber":"None",
               "passwordHash":"<concealed>",
               "passwordType":"SECRET_KEY",
               "partnerId":"None",

         "emergencyContactInformationId":"e2aa10d9-cf9d-421f-8f4c-f860c9a8ae46",
               "primaryPhotoId":"None",
               "mustChangePassword":true,
               "underSecurityLockout":"None",
               "emailAddress":"None",
               "firstName":"Williston",
               "lastName":"Smythe-Rumfelt",
               "mfaRequired":false,
               "gender":"MALE",
               "dateOfBirth":936856800000,
               "lifeSavingInformation":"Life Saving Notes Williston",
               "citizenship":"USA",
               "phoneVerified":true,
               "dateLastActivity":"None",
               "dateLastPasswordChange":"None",
               "emailVerified":true,
               "passwordResetPhoneCode":"None",
               "passwordResetEmailCode":"None",
               "passphraseAllClear":"custom all-clear",
               "passphraseDuress":"custom-duress",
               "restrictAccessToKnownComputers":false,
               "restrictAccessToIps":"None",
               "expiredPasswordChangeDeadlineDate":"None",
               "mfaCurrentCode":0,
               "mfaCurrentDate":0,
               "passwordResetExpirationDate":"None",
               "suspended":false,
               "description":"Notes",
               "name":"acme\\williston.smythe.rumfelt",
               "id":"d9d3605f-399d-4821-985e-6804d8976d35"
            },
            "organization":{
               "shortName":"acme",
               "eulaAcceptanceRequired":true,
               "googleApiKey":"AIzaSyA9lHeMfXSB_e2E6FXDLJmTabOPzRGtH9g",
               "securityPolicyId":"92def819-4992-4c32-9308-bb5effad6372",
               "firmwarePolicyId":"ec74635d-beb7-4da8-89a3-64624e1b0479",
               "dateRegistered":1583254185449,
               "numDevices":0,
               "emergencyResponsePolicyId":"0e1f5a65-d5f9-48f0-bd71-6959a653a5b8",
               "minEmergencyRetentionInDays":"None",
               "maxEmergencyRetentionInDays":1095,
               "inProduction":true,
               "userNameDomain":"acme\\",
               "minGpsRetentionInHours":"None",
               "deviceCellularPlan":"NORTH_AMERICA",
               "deviceMsisdnPolicy":"OPEN_LOOP",
               "emailInviteMessage":"Welcome to Acme, Corp.: {EMAILSUFFIX}",
               "smsInviteMessage":"Welcome to Acme, Corp.: {USERNAME} {PASSWORD}
{EMAIL}",
               "defaultPassphraseAllClear":"adrift",
               "defaultPassphraseDuress":"awash",

         "assignedPowerManagementPolicyId":"04cc23ba-66d1-408d-9426-2db12bd3ae5a",
```

```
"unassignedPowerManagementPolicyId":"f0806627-5d30-4b7f-b4fe-3c72ae5a5b2a",
            "passwordPolicy":"WEAK",
            "maxPasswordAgeInDays":"None",
            "graceDaysAfterPasswordExpiration":"None",
            "passwordSelfRecoveryPolicy":"ALLOW_VIA_MULTI_FACTOR",

"emergencyContactInformationId":"98e2c74c-a10c-46ad-970a-1694a707c685",
            "notesForEmergencyResponse":"None",
            "numUsers":0,
            "numActiveEmergencies":0,
            "numClosedEmergencies":0,
            "partnerAccess":"None",
            "description":"None",
            "name":"Acme, Corp.",
            "id":"a75e866c-06b5-497b-8936-e861ef01ae49"
         },
         "division":{
            "shortName":"div1",
            "organizationId":"a75e866c-06b5-497b-8936-e861ef01ae49",
            "securityPolicyId":"None",
            "firmwarePolicyId":"45438f61-8e11-4a6d-94cf-6c6e84c865de",
            "emergencyResponsePolicyId":"d8890433-63cd-4546-982d-1c48b424d199",
            "allowGroupsToUseDivisionDevices":true,
            "allowSelfAssignment":false,

"assignedPowerManagementPolicyId":"58044cf0-cbab-4184-abaa-5fdce7370b17",
            "unassignedPowerManagementPolicyId":"None",
            "emergencyContactInformationId":"None",
            "notesForEmergencyResponse":"None",
            "description":"None",
            "name":"Div1",
            "id":"36da2198-c09a-4451-be4e-e2c8ff38e141"
         },
         "personalEci":{
            "city":"Grants",
            "zipCode":"87020",
            "country":"USA",
            "secondaryPhoneNumber":"555-666-5555",
            "tertiaryPhoneNumber":"None",
            "secondaryEmailAddress":"None",
            "primaryEmergencyContactName":"Mrs. Smythe-Rumfelt",
            "primaryEmergencyContactPhone":"555-666-5554",
            "primaryEmergencyContactCountry":"USA",
            "primaryEmergencyContactEmail":"mrs.smythe.rumfelt@homesweethome.com",
            "secondaryEmergencyContactName":"Mr. Smythe-Rumfelt",
            "secondaryEmergencyContactPhone":"555-666-5557",
            "secondaryEmergencyContactCountry":"USA",

"secondaryEmergencyContactEmail":"mr.smythe.rumfelt@homesweethome.com",
            "state":"NM",
            "address":"356 CR 45",
            "id":"e2aa10d9-cf9d-421f-8f4c-f860c9a8ae46"
         },
         "enterpriseEci":{
            "city":"None",
            "zipCode":"None",
            "country":"None",
            "secondaryPhoneNumber":"None",
            "tertiaryPhoneNumber":"None",
            "secondaryEmailAddress":"None",
            "primaryEmergencyContactName":"None",
            "primaryEmergencyContactPhone":"None",
            "primaryEmergencyContactCountry":"None",
            "primaryEmergencyContactEmail":"None",
            "secondaryEmergencyContactName":"None",
```

```
            "secondaryEmergencyContactPhone":"None",
            "secondaryEmergencyContactCountry":"None",
            "secondaryEmergencyContactEmail":"None",
            "state":"None",
            "address":"None",
            "id":"98e2c74c-a10c-46ad-970a-1694a707c685"
         },
         "group":"None"
      }
   ]
}
```

DeviceUnassignedNotification Payload

Here is a complete example of a `DeviceUnassignedNotification` payload:

```
{
   "notifications":[
      {
         "deviceId":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd"
      }
   ]
}
```

Additionally, an administrator in the RiskBand ARIES Manager can select to *Force Third Party Re-Synchronization* which will send a notification payload that contains a list of all the devices that have been unassigned (if they have been assigned an emergency response policy that specifies sending device unassign notifications). In this context, more than one unassigned `deviceId` will be included in the payload. For example,

```
{
 'notifications':[
  {
   'deviceId':'1bad6f06-3405-406e-8c93-605e984ac401'
  },
  {
   'deviceId':'bb6b7796-2354-48a4-af57-ed64d93c93d6'
  },
  {
   'deviceId':'5c000f88-22b5-49c2-aece-a0c5ccc1f1d5'
  }
 ]
}
```

DeviceCalledHomeNotification Payload

If you have specified notifications for “All Events” in the emergency response policy, then you will be sent every new GPS coordinate acquired by the device—even if there is no emergency in effect. However, these coordinates do not arrive until the device calls home, and there may be multiple `gpsPositions` objects in the notification. See DeviceCalledHomeNotification Payload on page 105 for information about the `NewGpsPositionNotification` object.

An example of a call home payload that contains multiple GPS coordinates looks like this:


```
{
   "notifications":[
      {
         "device":{
            "lastCellSignalStrength":35,
            "firmwareBuildId":"be1baa11-33f5-4181-9d95-a1f40ccd25c5",
            "bootloaderBuildNumber":19400,
            "lastCellSignalQuality":57,
            "firmwareUpgradeBuildId":"None",
            "callInProgress":"false",
            "uptimeInSecs":571,
            "charging":false,
            "lastBatteryLevel":98,
            "lastBatteryMv":4045
         },
         "dateRegistered":1636824789194,
         "locationName":"None",
         "userId":"d9d3605f-399d-4821-985e-6804d8976d35",
         "locationStalenessInMins":0,
         "locationCertainty":"None",
         "deviceId":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd",
         "gpsPositions":[
            {},
            {},
            {}
         ]
      },
      {
         "device":{
            "lastCellSignalStrength":35,
            "firmwareBuildId":"be1baa11-33f5-4181-9d95-a1f40ccd25c5",
            "bootloaderBuildNumber":19400,
            "lastCellSignalQuality":57,
            "firmwareUpgradeBuildId":"None",
            "callInProgress":"false",
            "uptimeInSecs":577,
            "charging":false,
            "lastBatteryLevel":98,
            "lastBatteryMv":4045
         },
         "dateRegistered":1636824793051,
         "locationName":"None",
         "userId":"d9d3605f-399d-4821-985e-6804d8976d35",
         "locationStalenessInMins":0,
         "locationCertainty":"None",
         "deviceId":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd",
         "gpsPositions":[ ]
      },
      {
         "device":{ },
         "dateRegistered":1636824798014,
         "locationName":"None",
         "userId":"d9d3605f-399d-4821-985e-6804d8976d35",
         "locationStalenessInMins":0,
         "locationCertainty":"None",
         "deviceId":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd",
         "gpsPositions":[
            {}
         ]
      }
   ]
}
```

During a triggered emergency, the GPS coordinates contained in the initial emergency notification may be stale. This is due to the fact that if a GPS fix cannot be acquired immediately, it may take some time for the underlying hardware and software to acquire a GPS position. There may even be rare challenging environments where a RiskBand device (or any device) will never be able to acquire a GPS location (e.g. the device is in an underground tunnel). Nevertheless, the device attempts to acquire and send GPS coordinates as quickly as possible in accordance with the Call Home Frequency specified in the emergency response policy. (

Note: In the initial emergency notification, the `emergencyId` field will be set to None. After the the emergency is closed this field will be populated with the UUID of the emergency.

DeviceOfflineNotification Payload

Be aware that this notification is sent when a device goes offline and when the device comes back on line, The difference between an online and offline notification is that "eventType" is either DEVICE_CAME_ONLINE or DEVICE_WENT_OFFLINE . . ." Here is an example of a device


```
{
   "notifications":[
      {
         "device":{
            "groupId":"None",
            "registeredWithGeosE911":false,
            "manufacturer":"e513f986-4d92-4eef-9df2-d74242752f6c",
            "organizationId":"a75e866c-06b5-497b-8936-e861ef01ae49",
            "dateOrganizationRegistered":1631809207715,
            "dateLastHeartbeat":1636825241784,
            "lastAuthorizationRotation":1636824253518,
            "fullyManufactured":true,
            "securityPolicyId":"None",
            "powerManagementPolicyId":"None",
            "firmwarePolicyId":"None",
            "policyDictatedFirmwareUpgradeId":"None",
            "inAirplaneMode":false,
            "deprecatedDate":"None",
            "veryLateCallingHome":true,
            "reportedAsNeedingRecharge":false,
            "rxBytes":0,
            "txBytes":0,
            "cellularCarrier":"ATT",
            "divisionId":"36da2198-c09a-4451-be4e-e2c8ff38e141",
            "modelId":"a3fb260b-7ca2-4310-8d6d-17cbaa74574e",
            "dateRegistered":1614015922929,
            "forceFirmwareBuildId":"None",
            "deprecatedReason":"None",
            "lastCellSignalStrength":42,
            "registrationCode":"None",
            "registrationPassword":"None",
            "imei":"<concealed>",
            "locationName":"None",
            "iccid":"<concealed>",
            "cliEnabled":true,
            "debugModules":"None",
            "firmwareBuildId":"be1baa11-33f5-4181-9d95-a1f40ccd25c5",
            "bootloaderBuildNumber":19400,
            "lastCellSignalQuality":43,
            "firmwareUpgradeBuildId":"None",
            "callInProgress":"false",
            "userId":"d9d3605f-399d-4821-985e-6804d8976d35",
            "phoneNumber":"<concealed>",
```

```
            "uptimeInSecs":1022,
            "charging":false,
            "lastBatteryLevel":94,
            "lastBatteryMv":4022,
            "logCallHomeRequired":"None",
            "gpsCallHomeRequired":false,
            "locationStalenessInMins":"None",
            "arrayDebugModules":"None",
            "locationCertainty":"None",
            "passwordHash":"<concealed>",
            "cellularCarrierId":"<concealed>",
            "serialNumber":"200303-10001",
            "name":"div1-4",
            "id":"a5a1dc88-35f6-4405-bffa-c0b6faea5afd"
         },
         "eventType":"DEVICE_WENT_OFFLINE_WHILE_NORMAL_BATTERY",
         "user":{
            "metadata":"None",
            "groupId":"None",
            "googleApiKey":"None",
            "organizationId":"a75e866c-06b5-497b-8936-e861ef01ae49",
            "divisionId":"36da2198-c09a-4451-be4e-e2c8ff38e141",
            "dateRegistered":1583254300556,
            "phoneNumber":"None",
            "passwordHash":"<concealed>",
            "passwordType":"SECRET_KEY",
            "partnerId":"None",

"emergencyContactInformationId":"e2aa10d9-cf9d-421f-8f4c-f860c9a8ae46",
            "primaryPhotoId":"None",
            "mustChangePassword":true,
            "underSecurityLockout":"None",
            "emailAddress":"None",
            "firstName":"Williston",
            "lastName":"Smythe-Rumfelt",
            "mfaRequired":false,
            "gender":"MALE",
            "dateOfBirth":936856800000,
            "lifeSavingInformation":"Life Saving Notes Williston",
            "citizenship":"USA",
            "phoneVerified":true,
            "dateLastActivity":"None",
            "dateLastPasswordChange":"None",
            "emailVerified":true,
            "passwordResetPhoneCode":"None",
            "passwordResetEmailCode":"None",
            "passphraseAllClear":"custom all-clear",
            "passphraseDuress":"custom-duress",
            "restrictAccessToKnownComputers":false,
            "restrictAccessToIps":"None",
            "expiredPasswordChangeDeadlineDate":"None",
            "mfaCurrentCode":0,
            "mfaCurrentDate":0,
            "passwordResetExpirationDate":"None",
            "suspended":false,
            "description":"Notes",
            "name":"acme\\williston.smythe.rumfelt",
            "id":"d9d3605f-399d-4821-985e-6804d8976d35"
         }
      }
   ]
}
```

12 Working with Emergencies

When a device registers an emergency with the RiskBand ARIES Manager, an `emergency` object is created. Using the `id` of the `emergency` object, you can get information about the device and user that triggered the emergency which can be used in the crisis response.

When the emergency is closed, the emergency object persists. The chief difference between an active emergency and a closed emergency is that a closed emergency has a value set in the `dateClosed` property of the `emergency` object.

There are two ways to get the `id` of an `emergency` object:

- You can query the RiskBand ARIES Manager (pull) —This is accomplished by making the `GetEmergencies` call and then using the object ids that are returned in the payload to query the RiskBand ARIES Manager for additional information.
- You can have it sent (push)—This is accomplished by specifying a RESTful API endpoint in the *HTTPS Endpoint to Notify* field of the device's emergency response policy. The notification that will be sent to the endpoint contains several embedded objects, including the `emergency` object and `EmergencyContactInformation` objects. When the emergency is closed, the notification that is sent is just the emergency object that now contains the date the emergency was closed.

Calling GetEmergencies

In order to successfully call `GetEmergencies`, you will need to authenticate with a user whose permissions include either

- Super Administrator, or
- View Emergencies and, optionally, Track Users and Devices

If the user making the call is not at the top of the organizational hierarchy, then the emergencies that can be seen will be limited to those triggered in its division and subordinate groups, or, just to those emergencies triggered by devices in its group.

Below are examples of how to make the `GetEmergencies` call. Note the following items about the response payload:

- If the `dateClosed` property is `null`, then the emergency is active.
- The `id` of the emergency and the `passwordHash` can be used as EMERGENCY credentials to get emergency artifacts

GetEmergencies Response Payload

The response payload for the `GetEmergencies` call can, and often does, contain embedded `emergency` objects. For the examples that follow (except the Java SDK example), the successful response to the call will look something like this:


```
{{
"correlationId":null,
"resultCount":2,
"results":[
   {
      "dateCreated":1575404771987,
      "dateClosed":1575404790699,
      "dateRegistered":1575404773921,
      "userId":"d46b54a8-3388-46b3-9b68-f5c2f22fcfb3",
      "passwordHash":"2xaWbk9uP+BHMGOyYp/WvobZKGYpz4FR/fn/sB9KzQg=",
      "deviceId":"0a6547de-2f2b-4a8b-a356-5b5bf1011876",
      "dateRetentionNoticeSent":null,
      "emergencyResponsePolicyId":"d22c7104-d943-46b6-94ee-84c5f676c4d2",
      "triggerSource":"DEVICE_USER",
      "triggerDetails":null,
      "id":"1b9fafb3-fcf4-443d-a327-a9ddaaad22c7"
   },
   {
      "dateCreated":1576279579651,
      "dateClosed":null,
      "dateRegistered":1576279580551,
      "userId":"dedb90d7-0ef6-49fa-9009-07cece9104ab",
      "passwordHash":"wtmKXNV4/NEUkWBn9YixhZ6+kFmYOEY7AuU79Lc0Tig=",
      "deviceId":"f1c7037d-da7c-474f-877d-2d70956ae840",
      "dateRetentionNoticeSent":null,
      "emergencyResponsePolicyId":"f57ea2db-5d9b-4516-a75e-6f9fd88a4ab8",
      "triggerSource":"DEVICE_USER",
      "triggerDetails":null,
      "id":"9390e9de-51c1-407f-91d6-9d8e9cd4a7de"
   }
]
}
```


Note: In the response payload above, the value null for `dateClosed` indicates the emergency is active.,

If no emergencies are found the response will look like this:


```
{
  "results" : [ ],
  "correlationId" : "8519A-1-1",
  "resultCount" : 0
}
```


Calling GetEmergencies from a Browser

An example of a command that can be entered in a browser's address bar to get all opened and closed emergencies in the RiskBand ARIES Manager is


```
https://dev3.riskband.ws/?request=GetEmergencies&authorization-entityType=USER&au
thorization-id=074c2dfb-f53a-4033-a6ba-ca2f1986e08e&authorization-passwordHash=vt
TvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=&computeResultCount=true
```

Calling GetEmergencies using cURL

An example of a cURL command that will get all open and closed emergencies is

```
curl -iv -X PUT -H "Content-Length: 348" -H "Content-Type: application/json" -d
@_GetEmergencies_348.json  https://dev3.riskband.ws?GetEmergencies=348U
```

The `_GetEmergencies_348.json` request file that is sent to the server looks like this:

```
{
  "authorization": {
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": null,
  "limit": 1000,
  "offset": 0,
  "sortOrder": null
}
```

An example of a cURL command that will get only active emergencies is

```
curl -iv -X PUT -H "Content-Length: 453" -H "Content-Type: application/json" -d
@_GetEmergencies_active_453.json  https://dev3.riskband.ws?GetEmergencies=453U >
GetEmergencies_active_453.rsp
```

The `_GetEmergencies_active_453.json` request file that is sent to the server looks like this:

```
  {
    "authorization": {
      "entityType": "USER",
      "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
      "mfaCode": null,
      "id": "074c2dfb-f53a-4033-a6ba-ca2f1986e08e"
    },
    "filters": [
      {
        "property": "dateClosed",
        "value": null,
        "operator": "EQUALS"
      }
    ],
    "limit": 1000,
    "offset": 0,
    "sortOrder": null,
    "correlationId": "A2330-1-1",
    "any": false,
    "computeResultCount": true,
    "disableResponseCompression": false
  }
```

Calling GetEmergencies using the Java CLI

An example of a Java CLI command that will get all open and closed emergencies is

```
D:\CLI> runRbCli.bat GetEmergencies -server dev3.riskband.ws -authId
074c2dfb-f53a-4033-a6ba-ca2f1986e08e -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -computeResultCount
TRUE -prettyJson
```

An example of a Java CLI command that will get just active emergencies looks like this:

```
D:\CLI> runRbCli.bat GetEmergencies -server dev3.riskband.ws -authId
074c2dfb-f53a-4033-a6ba-ca2f1986e08e -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -computeResultCount
TRUE -filters-property "dateClosed" -filters-operator EQUALS -prettyJson
```

Note: In the get active emergencies example above, the filter options would normally be `-filters-property "dateClosed" -filters-value NULL -filters-operator EQUALS`. However, specifying NULL on the command line puts quotation marks around the value in the request payload which causes the command to fail. The workaround is to remove the `-filters-value` option from the command line. In the CLI, options that are not specified, default to null.

Calling GetEmergencies from PowerShell

An example of a PowerShell script that will get all open and closed emergencies is

```
$json = @"
{
  "authorization": {
    "entityType": "USER",
    "id": "074c2dfb-f53a-4033-a6ba-ca2f1986e08e",
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filters": null,
  "limit": 1000,
  "offset": 0,
  "sortOrder": null
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetEmergencies=" +$jsonLen +"U"

############################################################
# GetEmergencies
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json

Write-Host ("`n" +($response | ConvertTo-Json))
```

An example of a PowerShell script that will get just active emergencies looks like this:

```
$json = @"
{
  "authorization": {
    "entityType": "USER",
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=",
    "mfaCode": null,
    "id": "074c2dfb-f53a-4033-a6ba-ca2f1986e08e"
  },
  "filters": [
    {
      "property": "dateClosed",
      "value": null,
      "operator": "EQUALS"
    }
  ],
  "limit": 1000,
  "offset": 0,
  "sortOrder": null,
  "correlationId": "A2330-1-1",
  "any": false,
  "computeResultCount": true,
  "disableResponseCompression": false
}
"@

$jsonLen = $json | measure-object -character | select -expandproperty characters
$apiCall = "https://dev3.riskband.ws?GetEmergencies=" +$jsonLen +"U"

###############################################################
# GetEmergencies
$response = Invoke-RestMethod -Uri $apiCall  -ContentType "application/json"
-Method PUT -Body $json

Write-Host ("`n" +($response | ConvertTo-Json))
```

Calling GetEmergencies using the Java SDK

Example code for a Java class that will get all open and closed emergencies is

```
import ws.riskband.api.rest.request.GetEmergencies;
import ws.riskband.api.rest.response.GetEmergenciesResponse;

import ws.riskband.sdk.rb.client.RiskBandClient;
import ws.riskband.sdk.rb.client.RiskBandClientImpl;
import ws.riskband.util.bean.BeanFactory;

// Create RiskBand Client object
public class GetEmergenciesExample {
 public static void main(String[] args)  {

  org.apache.log4j.Logger.getRootLogger().setLevel(org.apache.log4j.Level.OFF);

  final RiskBandClient client = new RiskBandClientImpl(
    "https://dev3.riskband.ws",
    "acme\\sysadmin2",
    "password123" );

  // Get list of open and closed emergencies
  System.out.println("\nGetting open and closed emergencies");

  final GetEmergenciesResponse emers = client.getEmergencies(
    (GetEmergencies)
BeanFactory.newBean(GetEmergencies.class).setComputeResultCount(true));

  System.out.print("Emergencies count: " + emers.getResultCount() + "\n\n" );

  for ( int i = 0; i < emers.getResultCount(); ++i ) {
    System.out.print("Emergency "+(i+1)+"\n");
    System.out.print("dateCreated :
"+emers.getResults()[i].getDateCreated()+"\n");
    System.out.print("dateRegistered :
"+emers.getResults()[i].getDateRegistered()+"\n");
    System.out.print("passwordHash :
"+emers.getResults()[i].getPasswordHash()+"\n");
    System.out.print("triggerDetails :
"+emers.getResults()[i].getTriggerDetails()+"\n");
    System.out.print("dateClosed : "+emers.getResults()[i].getDateClosed()+"\n");
    System.out.print("dateRetentionNoticeSent :
"+emers.getResults()[i].getDateRetentionNoticeSent()+"\n");
    System.out.print("deviceId : "+emers.getResults()[i].getDeviceId()+"\n");
    System.out.print("emergencyResponsePolicyId :
"+emers.getResults()[i].getEmergencyResponsePolicyId()+"\n");
    System.out.print("id : "+emers.getResults()[i].getId()+"\n");
    System.out.print("triggerSource :
"+emers.getResults()[i].getTriggerSource()+"\n");
   System.out.print("userId : "+emers.getResults()[i].getUserId()+"\n\n"  );
    }
  }
}
```

If the call completes successfully, the response will look something like this

```
Getting open and closed emergencies
Emergencies count: 2

Emergency 1
dateCreated : 1575404771987
dateRegistered : 1575404773921
passwordHash : 2xaWbk9uP+BHMGOyYp/WvobZKGYpz4FR/fn/sB9KzQg=
triggerDetails : null
dateClosed : 1575404790699
```

```
dateRetentionNoticeSent : null
deviceId : 0a6547de-2f2b-4a8b-a356-5b5bf1011876
emergencyResponsePolicyId : d22c7104-d943-46b6-94ee-84c5f676c4d2
id : 1b9fafb3-fcf4-443d-a327-a9ddaaad22c7
triggerSource : DEVICE_USER
userId : d46b54a8-3388-46b3-9b68-f5c2f22fcfb3

Emergency 2
dateCreated : 1576620139157
dateRegistered : 1576620141393
passwordHash : SBGPg5Hxewo3l3m31OUouOlh6vwbmtH2TEfIsFYuY20=
triggerDetails : null
dateClosed : null
dateRetentionNoticeSent : null
deviceId : f1c7037d-da7c-474f-877d-2d70956ae840
emergencyResponsePolicyId : f57ea2db-5d9b-4516-a75e-6f9fd88a4ab8
id : 5155fee3-13eb-4ee4-8493-a75e9931e04a
triggerSource : DEVICE_USER
userId : dedb90d7-0ef6-49fa-9009-07cece9104ab
```

Receiving Emergency Notifications

The data payload that is sent with an emergency registered notification contains the following embedded objects:

- `model` (device model)
- `device`
- `user`
- `latestGPS`
- `emergency`
- `organization`
- `division`
- `personalEci` (personal emergency contact information for the user triggering the emergency)
- `enterpriseEci` (enterprise emergency contact information for organizaiton, division or group where the device resides)
- `group`

Additionally, the payload contains the following two fields:

- `photosUrl` — this is the URL where you can view photos that have been uploaded to the RiskBand ARIES Manager during the emergency.
- `password`—this is a password that you can use in conjunction with the emergency ID to login to the RiskBand ARIES Manager with emergency credentials. Note, however, that you will need to convert the password into a passwordHash in order to use this for emergency credentials.

A high-level overview with JSON formatting of the payload is given below, with a + symbol providing a placeholder for the contents of the embedded object.

Additionally, by using the `id` of the `emergency` object and the `password` as credentials, you can pull the same information from the RiskBand ARIES Manager using the API.

When an emergency is closed, just the emergency object is resent with a value supplied in `dateClosed` property. For closed emergency notifications, the `passwordHash` will be `<concealed>`, so in order to query the closed emergency with emergency credentials, you will need to have retained the password that was sent in the original emergency notification. (For information on how to convert the password to a hash, see Getting the passwordHash on page 19.)

For additional information see Receiving Notifications from the RiskBand ARIES Manager on page 93.

Getting Emergency Contact Information

There are two kinds of emergency contact information in the RiskBand ARIES Manager:

- Organizational — this is emergency contact information for an organization, a division, or a group. This information may include administrator contact information and your Critical Response Team's (CRT) contact information for these hierarchical entities.
- User — this is emergency contact information for the user. This information may include a second phone number for the user, as well as names, phone numbers, and email addresses of emergency contacts.

If you receive an emergency notification, the payload will contain the user emergency contact information in an embedded object called `personalEci` and the organization, division, or group information where the user resides in an embedded object called `enterpriseEci`.

Additionally, the emergency notification payload contains an `organization`, `division`, `group`, and `user` object, and each of these objects contain an `emergencyContactInformationId` that corresponds to their emergency contact information. You can use this ID get the emergency contact information with the `GetEmergencyContactInformation` call .

For example, if you are using cURL a 445-character request payload for `GetEmergencyContactInformation` might look like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "filter": {
    "property": "id",
    "value": "eed144bc-30cc-4bd8-b44e-2689a3d52d2b"
  },
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true
}
```

Here is another example of using the Java CLI to authenticate with emergency credentials to get the organization contact information:

```
D:\CLI> runRbCli.bat GetEmergencyContactInformation -server dev3.riskband.ws
-authId af589ce4-3c9f-408d-a763-0112ce047382 -authKey
500quESkyOUiyFR4I4iGg6g2kac99WUvbjQhpIlX77M= -authType EMERGENCY -filter-property
id -filter-value eed144bc-30cc-4bd8-b44e-2689a3d52d2b -prettyJson
```

In both cases the results would look something like this:

```
{
  "correlationId": "RANDOM-0-002",
  "result": {
    "city": "Acmeville",
    "country": "USA",
    "secondaryPhoneNumber": "303-333-1111",
    "tertiaryPhoneNumber": "303-333-2222",
    "secondaryEmailAddress": "acme.admins.group@acme.com",
    "primaryEmergencyContactName": "CRT Leader",
    "primaryEmergencyContactPhone": "303-333-3333",
    "primaryEmergencyContactCountry": null,
    "primaryEmergencyContactEmail": "crt.leader@crt.com",
    "secondaryEmergencyContactName": "CRT Night Supervisor",
    "secondaryEmergencyContactPhone": "303-333-4444",
    "secondaryEmergencyContactCountry": null,
    "secondaryEmergencyContactEmail": "crt.night.supervison@crt.com",
    "zipCode": "80001",
    "state": "CO",
    "address": "999 Acme Way",
    "id": "eed144bc-30cc-4bd8-b44e-2689a3d52d2b"
  }
}
```

Getting Emergency GPS Coordinates

Whenever a device reports its GPS coordinates to the RiskBand ARIES Manager, a GPS position object is created. If the coordinates are reported as part of an emergency, then the UUID of the emergency is included in the GPS position object.

If you receive an emergency notification from the RiskBand ARIES Manager, the payload will contain the most recent GPS position object for the device *before the emergency*, and with each additional “call home” during the emergency, you will receive updated GPS position notifications.

The GPS position object that is sent as an emergency notification is slightly different that the GPS position object retrieved with an API call. The table below sorts and formats the objects' contents to illustrate the differences.

GPS Position Object from API	GPS Position Object Sent as Notification

<pre>{
  "results" : [{
    "dateRegistered" : 1578609063680,
    "deviceId" : "3808c804-9094-4a43-...",
    "userId" : "d46b54a8-3388-46b3-9b...",
 	
    "altitudeInMeters" : 100,
    "dateCreated" : 1578609062202,
    "dateRegistered" : 1578609063680,
    "deviceId" : "3808c804-9094-4a43-...",
    "emergencyId" : null,
    "gpsFix" : "GPS_3D",
    "heading" : 0,
    "horizontalUncertaintyInMeters" : 5,
    "id" : "c02ea448-fc5e-461a-97b1-7...",
    "latestForDevice" : true,
    "latestForUser" : true,
    "latitude" : 30.38292,
    "longitude" : -85.80549,
    "numSatellites" : 3,
    "speedInKmph" : 79,
    "stale" : false,
    "verticalUncertaintyInMeters" : 0
}</pre>
	<pre>{
	  'dateRegistered': 1578609027949,
	  'deviceId': '3808c804-9094-4a43-b6...',
	  'userId': 'd46b54a8-3388-46b3-9b68...',
	  'gpsPositions': [
	    {
	      'altitudeInMeters': 100,
	      'dateCreated': 1578609025731,
	
	      'gpsFix': 'GPS_3D',
	      'heading': 0,
	      'horizontalUncertaintyInMeters': 5,
	
	      'latitude': 30.37571,
	      'longitude': -85.80549,
	      'numSatellites': 3,
	      'speedInKmph': 79,
	      'stale': False,
	      'verticalUncertaintyInMeters': 0
	    }
	  ]
	}</pre>

The `GetGpsPosition` call returns all the GPS position objects in the system. Unfortunately, this will always be a very large number of objects. Consequently, you can filter the results of the call in the following ways:

- Using the `-filters-property`, `-filters-value`, and `-filters-operator` options. For example, you could filter to get just the objects whose `deviceId` is equal to the UUID of a specific device:

  ```
  "filters" : [ {
  "property" : "deviceId",
  "value" : "5d4b40a1-8415-472e-a380-3324e4c887d2",
  "operator" : "EQUALS"
  } ]
  ```

- Specifying the `deviceId` that you're interested in:

  ```
  "deviceIds":
  ["2a44cbe5-f995-4e4e-b2dd-018049ffc966","346f9d45-61c9-409d-b955-34298fdfb1e7"
  ],
  ```

- Specifying the `userId` that you're interested in:

  ```
  "userIds":
  ["dedb90d7-0ef6-49fa-9009-07cece9104ab","d46b54a8-3388-46b3-9b68-f5c2f22fcfb3"
  ],
  ```

- Specifying the `emergencyId` that you're interested in:

  ```
  "emergencyIds": ["73fb509e-14fc-44f2-9265-04716352a660"],
  ```

Here is a CLI example for getting GPS positions for one specific device:

```
D:\CLI\> runRbCli.bat GetGpsPositions -server dev3.riskband.ws -user acme\admin
-pwd password123 -computeResultCount true -filters-property deviceId
-filters-value 346f9d45-61c9-409d-b955-34298fdfb1e7 -filters-operator EQUALS
-prettyJson
{
  "results" : [ {
    "stale" : false,
    "dateRegistered" : 1579126180440,
    "deviceId" : "346f9d45-61c9-409d-b955-34298fdfb1e7",
    "userId" : "dedb90d7-0ef6-49fa-9009-07cece9104ab",
    "dateCreated" : 1579126177492,
    "gpsFix" : "GPS_3D",
    "emergencyId" : null,
    "numSatellites" : 3,
    "latestForUser" : false,
    "longitude" : -85.80549,
    "latitude" : 30.24356,
    "heading" : 0,
    "speedInKmph" : 79,
    "horizontalUncertaintyInMeters" : 5,
    "verticalUncertaintyInMeters" : 0,
    "altitudeInMeters" : 100,
    "latestForDevice" : false,
    "id" : "e076805c-e34c-483c-9e82-a74b8225ed25"
  }, {
    "stale" : false,
    "dateRegistered" : 1579126219720,
    "deviceId" : "346f9d45-61c9-409d-b955-34298fdfb1e7",
    "userId" : "dedb90d7-0ef6-49fa-9009-07cece9104ab",
    "dateCreated" : 1579126216874,
    "gpsFix" : "GPS_3D",
    "emergencyId" : null,
    "numSatellites" : 3,
    "latestForUser" : true,
    "longitude" : -85.80549,
    "latitude" : 30.17815,
    "heading" : 0,
    "speedInKmph" : 79,
    "horizontalUncertaintyInMeters" : 5,
    "verticalUncertaintyInMeters" : 0,
    "altitudeInMeters" : 100,
    "latestForDevice" : true,
    "id" : "36f665cc-2262-493a-8ec5-7dd5541d790c"
  } ],
  "correlationId" : "66A10-1-1",
  "resultCount" : 2
}
```

If you are using cURL, here is a 691-character request payload for getting GPS positions for just two specified users:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "limit": 1000,
  "offset": 0,
  "sortOrder": null,
  "filters": null,
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "maxHorizontalUncertainty": 75,
  "any": false,
  "userIds": null,
  "maxAgeInHours": 9,
  "emergencyIds": null,
  "deviceIds": null,
  "userIds":
["dedb90d7-0ef6-49fa-9009-07cece9104ab","d46b54a8-3388-46b3-9b68-f5c2f22fcfb3"],
  "emergencyIds": null,
  "maxAgeInHours": 1
}
```

And here is an example of a 672-character request payload for getting just the GPS locations associated with a specific emergency:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "limit": 1000,
  "offset": 0,
  "sortOrder": null,
  "filters": null,
  "any": true,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "maxHorizontalUncertainty": 75,
  "any": false,
  "deviceIds": null,
  "userIds": null,
  "maxAgeInHours": 9,
  "emergencyIds": null,
  "deviceIds": null,
  "userIds": null,
  "emergencyIds": ["73fb509e-14fc-44f2-9265-04716352a660"],
  "maxAgeInHours": 1
}
```

Getting Emergency Artifact Objects

For every emergency artifact stored in the RiskBand ARIES Manager, there is a corresponding emergency artifact object that contains information, or metadata, about the artifact. You can get a list of all the emergency artifact objects in the system using the `GetEmergencyArtifacts` call. The Java CLI would look something like this:

```
D:\CLI> runRbCli.bat GetEmergencyArtifacts -server dev3.riskband.ws -user
acme\admin -pwd password123 -computeResultCount true
```

The results payload will contain a number of embedded emergency artifact objects that look something like this:


```
{
  "name": "1578608041.447.jpeg",
  "type": "PHOTO",
  "dateCreated": 1578608041447,
  "dateRegistered": 1578608055473,
  "gpsPosition": {
    "stale": false,
    "userId": "d46b54a8-3388-46b3-9b68-f5c2f22fcfb3",
    "dateCreated": 1578608042214,
    "deviceId": "3808c804-9094-4a43-b662-a93b04ca29c5",
    "dateRegistered": 1578608044129,
    "speedInKmph": 79,
    "latitude": 30.18134,
    "latestForUser": false,
    "numSatellites": 3,
    "gpsFix": "GPS_3D",
    "longitude": -85.80549,
    "emergencyId": "37925e3e-1b7f-4e95-ad20-3dc5f62a7b13",
    "heading": 0,
    "horizontalUncertaintyInMeters": 5,
    "verticalUncertaintyInMeters": 0,
    "latestForDevice": false,
    "altitudeInMeters": 100,
    "id": "e8112a4c-9734-46bf-b047-2afac9301fe1"
  },
  "gpsPositionId": "e8112a4c-9734-46bf-b047-2afac9301fe1",
  "emergencyId": "37925e3e-1b7f-4e95-ad20-3dc5f62a7b13",
  "downloadLength": 29410,
  "id": "000cb978-0ae1-41bf-9241-0cb872b42f2e"
}
```


If you are using cURL, a 410-character JSON request payload for the call would look something like this:


```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "any": false,
  "computeResultCount": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "limit" : 1000,
  "offset" : 0,
  "sortOrder" : null
}
```

If you know the UUID of an emergency you can get a count of the number of artifacts associated with that emergency using the `GetEmergencyArtifactsCount`. The CLI command would look something like this:

```
D:\CLI> runRbCli.bat GetEmergencyArtifactCounts -server dev3.riskband.ws -user
acme\admin -pwd password123 -emergencyIds "{37925e3e-1b7f-4e95-ad20-3dc5f62a7b13}"
-prettyJson
{
  "correlationId" : "A404A-1-1",
  "counts" : [ {
    "count" : 30,
    "emergencyId" : "37925e3e-1b7f-4e95-ad20-3dc5f62a7b13"
  } ]
}
```

Downloading Emergency Artifacts

From the contents of an emergency artifact object you can determine the “type” and the “download length” of an artifact.

Downloading Emergency Artifacts with cURL

If, for example, the artifact is a photo (as shown in the sample emergency artifact object above) you can use cURL to download it using the `DownloadEmergencyArtifact` call with a 442-character JSON request payload that looks something like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "filter": {
    "property": "id",
    "value": "000cb978-0ae1-41bf-9241-0cb872b42f2e"
  },
  "byteOffset": 0,
  "byteLength": 29410
}
```

The curl command you would use is slightly different than that used in other examples in this document. The `curl` command does not use the `-i` option, as the header information is not wanted, and it uses the `--output` command to specify that the output (in this case the specified photo) should be output to a file.

```
curl -v -X PUT -H "Content-Length: 442" -H "Content-Type: application/json" -d
@downloadEmergencyArtifact_442.json
https://dev3.riskband.ws?DownloadEmergencyArtifact=442U --output file2.jpg
```

Downloading Emergency Artifacts with the Java CLI

If you know the name of the artifact you want to download, you can use the CLI to download the artifact. Here is an example of how to do so using emergency credentials:

```
runRbCli.bat DownloadEmergencyArtifact -server demo.riskband.ws -authId
9ab398cd-a1f3-4036-bdb8-844834f92ec6 -authKey
JJU2A7ypB/41hXWj9zgKd2afzmwbhZmMq9bCLObcJlA= -authType EMERGENCY
-filter-property name -filter-value 1625165668.500.jpeg >
downloaded_1625165668.500.jpeg
```

Closing an Emergency

You can close an active emergency using the `CloseEmergencyByUser` call. To make this call you need the UUID of the emergency, and you must specify a reason or comment regarding why the emergency is being closed.

Here is an example of closing an emergency using the Java CLI:

```
D:\CLI> runRbCli.bat CloseEmergencyByUser -server dev3.riskband.ws -user
acme\admin -pwd password123 -id 78350923-48e9-4e11-800e-ee765ec2b60e -note "closed
from API" -prettyJson

{
  "result" : {
    "dateRegistered" : 1579097186014,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "id" : "78350923-48e9-4e11-800e-ee765ec2b60e"
  },
  "correlationId" : "7CF4D-1-1"
}
```

If you are using cURL, here is an example of a 401-character request payload to close an emergency:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "note" : "closed from API using curl",
  "id" : "e31e6e45-c438-40f1-8ac9-7490be622270"
}
```

Adding Additional Emergency Notes

You can add additional notes to an emergency using the `CreateEmergencyNote` call. An example of that call using the Java CLI would look something like this:

```
D:\CLI> runRbCli.bat CreateEmergencyNote -server dev3.riskband.ws -user acme\admin
-pwd password123 -emergencyId 78350923-48e9-4e11-800e-ee765ec2b60e -note "2nd
comment added from Java CLI" -prettyJson
{
  "result" : {
    "note" : "2nd comment added from Java CLI",
    "emergencyId" : "78350923-48e9-4e11-800e-ee765ec2b60e",
    "dateRegistered" : 1579105393027,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "id" : "338e0974-707e-43a0-aa80-be24ff983535"
  },
  "correlationId" : "E71A6-1-1"
}
```

Getting Emergency Notes

You can view all the notes associated with all emergencies using the `GetEmergencyNotes` call. Using the CLI the call might look something like this:

```
D:\CLI> runRbCli.bat GetEmergencyNotes -server dev3.riskband.ws -user acme\admin
-pwd password123 -computeResultCount true -prettyJson
{
  "results" : [ {
    "dateRegistered" : 1578609062761,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "207a64be-5056-4039-b1c4-624e983046b9",
    "note" : "closed",
    "id" : "08f0bf96-b8be-4082-821f-0b6bc9c37182"
  }, {
    "dateRegistered" : 1579103206143,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "3rd Note",
    "id" : "3f13421b-62f7-4c61-b050-f5f8661c00a8"
  }, {
    "dateRegistered" : 1579097186032,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "emergencyId" : "78350923-48e9-4e11-800e-ee765ec2b60e",
    "note" : "closed from API",
    "id" : "9906f81c-b490-4ec7-896f-842ccd41703e"
  }, {
    "dateRegistered" : 1579097926570,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "closed from API using curl",
    "id" : "a0f666a6-bd25-42d8-bd89-283a28724591"
  }, {
    "dateRegistered" : 1579103199654,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "2nd Note",
    "id" : "ae257c7c-6a23-4073-8bd6-6c307cac80f4"
  }, {
    "dateRegistered" : 1578608070779,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "37925e3e-1b7f-4e95-ad20-3dc5f62a7b13",
    "note" : "closed",
    "id" : "c6815e46-8615-407b-a668-c25ad2b6f230"
  } ],
  "correlationId" : "9BB27-1-1",
  "resultCount" : 6
}
```

Using the `-sortOrder-property` option you can also group the notes by emergency ID:

```
D:\CLI> runRbCli.bat GetEmergencyNotes -server dev3.riskband.ws -user acme\admin
-pwd password123 -computeResultCount true -sortOrder-property emergencyId
-prettyJson
{
  "results" : [ {
    "dateRegistered" : 1578609062761,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "207a64be-5056-4039-b1c4-624e983046b9",
    "note" : "closed",
    "id" : "08f0bf96-b8be-4082-821f-0b6bc9c37182"
  }, {
    "dateRegistered" : 1578608070779,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "37925e3e-1b7f-4e95-ad20-3dc5f62a7b13",
    "note" : "closed",
    "id" : "c6815e46-8615-407b-a668-c25ad2b6f230"
  }, {
    "dateRegistered" : 1579097186032,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "emergencyId" : "78350923-48e9-4e11-800e-ee765ec2b60e",
    "note" : "closed from API",
    "id" : "9906f81c-b490-4ec7-896f-842ccd41703e"
  }, {
    "dateRegistered" : 1579103206143,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "3rd Note",
    "id" : "3f13421b-62f7-4c61-b050-f5f8661c00a8"
  }, {
    "dateRegistered" : 1579097926570,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "closed from API using curl",
    "id" : "a0f666a6-bd25-42d8-bd89-283a28724591"
  }, {
    "dateRegistered" : 1579103199654,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "2nd Note",
    "id" : "ae257c7c-6a23-4073-8bd6-6c307cac80f4"
  } ],
  "correlationId" : "A2636-1-1",
  "resultCount" : 6
}
```

Alternatively, you can filter the call to just return the notes from a specific emergency and sort the notes in the order they were entered:

```
D:\CLI> runRbCli.bat GetEmergencyNotes -server dev3.riskband.ws -user acme\admin
-pwd password123 -computeResultCount true -filters-property emergencyId
-filters-value e31e6e45-c438-40f1-8ac9-7490be622270 -filters-operator EQUALS
-sortOrder-property dateRegistered -prettyJson
{
  "results" : [ {
    "dateRegistered" : 1579097926570,
    "userId" : "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "closed from API using curl",
    "id" : "a0f666a6-bd25-42d8-bd89-283a28724591"
  }, {
    "dateRegistered" : 1579103199654,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "2nd Note",
    "id" : "ae257c7c-6a23-4073-8bd6-6c307cac80f4"
  }, {
    "dateRegistered" : 1579103206143,
    "userId" : "1f7d2b40-7c3d-49aa-bae5-6a9a1ec6cca1",
    "emergencyId" : "e31e6e45-c438-40f1-8ac9-7490be622270",
    "note" : "3rd Note",
    "id" : "3f13421b-62f7-4c61-b050-f5f8661c00a8"
  } ],
  "correlationId" : "C6AEC-1-1",
  "resultCount" : 3
}
```

If you want to make the call in cURL, a 671-character request payload for the call to limit output to a specific emergency and to sort the output by date might look like this:

```
{
  "authorization": {
    "computerId": "00000000-0000-0000-0000-000000012345",
    "entityType": "USER",
    "id": "1420cfd4-a05e-4bd2-a755-a3026371fca1",
    "mfaCode": null,
    "passwordHash": "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="
  },
  "limit": 1000,
  "filters": [
    {
      "operator": "EQUALS",
      "property": "emergencyId",
      "value": "e31e6e45-c438-40f1-8ac9-7490be622270"
    }
  ],
  "offset": 0,
  "sortOrder": [
    {
      "property": "dateRegistered",
      "ascending": true
    }
  ],
  "any": true,
  "correlationId": "RANDOM-0-002",
  "disableResponseCompression": true,
  "note": "closed from API using curl",
  "id": "e31e6e45-c438-40f1-8ac9-7490be622270"
}
```

A RiskBand API Syntax Examples

The following section provides examples of syntax for RESTful API calls supported by the RiskBand ARIES Manager.

ActivateManufactureInProgressDevice

There is no response to the ActivateManufactureInProgressDevice API call. However, the call is synchronous, and if it returns with an ok HTTP code, then the device is active at the time the API call returns.

```
D:\CLI\20254> runRbCli.bat ManufactureDevice -server test2.riskband.ws -user root
-pwd password123 -authType USER -computerId 50c2ef54-d65a-4313-b7a4-505e4b13401f
-cellularCarrier TELIT -firmwareBuildId b8057dc5-6740-43bb-b1d5-49fb0bfcd9b6
-iccid 89014104256474424908 -imei 12345612345699 -modelId
5b3495ba-36a9-49bd-919e-5edeea01adc9 -serialNumber 200710-00001 -prettyJson |
findstr id

    "iccid" : "89014104256474424908",
    "id" : "770c2eb1-e575-4002-9708-b691b0bdbf1b"

D:\CLI\20254> runRbCli.bat ActivateManufactureInProgressDevice -server
test2.riskband.ws -user root -pwd password123 -authType USER -computerId
5b3495ba-d65a-4313-b7a4-505e4b13401f -filter-property id -filter-value
770c2eb1-e575-4002-9708-b691b0bdbf1b -prettyJson
```

CloseEmergencyByEmergencyResponder

As an emergency responder you can close an emergency with the CloseEmergencyByEmergencyResponder API call using emergency credentials. This is a synchronous call; the response is not sent from the server until the emergency has been closed.

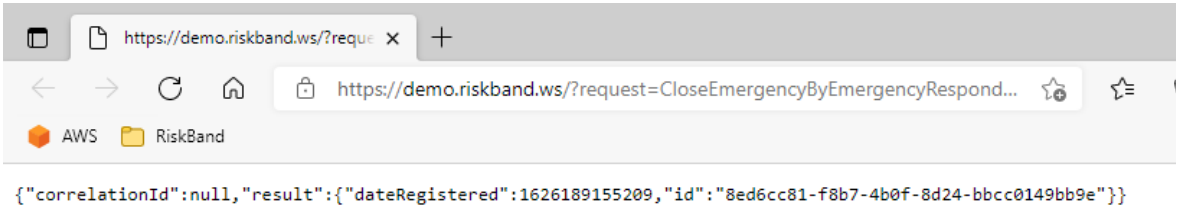
The emergency credentials are provided in the emergency notification payload that is sent to the emergency providers endpoint. Here is an example using the CLI:

```
D:\CLI\> runRbCli.bat CloseEmergencyByEmergencyResponder -server demo2.riskband.ws
-authId 0e31799c-09a0-42dc-8f38-228c4e425b86 -authKey
A5x/n2bINKm8iYrU+wU5IAwOl1bjbu8Pf5/tQt+uNW4= -authType EMERGENCY -id
0e31799c-09a0-42dc-8f38-228c4e425b86 -note "Closed by Emergency Responder with
Emergency Credentials" -prettyJson

{
  "result" : {
    "dateRegistered" : 1626127668097,
    "id" : "0e31799c-09a0-42dc-8f38-228c4e425b86"
  },
  "correlationId" : "C2E66-1-1"
}
```

Here is an example using the address bar in a brower:

```
https://demo.riskband.ws/?request=CloseEmergencyByEmergencyResponder&authorizatio
n-entityType=EMERGENCY&authorization-id=8ed6cc81-f8b7-4b0f-8d24-bbcc0149bb9e&auth
orization-passwordHash=yumoKzLhoOz2kmHwi5kcG81HpleLWfR4AB4F3GJTzBw%3D&id=8ed6cc81
-f8b7-4b0f-8d24-bbcc0149bb9e&note=Closed%20by%20Emergency%20Responder
```



CloseEmergencyByUser

For Java CLI and cURL examples, see Adding Additional Emergency Notes on page 124.

This is a synchronous call; the response is not sent from the server until the emergency has been closed.

CreateActionMessage

Action messages can be sent to devices as EMERGENCY, URGENT, or NON_URGENT. With Emergency and Urgent messages, a text message is sent the device telling it to call home and retrieve the message. With non-urgent, messages the device retrieves the message at the next scheduled call home interval.

```
D:\CLI\>runRbCli.bat CreateActionMessage -server test.riskband.ws -user root -pwd
password -type NON_URGENT -userId 0ea90f62-c07b-4fea-ac1c-8cfb31ac3f02 -message
"Update service logs at end of shift." -prettyJson
{
  "result" : {
    "message" : "Update service logs at end of shift.",
    "type" : "NON_URGENT",
    "dateRegistered" : 1630423393350,
    "userId" : "0ea90f62-c07b-4fea-ac1c-8cfb31ac3f02",
    "dateClosed" : null,
    "id" : "e70558c0-7b2b-4d28-a7af-57410f6aa9b7"
  },
  "correlationId" : "RB-CLI-3DZ6D-1-1"
}
```

CreateAdministratorPermission

Here is an example of using the CLI to grant permission to a system administrator:

```
D:\CLI\21680> runRbCli.bat CreateAdministratorPermission -server test2.riskband.ws
-user admin -pwd password123 -userId 7692a4f3-e383-4fc8-9260-4c18d0de81bb
-permission "GUI"

{"result":{"permission":"GUI","userId":"7692a4f3-e383-4fc8-9260-4c18d0de81bb","id
":"af93f16a-2500-4336-9eda-f5656b0037ca"},"correlationId":"9CF25-1-1"}
```

CreateDeviceModel

Here is a CLI example for creating a device model named “RB5xp”

```
D:\21680> runRbCli.bat CreateDeviceModel -server test2.riskband.ws -user admin
-pwd password123 -authType USER -name "RB5xp" -prettyJson

{
  "result" : {
    "name" : "RB5xp",
    "description" : null,
    "dateRegistered" : 1626212021721,
    "id" : "15fff400-d1a7-4272-80ae-a38d14996a93"
  },
  "correlationId" : "29D63-1-1"
}
```

CreateDeviceSecurityPolicy

Here is an example of the CLI command:

```
D:\CLI>runRbCli.bat CreateDeviceSecurityPolicy -server test2.riskband.ws -user
admin -pwd password123 -name "Device Security Policy A" -description "Security
Policy Created by CLI" -allowAirplaneMode TRUE -allowDemoMode TRUE -allowPowerOff
TRUE -deviceAuthorizationRotationFrequencyInHours 24 -prettyJson
{
  "result" : {
    "name" : "Device Security Policy A",
    "allowAirplaneMode" : true,
    "organizationId" : null,
    "description" : "Security Policy Created by CLI",
    "allowDemoMode" : true,
    "allowPowerOff" : true,
    "deviceAuthorizationRotationFrequencyInHours" : 24,
    "id" : "8d68a26a-6971-40d3-a7dd-af68f2b823cb"
  },
  "correlationId" : "68988-1-1"
}
```

Here is an example of the CLI command called from a batch file:

```
@echo off

SET ARESSERVER=test2.riskband.ws
SET SYSADMINNAME=root
SET SYSADMINPWD=password123

SET SECPOLNAME="Device Security Policy A"
SET SECPOLDESC="Security Policy Created by CLI"

echo on
call runRbCli.bat CreateDeviceSecurityPolicy^
 -server %ARESSERVER%^
 -user %SYSADMINNAME%^
 -pwd %SYSADMINPWD%^
 -name %SECPOLNAME%^
 -description %SECPOLDESC%^
 -allowAirplaneMode TRUE^
 -allowDemoMode TRUE^
 -allowPowerOff TRUE^
 -deviceAuthorizationRotationFrequencyInHours 24^
 -prettyJson
```

CreateEmergencyNote

For Java CLI example, see Adding Additional Emergency Notes on page 124.

CreateEmergencyResponsePolicy

Here is an example of creating an Emergency Response Policy with minimum inputs. Unspecified options are set to default values.

```
D:\CLI> runRbCli.bat CreateEmergencyResponsePolicy -server test2.riskband.ws -user
acme\admin -pwd bfuQMs7FUKtgpBSP8j3aRDwMVKaHm9TQY3YELMvv -organizationId
"a86e9f8f-82a4-4501-8c7b-b3f54250b027" -name "Emergency Response Policy H"
-handledByGeos NEVER -acceptIncomingCalls FALSE -prettyJson

{
  "result" : {
    "name" : "acme\\Emergency Response Policy H",
    "maxMinsOfAudioPriorToEmergencyTriggeredToUpload" : 5,
    "maxMinsOfPhotosPriorToEmergencyTriggeredToUpload" : 5,
    "organizationId" : "a86e9f8f-82a4-4501-8c7b-b3f54250b027",
    "enableUiMasking" : false,
    "notificationLevel" : "EMERGENCY_EVENTS",
    "enableAlwaysOnPhotoCapture" : true,
    "selfCancellationAllowedInSecs" : null,
    "enableAlwaysOnAudioRecording" : true,
    "requiredCallerIdSubstring" : null,
    "manDownEnabled" : false,
    "deviceHapticIndication" : "DISCREET",
    "manDownSecsRequired" : 600,
    "notificationEndpoint" : null,
    "impactSensitivity" : null,
    "dialOutPhoneNumber" : null,
    "dialOutTrigger" : "UPON_EMERGENCY_TRIGGERED",
    "acceptIncomingCalls" : false,
    "phoneTimeoutInSecs" : 0,
    "speakerVolumeLevel" : 85,
    "dialOutWithCallerId" : false,
    "disableAudioNotifications" : false,
    "deviceVisualIndication" : "DISCREET",
    "disableUiFlashing" : false,
    "minEmergencyRetentionInDays" : null,
    "photoOptimization" : "MAXIMUM_UPLOAD_SPEED",
    "phase1DurationInSecs" : 90,
    "phase2DurationInSecs" : 300,
    "numPhotosToCaptureRapidly" : 20,
    "maxEmergencyRetentionInDays" : null,
    "voiceOnlyFailbackTimeoutInSecs" : 240,
    "impactDelayToTriggerEmergencyInSecs" : 300,
    "secondsBetweenAlwaysOnPhotoCapture" : 3,
    "closedEmergencyProviderAccessInDays" : 90,
    "manDownDelayToTriggerEmergencyInSecs" : 300,
    "phase3GpsCallHomeFrequencyInSecs" : 300,
    "phase1PhotoCaptureFrequencyInSecs" : 3,
    "phase1GpsCallHomeFrequencyInSecs" : 6,
    "minsOfAudioRecordingAfterEmergencyTriggered" : 0,
    "phase2GpsCallHomeFrequencyInSecs" : 30,
    "phase2PhotoCaptureFrequencyInSecs" : 9,
    "phase4GpsCallHomeFrequencyInSecs" : 1800,
    "phase3PhotoCaptureFrequencyInSecs" : 90,
    "description" : null,
    "handledByGeos" : "NEVER",
    "impactEnabled" : false,
    "id" : "004b0a69-e142-4588-b142-7a07008ec2e5"
  },
```

```
  "correlationId" : "67B75-1-1"
}
```

Here is an example with all the options specified:

```
D:\CLI>runRbCli.bat CreateEmergencyResponsePolicy -server test2.riskband.ws -user
acme\admin -pwd bfuQMs7FUKtgpBSP8j3aRDwMVKaHm9TQY3YELMvv -organizationId
"a86e9f8f-82a4-4501-8c7b-b3f54250b027" -name "Emergency Response Policy K"
-description "Emergency Response Policy created by CLI" -handledByGeos NEVER
-deviceHapticIndication DISCREET -deviceVisualIndication DISCREET
-selfCancellationAllowedInSecs 120 -minEmergencyRetentionInDays 365
-maxEmergencyRetentionInDays 730 -dialOutPhoneNumber "555-555-5555"
-dialOutWithCallerId TRUE -dialOutTrigger UPON_EMERGENCY_TRIGGERED
-acceptIncomingCalls FALSE -requiredCallerIdSubstring "Security Office"
-speakerVolumeLevel 85 -phoneTimeoutInSecs 0 -voiceOnlyFailbackTimeoutInSecs 240
-phase1DurationInSecs 90 -phase2DurationInSecs 300 -photoOptimization
MAXIMUM_UPLOAD_SPEED -numPhotosToCaptureRapidly 20
-phase1PhotoCaptureFrequencyInSecs 3 -phase2PhotoCaptureFrequencyInSecs 9
-phase3PhotoCaptureFrequencyInSecs 90 -phase1GpsCallHomeFrequencyInSecs 6
-phase2GpsCallHomeFrequencyInSecs 30 -phase3GpsCallHomeFrequencyInSecs 300
-phase4GpsCallHomeFrequencyInSecs 1800 -closedEmergencyProviderAccessInDays 90
-notificationEndpoint "https://docs.riskband.com" -notificationLevel ALL_EVENTS
-manDownEnabled TRUE -manDownSecsRequired 600
-manDownDelayToTriggerEmergencyInSecs 300 -impactEnabled TRUE -impactSensitivity
HIGH -impactDelayToTriggerEmergencyInSecs 300 -disableUiFlashing FALSE
-disableAudioNotifications FALSE -enableUiMasking FALSE -prettyJson

{
  "result" : {
    "name" : "acme\\Emergency Response Policy K",
    "maxMinsOfPhotosPriorToEmergencyTriggeredToUpload" : 5,
    "maxMinsOfAudioPriorToEmergencyTriggeredToUpload" : 5,
    "phoneTimeoutInSecs" : 0,
    "dialOutPhoneNumber" : "555-555-5555",
    "dialOutTrigger" : "UPON_EMERGENCY_TRIGGERED",
    "acceptIncomingCalls" : false,
    "dialOutWithCallerId" : true,
    "requiredCallerIdSubstring" : "Security Office",
    "phase1DurationInSecs" : 90,
    "selfCancellationAllowedInSecs" : 120,
    "phase2DurationInSecs" : 300,
    "photoOptimization" : "MAXIMUM_UPLOAD_SPEED",
    "manDownEnabled" : true,
    "numPhotosToCaptureRapidly" : 20,
    "enableUiMasking" : false,
    "impactSensitivity" : "HIGH",
    "enableAlwaysOnAudioRecording" : true,
    "speakerVolumeLevel" : 85,
    "deviceHapticIndication" : "DISCREET",
    "disableAudioNotifications" : false,
    "notificationLevel" : "ALL_EVENTS",
    "notificationEndpoint" : "https://docs.riskband.com",
    "deviceVisualIndication" : "DISCREET",
    "disableUiFlashing" : false,
    "manDownSecsRequired" : 600,
    "enableAlwaysOnPhotoCapture" : true,
    "minEmergencyRetentionInDays" : 365,
    "organizationId" : "a86e9f8f-82a4-4501-8c7b-b3f54250b027",
    "maxEmergencyRetentionInDays" : 730,
    "phase2PhotoCaptureFrequencyInSecs" : 9,
    "minsOfAudioRecordingAfterEmergencyTriggered" : 0,
    "phase4GpsCallHomeFrequencyInSecs" : 1800,
    "phase1GpsCallHomeFrequencyInSecs" : 6,
    "phase3GpsCallHomeFrequencyInSecs" : 300,
    "phase2GpsCallHomeFrequencyInSecs" : 30,
    "phase3PhotoCaptureFrequencyInSecs" : 90,
```

```
    "phase1PhotoCaptureFrequencyInSecs" : 3,
    "handledByGeos" : "NEVER",
    "description" : "Emergency Response Policy created by CLI",
    "impactEnabled" : true,
    "impactDelayToTriggerEmergencyInSecs" : 300,
    "secondsBetweenAlwaysOnPhotoCapture" : 3,
    "voiceOnlyFailbackTimeoutInSecs" : 240,
    "manDownDelayToTriggerEmergencyInSecs" : 300,
    "closedEmergencyProviderAccessInDays" : 90,
    "id" : "9a8f3a99-e99b-4d82-8f60-7a271f24352a"
  },
  "correlationId" : "77439-1-1"
}
```

CreateGeoFence

For browser example, see Calling CreateGeoFence on page 56.

For cURL example, see Calling CreateGeoFence on page 63.

for PowerShell example, see Calling CreateGeoFence on page 70.

For Java CLI, see Calling CreateGeoFence on page 83.

For Java SDK, see Calling CreateGeoFence on page 86.

CreateOrganizationPermissions

For Java CLI example see :: CreateOrganizationPermission on page 152.

CreateUser

For Java CLI example see :: CreateUser on page 152.

DeleteUser

Here is a CLI example of how to delete a user:

```
D:\21680> runRbCli.bat DeleteUser -server test2.riskband.ws -user admin -pwd
password123 -id 7692a4f3-e383-4fc8-9260-4c18d0de81bb

{"correlationId":"BEE4E-1-1"}
```

DownloadEmergencyArtifact

For CLI and cURL examples, see Downloading Emergency Artifacts on page 122.

DownloadGuiBuild

Here is a CLI example of how to download a specific GUI build from the server:

```
runRbCli.bat DownloadGuiBuild -server demo.riskband.ws -user admin -pwd
password123 -filter-property buildNumber -filter-value 21680 > down21680Gui_b.jar
```

GetAdministratorPermissions

Here is an example of using the CLI to get the permissions assigned to an administrator:

```
D:\CLI\21680> runRbCli.bat GetAdministratorPermissions -server test2.riskband.ws
-user admin -pwd password123 -computeResultCount TRUE -filters-operator MATCHES
-filters-property userId -filters-value "6d2694c0-de27-4f0f-ba97-de416221541f"
-prettyJson

{
  "results" : [ {
    "permission" : "OWNER",
    "userId" : "6d2694c0-de27-4f0f-ba97-de416221541f",
    "id" : "ebe825d0-37be-4efd-9ace-668cd226857b"
  }, {
    "permission" : "ORGANIZATION_ADMIN_POLICY",
    "userId" : "6d2694c0-de27-4f0f-ba97-de416221541f",
    "id" : "a2989cd6-c067-4a25-af1c-4eb9203c5a51"
  }, {
    "permission" : "POLICY",
    "userId" : "6d2694c0-de27-4f0f-ba97-de416221541f",
    "id" : "c4fedaa3-05f9-4501-80c6-03cee3ea3832"
  },
  .
  .
  .
     {
    "permission" : "PARTNER_OWNER",
    "userId" : "6d2694c0-de27-4f0f-ba97-de416221541f",
    "id" : "9d1018dc-b6b0-40eb-8ca1-f5bf22709da5"
  } ],
  "correlationId" : "79607-1-1",
  "resultCount" : 29
}
```

GetDevices

Here is an example of how to get all the devices that have been manufactured on an ARIES instance:

```
C:\CLI\> runRbCli.bat GetDevices -user admin -pwd password123 -prettyJson
-computeResultCount true -server test2.riskband.ws

{
  "results" : [ {
    "name" : "nadir-1",
    "passwordHash" : "<concealed>",
    "dateLastHeartbeat" : 0,
    "dateOrganizationRegistered" : 1621291712843,
    "dateRegistered" : 1620943116553,
    "organizationId" : "607291fb-1f77-403e-9698-242d98c1368e",
    "registeredWithGeosE911" : false,
    "cellularCarrierId" : "<concealed>",
    "divisionId" : null,
    "groupId" : null,
    "phoneNumber" : "<concealed>",
    "manufacturer" : "c0e94b3f-cb9e-49fb-a45c-497c8168ff0b",
    "inAirplaneMode" : false,
    "serialNumber" : "200504-00013",
    "rxBytes" : 0,
    "txBytes" : 0,
    "reportedAsNeedingRecharge" : false,
    "securityPolicyId" : "50e2e058-d1cb-4609-981c-e0c5ae162b44",
    "bootloaderBuildNumber" : null,
```

```
    "lastAuthorizationRotation" : 1620943116553,
    "registrationCode" : null,
    "firmwareBuildId" : "e766e052-e981-4333-b6f2-1e42657a2bae",
    "veryLateCallingHome" : false,
    "forceFirmwareBuildId" : null,
    "firmwarePolicyId" : "497692f0-1c54-421b-a16f-8ef9bfa9cf33",
    "cellularCarrier" : "SIMULATED",
    "gpsCallHomeRequired" : false,
    "powerManagementPolicyId" : "424807c7-1fae-477c-a0f5-1e3fca9363d7",
    "fullyManufactured" : true,
    "registrationPassword" : null,
    "logCallHomeRequired" : null,
    "arrayDebugModules" : [ ],
    "locationStalenessInMins" : null,
    "locationCertainty" : null,
    "lastCellSignalQuality" : null,
    "firmwareUpgradeBuildId" : null,
    "callInProgress" : null,
    "lastBatteryLevel" : 1,
    "lastCellSignalStrength" : null,
    "policyDictatedFirmwareUpgradeId" : null,
    "modelId" : "77539469-2c85-4070-bd9c-5cfa696022b2",
    "uptimeInSecs" : 0,
    "userId" : null,
    "locationName" : null,
    "charging" : null,
    "iccid" : "555000003",
    "debugModules" : null,
    "cliEnabled" : false,
    "lastBatteryMv" : null,
    "imei" : "<concealed>",
    "id" : "056acb16-063a-49bd-8993-64bda181c65c"
  },
.
.
.
.    {
    "name" : "sym-5",
    "passwordHash" : "<concealed>",
    "dateLastHeartbeat" : 1621371176601,
    "dateOrganizationRegistered" : 1621369219816,
    "dateRegistered" : 1620943106650,
    "organizationId" : "bbab6647-1ef6-4897-9ecb-18c1f1a428ca",
    "registeredWithGeosE911" : false,
    "cellularCarrierId" : "<concealed>",
    "divisionId" : null,
    "groupId" : null,
    "phoneNumber" : "<concealed>",
    "manufacturer" : "c0e94b3f-cb9e-49fb-a45c-497c8168ff0b",
    "inAirplaneMode" : false,
    "serialNumber" : "210504-00011",
    "rxBytes" : 0,
    "txBytes" : 0,
    "reportedAsNeedingRecharge" : false,
    "securityPolicyId" : "50e2e058-d1cb-4609-981c-e0c5ae162b44",
    "bootloaderBuildNumber" : 0,
    "lastAuthorizationRotation" : 1620943106650,
    "registrationCode" : null,
    "firmwareBuildId" : "e766e052-e981-4333-b6f2-1e42657a2bae",
    "veryLateCallingHome" : true,
    "forceFirmwareBuildId" : null,
    "firmwarePolicyId" : "497692f0-1c54-421b-a16f-8ef9bfa9cf33",
    "cellularCarrier" : "ATT",
    "gpsCallHomeRequired" : false,
    "powerManagementPolicyId" : "b4c77f2d-63c8-4f46-aa14-cebe99cbf06a",
    "fullyManufactured" : true,
```

```
    "registrationPassword" : null,
    "logCallHomeRequired" : null,
    "arrayDebugModules" : [ ],
    "locationStalenessInMins" : 71897,
    "locationCertainty" : null,
    "lastCellSignalQuality" : null,
    "firmwareUpgradeBuildId" : null,
    "callInProgress" : "false",
    "lastBatteryLevel" : 60,
    "lastCellSignalStrength" : 50,
    "policyDictatedFirmwareUpgradeId" : null,
    "modelId" : "77539469-2c85-4070-bd9c-5cfa696022b2",
    "uptimeInSecs" : 252,
    "userId" : "2c40a883-8402-4afa-83fb-3cf60afec6bc",
    "locationName" : null,
    "charging" : true,
    "iccid" : "89011702272095202542",
    "debugModules" : null,
    "cliEnabled" : false,
    "lastBatteryMv" : 4000,
    "imei" : "<concealed>",
    "id" : "d716af5c-8aa9-4b7e-9a0c-34d4de3686ef"
  } ],
  "correlationId" : "D66E7-1-1",
  "resultCount" : 12
}
```


On Windows, you can filter out some of the output of this command using the *findstr* filter in conjuction with `-prettyJson`. For example, if you only want to see the device name, date of last heartbeat, and last battery level, you could enter something like this:

```
D:\\CLI\> runRbCli.bat GetDevices -server test2.riskband.ws -user admin -pwd
password123 -prettyJson | findstr "name lastBatteryLevel dateLastHeartbeat {"
{
  "results" : [ {
    "name" : "div1-8",
    "dateLastHeartbeat" : 1602542090511,
    "lastBatteryLevel" : 30,
  }, {
    "name" : "group1-3",
    "dateLastHeartbeat" : 1602541816012,
    "lastBatteryLevel" : 60,
  }, {
    "name" : "190921-00012",
    "dateLastHeartbeat" : 0,
    "lastBatteryLevel" : 1,
```

Remember dates in ARIES objects are in UNIX time format. For an example of how to programmatically convert a date form UNIX time to human time see Converting UNIX Time with Perl on page 153.

GetEmergencies

For browser example, see Calling GetEmergencies from a Browser on page 110.

For cURL example, see Calling GetEmergencies using cURL on page 111.

for PowerShell example, see Calling GetEmergencies from PowerShell on page 112.

For Java CLI, see Calling GetEmergencies using the Java CLI on page 111.

For Java SDK, see Calling GetEmergencies using the Java SDK on page 114.

CLI example as organization administrator, getting all emergencies:

```
runRbCli.bat GetEmergencies -server dev3.riskband.ws -authId
1420cfd4-a05e-4bd2-a755-a3026371fca1 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -computeResultCount
TRUE -prettyJson
```

CLI example as organization administrator, getting emergencies on a specific date:

```
runRbCli.bat GetEmergencies -server dev3.riskband.ws -filters-property
"dateRegistered" -filters-value 1576108060170 -filters-operator EQUALS -any
false -authId 1420cfd4-a05e-4bd2-a755-a3026371fca1 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -prettyJson
```

CLI example as organization administrator, getting active emergencies:

```
runRbCli.bat GetEmergencies -server dev3.riskband.ws -filters-property
"dateClosed" -filters-operator EQUALS -any false -authId
1420cfd4-a05e-4bd2-a755-a3026371fca1 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -prettyJson
```

Note: In the example above, the filter options would normally be `-filters-property "dateClosed" -filters-value NULL -filters-operator EQUALS`. However, specifying NULL on the command line puts quotation marks around the value in the request payload which causes the command to fail. The workaround is to remove the `-filters-value` option from the command line. In the CLI, options that are not specified, default to NULL.

GetEmergencyArtifacts

For Java CLI and cURL examples, see Getting Emergency Artifact Objects on page 121.

GetEmergencyArtifactsCount

For Java CLI and cURL examples, see Getting Emergency Artifact Objects on page 121.

GetEmergencyContactInformation

For Java CLI and cURL examples, see Getting Emergency Contact Information on page 116.

GetEmergencyContactInformations

Below is an example of how to get emergency contact information for the device user with the RiskBand CLI using emergency credentials.

If emergency contact information is defined, `EmergencyContactInformation` objects can be returned for the organization, the division, the group, and the user.

The *authId* and *authKey* values come from the emergency notifcation payload that RiskBand sends to the notification endpoint. The values are also available in the emergency photos URL.

```
D:\CLI\21680> runRbCli.bat GetEmergencyContactInformations -server
demo.riskband.ws -authId 9ab398cd-a1f3-4036-bdb8-844834f92ec6 -authKey
JJU2A7ypB/41hXWj9zgKd2afzmwbhZmMq9bCLObcJlA= -authType EMERGENCY
-computeResultCount true -prettyJson

{
  "correlationId" : "CBF2C-1-1",
  "results" : [ {
    "address" : null,
    "state" : null,
    "country" : null,
    "secondaryEmergencyContactCountry" : null,
    "primaryEmergencyContactCountry" : null,
    "secondaryEmergencyContactEmail" : null,
    "secondaryEmergencyContactPhone" : null,
    "city" : null,
    "zipCode" : null,
    "secondaryPhoneNumber" : null,
    "tertiaryPhoneNumber" : null,
    "secondaryEmailAddress" : null,
    "primaryEmergencyContactName" : null,
    "primaryEmergencyContactPhone" : null,
    "secondaryEmergencyContactName" : null,
    "primaryEmergencyContactEmail" : null,
    "id" : "032217fb-0f39-4ae2-b26e-0e845bd7d83e"
  }, {
    "address" : "356 CR 45",
    "state" : "NM",
    "country" : "USA",
    "secondaryEmergencyContactCountry" : "USA",
    "primaryEmergencyContactCountry" : "USA",
    "secondaryEmergencyContactEmail" : "mr.smythe.rumfelt@homesweethome.com",
    "secondaryEmergencyContactPhone" : "555-666-5557",
    "city" : "Grants",
    "zipCode" : "87020",
    "secondaryPhoneNumber" : "555-666-5555",
    "tertiaryPhoneNumber" : null,
    "secondaryEmailAddress" : null,
    "primaryEmergencyContactName" : "Mrs. Smythe-Rumfelt",
    "primaryEmergencyContactPhone" : "555-666-5554",
    "secondaryEmergencyContactName" : "Mr. Smythe-Rumfelt",
    "primaryEmergencyContactEmail" : "mrs.smythe.rumfelt@homesweethome.com",
    "id" : "cba37263-acfa-4b82-8696-e56ea7b21716"
  } ],
  "resultCount" : 2
}
```

GetEmergencyNotes

For Java CLI example, see Getting Emergency Notes on page 125.

GetGeoFences

For browser example, see Calling GetGeoFences on page 57.

For cURL example, see Calling GetGeoFences on page 64.

for PowerShell example, see Calling GetGeoFences on page 71.

For Java CLI example, see Calling GetGeoFences on page 84.

For Java SDK example , see Calling GetGeoFences on page 87.

GetGpsPositions

For Java CLI and cURL examples, see Getting Emergency GPS Coordinates on page 117.

```
runRbCli.bat GetGpsPositions -server dev3.riskband.ws -user root -pwd
rootPassword -computeResultCount true -filters-property deviceId
-filters-value 0e254cba-1ea2-47cc-ae6a-9bb5fbb61ae5 -filters-operator
EQUALS -prettyJson
```

GetOrganization

CLI example with user credentials

```
runRbCli.bat GetOrganization -server dev3.riskband.ws -filter-property name
-filter-value "Acme, Corp."  -authId 1420cfd4-a05e-4bd2-a755-a3026371fca1
-authKey vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER
-prettyJson
```

CLI example with emergency credentials

```
runRbCli.bat GetOrganization -server dev3.riskband.ws -filter-property name
-filter-value "Acme, Corp."  -authId bae4f969-83bf-4905-a844-04eba246d654
-authKey z7AZeZi9C5An+YhA6NePFyoTgnFwLwPkaS0vJmWZIHg= -authType EMERGENCY
-prettyJson
```

GetOrganizations

Here is a CLI example of using the GetOrganizations call:

```
runrbcli.bat GetOrganizations -server demo.riskband.ws -user admin -pwd
password123 -filters-property inProduction -filters-operator EQUALS -filters-value
false -prettyJson -computeResultCount true
```

Because a lot of data is returned, on the command line it can be useful to use the prettyJson option and filter the output. For example,

```
D:\CLI\21680> runrbcli.bat GetOrganizations -server demo.riskband.ws -user admin
-pwd password123 -filters-property inProduction -filters-operator EQUALS
-filters-value false -prettyJson -computeResultCount true  | findstr "name
numDevices numUsers resultCount"

    "name" : "3.x Pilot",
    "numUsers" : 23,
    "numDevices" : 0,
    "name" : "EngineeringDemo",
    "numUsers" : 2,
    "numDevices" : 2,
```

```
    "name" : "GEOS Closed Loop",
    "numUsers" : 3,
    "numDevices" : 1,
    "name" : "GEOS Open Loop",
    "numUsers" : 2,
    "numDevices" : 1,
    "name" : "Performance",
    "numUsers" : 3,
    "numDevices" : 0,
    "name" : "33org",
    "numUsers" : 3,
    "numDevices" : 0,
    "name" : "RiskBandDemo",
    "numUsers" : 19,
    "numDevices" : 15,
  "resultCount" : 7
```

GetOrganizationDivisions

For browser example, see Calling GetOrganizationDivisions on page 55.

For cURL example, see Calling GetOrganizationDivisions on page 61.

for PowerShell example, see Calling GetOrganizationDivisions on page 68.

For Java CLI, see Calling GetOrganizationDivisions on page 82.

For Java SDK, see Calling GetOrganizationDivisions on page 86.

GetOrganizationPermissions

For Java CLI example see :: GetOrganizationPermissions on page 152.

GetUser

Here are two examples of how to call GetUser using the Java CLI:

```
runRbCli.bat GetUser -server dev3.riskband.ws -authId
1420cfd4-a05e-4bd2-a755-a3026371fca1 -authKey
vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc= -authType USER -filter-property
name -filter-value acme\cchesterton -prettyJson

runRbCli.bat GetUser -server test2.riskband.ws -user sysadmin -pwd
DvjMVg4rAUtuRMSF9BSCWEJ63rWmuyv4JUN7js69 -filter-property name -filter-value
rogert -prettyJson | findstr "name emailAddress phoneNumber"

    "name" : "rogert",
    "phoneNumber" : "555.535.3454",
    "emailAddress" : "rogert@acme.com",
```

GetUsers

Here is an example of making the GetUsers call using the Java CLI on Windows:

```
runRbCli.bat GetUsers -server dev3.riskband.ws -user acme\admin -pwd
yxLS6ySSzpjJFjcL8EqeqYhrFqqy29CV3yjd95Qd -prettyJson
```

The result is a list of user objects. The user objects look like this:


```
{
    "name" : "acme\\bblack",
    "passwordHash" : "<concealed>",
    "dateLastPasswordChange" : 1594140700892,
    "passphraseAllClear" : "flotsam",
    "passphraseDuress" : "jestam",
    "lifeSavingInformation" : "Life Saving Notes bblack",
    "primaryPhotoId" : null,
    "restrictAccessToIps" : null,
    "organizationId" : "a75e866c-06b5-497b-8936-e861ef01ae49",
    "mfaCurrentCode" : 0,
    "dateRegistered" : 1583254297505,
    "geosFusionUserId" : null,
    "mfaCurrentDate" : 0,
    "mustChangePassword" : false,
    "dateLastActivity" : 1594140701541,
    "expiredPasswordChangeDeadlineDate" : null,
    "description" : "Notes",
    "partnerId" : null,
    "lastName" : "Black",
    "passwordType" : "SECRET_KEY",
    "divisionId" : null,
    "groupId" : null,
    "metadata" : null,
    "firstName" : "Byron",
    "emailAddress" : "b.black@acemeinc.com",
    "suspended" : false,
    "mfaRequired" : false,
    "phoneNumber" : "303-434-5555",
    "gender" : "MALE",
    "citizenship" : "USA",
    "emailVerified" : true,
    "dateOfBirth" : 260089200000,
    "phoneVerified" : true,
    "emergencyContactInformationId" : "4a3faceb-f41c-488a-887b-9e2e46821591",
    "passwordResetExpirationDate" : null,
    "passwordResetEmailCode" : null,
    "passwordResetPhoneCode" : null,
    "restrictAccessToKnownComputers" : false,
    "googleApiKey" : null,
    "id" : "f3641d57-41bf-497d-a42d-bdf3975f91ad"
  }
```

As this list can be long, you can filter the output to just show things certain attributes in the object. For example, you can filter the output to just show user name and email address like this:

```
runRbCli.bat GetUsers -server dev3.riskband.ws -user acme\admin -pwd
yxLS6ySSzpjJFjcL8EqeqYhrFqqy29CV3yjd95Qd -prettyJson| findstr /i "name address"

    "name" : "acme\\eeclles",
    "firstName" : "Ephraim",
    "lastName" : "Eccles",
    "emailAddress" : "e.eclles@acmeinc.com",
    "name" : "acme\\bblack",
    "firstName" : "Byron",
    "lastName" : "Black",
    "emailAddress" : "b.black@acmeinc.com",
```

For Java CLI example being called from PowerShell, see # GetUsers on page 149.

LoginByUser

For browser example, see LoginByUser using Browser Address Bar on page 9.

For cURL example, see Calling LoginByUser on page 60.

for PowerShell example, see Calling LoginByUser on page 67.

For Java CLI, see Calling LoginByUser on page 81.

For Java SDK, see Calling RiskBandClientImpl on page 85.

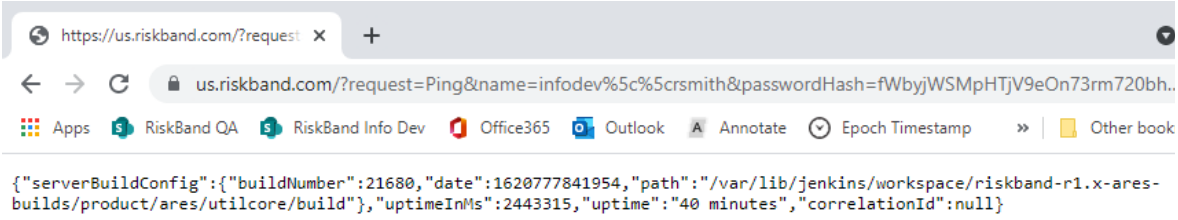
ModifyUser

For Java CLI example see # ModifyUser on page 149.

Ping

The Ping API is useful in troubleshooting connectivity from corporate networks to the RiskBand ARIES Manager. For quick troubleshooting, you can run a command similar to this from the browser address bar:

```
https://us.riskband.com/?request=Ping&name=demo%5c%5user1&passwordHash=fWbyjWSMpH
Tj9VeOn37rm720bZXY1Oxj69/+IX5kAPM=
```



If the command is successful it tells you that the browser is configured in a way that allows connectivity to the RiskBand ARIES Manager from inside the company network. However, the RiskBand ARIES Manager client sends API commands using outgoing TCP/IP on port 443 which is sometimes blocked.

RegisterDevice

Below is an example of the RegisterDevice call. Note the `-id` option needs to be the the UUID of the organization where the device is being registered. The device is identified by its `-registrationCode` and `-registrationPassword`.

The `-commit` option defaults to true. If it is set to false, the API does not perform the register device operation, and it only checks to ensure there are no validation or security errors.

```
D:\CLI\22020> runRbCli.bat RegisterDevice -server test2.riskband.ws -user sysadmin
-pwd password123 -id 2980be65-54b4-4f62-b135-2a549e11dae7 -registrationCode 70
-registrationPassword XY7G -prettyJson
{
  "result" : {
    "name" : "nadir-2",
    "passwordHash" : "<concealed>",
    "registrationCode" : null,
    "registrationPassword" : null,
    "dateRegistered" : 1626909412580,
    "organizationId" : "2980be65-54b4-4f62-b135-2a549e11dae7",
    "divisionId" : null,
    "groupId" : null,
    "phoneNumber" : "<concealed>",
    "registeredWithGeosE911" : false,
    "cellularCarrier" : "TELIT",
    "dateOrganizationRegistered" : 1627402155435,
    "powerManagementPolicyId" : null,
    "veryLateCallingHome" : false,
    "forceFirmwareBuildId" : null,
    "deprecatedReason" : null,
    "cellularCarrierId" : "<concealed>",
    "firmwarePolicyId" : null,
    "reportedAsNeedingRecharge" : false,
    "firmwareBuildId" : "80d3cdc3-e0cb-41a8-93ed-d97e0dcf158d",
    "logCallHomeRequired" : null,
    "gpsCallHomeRequired" : false,
    "lastAuthorizationRotation" : 1626909412580,
    "dateLastHeartbeat" : 0,
    "fullyManufactured" : true,
    "inAirplaneMode" : false,
    "deprecatedDate" : null,
    "bootloaderBuildNumber" : null,
    "securityPolicyId" : null,
    "locationStalenessInMins" : null,
    "lastCellSignalQuality" : null,
    "locationCertainty" : null,
    "lastBatteryLevel" : 1,
    "callInProgress" : null,
    "lastCellSignalStrength" : null,
    "firmwareUpgradeBuildId" : null,
    "arrayDebugModules" : [ ],
    "policyDictatedFirmwareUpgradeId" : null,
    "serialNumber" : "190721-00013",
    "manufacturer" : "b5810152-badc-491a-872e-3e25735134f8",
    "rxBytes" : 0,
    "txBytes" : 0,
    "lastBatteryMv" : null,
    "debugModules" : null,
    "iccid" : "<concealed>",
    "userId" : null,
    "imei" : "<concealed>",
    "uptimeInSecs" : 0,
    "cliEnabled" : false,
    "charging" : null,
```

```
    "locationName" : null,
    "modelId" : "2404f332-9478-46d1-8fb6-3a5d91be14cf",
    "id" : "7bb4d909-4782-4d12-9233-f1e0755424d1"
  },
  "correlationId" : "RB-CLI-412F1-1-1",
  "warnDeviceUserOfPilot" : null
}
```

ReprogramDeviceAuthorization

If a device loses its credentials and is no longer able to connect to the RiskBand ARIES Manager, this API will provide a new password hash for the device.

```
D:\CLI> runRbCli.bat ReprogramDeviceAuthorization -user root -pwd password123
-server test2.riskband.ws -id 91837849-2a38-4a1b-a4db-2454c6ca6f0e -prettyJson
{
  "result" : {
    "name" : "190504-00011",
    "passwordHash" : "C2ywOceLr5ZFmgjVFDk1CVFbalJe8YqwKh3ZKPUyQ5Q=",
    "firmwarePolicyId" : null,
    "powerManagementPolicyId" : null,
    "dateLastHeartbeat" : 0,
    "registeredWithGeosE911" : false,
    "cellularCarrierId" : "••••••••••••••",
    "dateOrganizationRegistered" : null,
    "lastAuthorizationRotation" : 1620943103677,
    "fullyManufactured" : true,
    "securityPolicyId" : null,
    "organizationId" : null,
    "dateRegistered" : 1620943103677,
    "groupId" : null,
    "phoneNumber" : "••••••••••,
    "divisionId" : null,
    "uptimeInSecs" : 0,
    "lastBatteryMv" : null,
    "charging" : null,
    "locationName" : null,
    "userId" : null,
    "debugModules" : null,
    "rxBytes" : 0,
    "iccid" : "••••••••••••••••••••",
    "serialNumber" : "190504-00011",
    "cliEnabled" : false,
    "imei" : "••••••••••••••",
    "modelId" : "77539469-2c85-4070-bd9c-5cfa696022b2",
    "txBytes" : 0,
    "manufacturer" : "c0e94b3f-cb9e-49fb-a45c-497c8168ff0b",
    "inAirplaneMode" : false,
    "policyDictatedFirmwareUpgradeId" : null,
    "lastCellSignalQuality" : null,
    "logCallHomeRequired" : null,
    "cellularCarrier" : "•••••",
    "veryLateCallingHome" : false,
    "firmwareBuildId" : "e766e052-e981-4333-b6f2-1e42657a2bae",
    "reportedAsNeedingRecharge" : false,
    "registrationCode" : "2",
    "forceFirmwareBuildId" : null,
    "bootloaderBuildNumber" : null,
    "registrationPassword" : "DCUV",
    "gpsCallHomeRequired" : false,
    "lastBatteryLevel" : 1,
    "lastCellSignalStrength" : null,
    "firmwareUpgradeBuildId" : null,
    "callInProgress" : null,
    "arrayDebugModules" : [ ],
```

```
    "locationStalenessInMins" : null,
    "locationCertainty" : null,
    "id" : "91837849-2a38-4a1b-a4db-2454c6ca6f0e"
  },
  "correlationId" : "52192-1-1",
  "warnDeviceUserOfPilot" : null
}
```

ResetUserPassword

Here is a CLI example of resetting a user password:

```
D:\CLI\21680> runRbCli.bat ResetUserPassword -server test2.riskband.ws -user root
-pwd password123 -id 6d2694c0-de27-4f0f-ba97-de416221541f -password "xkjyFF413732"
-prettyJson


{
  "correlationId" : "10521-1-1",
  "newPassword" : null,
  "newPasswordHash" : null
}
```

RiskBandClientImpl

While technically not one of the RiskBand APIs, this is the Java constructor for the RiskBandClient object. It is roughly equivalent to LoginByUser. See Calling RiskBandClientImpl on page 85.

UploadDeviceFirmwareBuild

Here is an example of how to upload new device firmware to ARIES using the CLI:

```
runRbCli.bat UploadDeviceFirmwareBuild -server test2.riskband.ws -user
sysAdminName -pwd fgudNHYTBabe6Yj2XLjycZGcQ5N8KT9TA5LknabT -authType USER
-computerId 00000000-0000-0000-0000-43g49a74023d -authMfaCode 985112 -name
rb2clipv1_4-21680.bin -buildNumber 21680 -endToEndMd5Hash
94c362aa01e391e5fc6cb53c6c08f5f8 -prettyJson < rb2clipv1_4-21680.bin
```

UploadGuiBuild

Here is an example of how to upload a new GUI build to ARIES using the CLI:

```
runRbCli.bat UploadGuiBuild -server test2.riskband.ws -user sysAdminName -pwd
fgudNHYTBabe6Yj2XLjycZGcQ5N8KT9TA5LknabT -authType USER -computerId
00000000-0000-0000-0000-43g49a74023d -authMfaCode 778412 -buildNumber 21680
-endToEndMd5Hash cf0a2db4562d57718e52a91ac4cf7182 -prettyJson < guirb-21680.jar
```

B Utilities

This section contains various utilities that illustrate how the RiskBand API can be used. The following utility examples are provided:

- Require All Users to Change Password Next Login on page 148
- Create User that is an Organizational Administrator on page 151
- Converting UNIX Time with Perl on page 153

Require All Users to Change Password Next Login

This utility is a PowerShell script that use the RiskBand Java CLI to set the *mustChangePassword* flag for each user in an organization to require them to change their password on the next login.

The utility makes the following API calls:

- LoginByUser — this call returns the UUID for the user that will be making the API calls. In this example, the user is acme\apier.
- GetUsers— this gets all the users in the same organization as apier (that is, in the Acme organization). By setting the computeResultCount to 1 (or TRUE), the to total number of users is returned.
- ModifyUser — the ModifyUser call is made for each user, with the exception of the user making the API calls (acme\apier) and the organizational admin user (acme\admin). The only modification made to each user is to set the mustChangePassword flag to TRUE.

The utility is implemented as a Windows PowerShell script that calls the `runRbCli.bat` file. The utility needs to be run from the directory where batch file and the `RiskBand-Java-CLI-•••••.jar` file reside.

Full Listing of ModifyAllUsers_MustChangePassword.ps1 Utility

```
Login# As necessary, change values of "endPoint", "userName", and "pwdHash"
$endPoint = "dev3.riskband.ws"
$userName = "acme\apier"
$pwdHash = "vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc="


################################################################
# LoginByUser
$response = .\runRbCli.bat LoginByUser -server $endPoint -name $userName
-passwordHash $pwdHash -prettyJson

# Convert JSON response to an array
$responseJSONobject = $response | ConvertFrom-Json
$id = $responseJSONobject.id


################################################################
# GetUsers
$response = .\runRbCli.bat GetUsers -server $endPoint -authId $id -authKey
$pwdHash -authType USER -computeResultCount 1 -prettyJson

# Convert JSON response to an array
$responseJSONobject = $response | ConvertFrom-Json
$numUsers = $responseJSONobject.resultcount
Write-Host ("`n Number of users to modify: " + $numUsers)


################################################################
# ModifyUser
# set to require password change at next login
for ($i=0; $i -lt $numUsers; $i++)
{
    $_name = $responseJSONobject.results.name[$i]
    $_firstName = $responseJSONobject.results.firstName[$i]
    $_lastName = $responseJSONobject.results.lastName[$i]
    $_userId = $responseJSONobject.results.id[$i]
    $_orgMustChangePassword = $responseJSONobject.results.mustChangePassword[$i]

    # Exceptions
    # Don't change user we're authenticating with (acme\apier) or acme\admin
    if ($_name -eq $userName -Or $_name -eq "acme\admin")
    {
     Write-Host ("" + ($i+1) + ". Skipping " + $_name )
     continue
    }

    $loopResponse = .\runRbCli.bat ModifyUser -server $endPoint -authId $id
-authKey $pwdHash -authType USER -name $_name -firstName $_firstName -lastName
$_lastName -id $_userId  -mustChangePassword "True" -propertiesToModify
"{mustChangePassword}" -prettyJson

    # Convert JSON response to an array
    $loopResponseJSONobject = $loopResponse | ConvertFrom-Json
    Write-Host ("" + ($i+1) + ". " + $_name + " mustChangePassword flag changed
from " + $_orgMustChangePassword + " to " +
$loopResponseJSONobject.result.mustChangePassword)

    $loopResponse = ""
    $loopResponseJSONobject = ""
}
```

ModifyAllUsers_MustChangePassword.ps1 Output

```
> .\ModifyAllUsers_MustChangePassword.ps1

 Number of users to modify: 28
1. acme\egray mustChangePassword flag changed from False to True
2. acme\username2 mustChangePassword flag changed from False to True
3. acme\pparker mustChangePassword flag changed from False to True
4. acme\ddanielson mustChangePassword flag changed from False to True
5. acme\user5 mustChangePassword flag changed from False to True
6. acme\username5 mustChangePassword flag changed from False to True
7. acme\orvillewright786543 mustChangePassword flag changed from False to True
8. acme\bblack mustChangePassword flag changed from False to True
9. acme\eeclles mustChangePassword flag changed from False to True
10. acme\ggreen mustChangePassword flag changed from False to True
11. acme\rsmith mustChangePassword flag changed from False to True
12. acme\user2 mustChangePassword flag changed from True to True
13. acme\username4 mustChangePassword flag changed from False to True
14. acme\user1 mustChangePassword flag changed from True to True
15. acme\axel.andersen mustChangePassword flag changed from False to True
16. acme\wwhite mustChangePassword flag changed from False to True
17. acme\user3 mustChangePassword flag changed from True to True
18. Skipping acme\apier
19. acme\username6 mustChangePassword flag changed from False to True
20. acme\user4 mustChangePassword flag changed from False to True
21. acme\williston.smythe.rumfelt mustChangePassword flag changed from False to
True
22. acme\username3 mustChangePassword flag changed from False to True
23. Skipping acme\admin
24. acme\username1 mustChangePassword flag changed from False to True
25. acme\bbrown mustChangePassword flag changed from False to True
26. acme\cchesteron mustChangePassword flag changed from False to True
27. acme\username mustChangePassword flag changed from False to True
28. acme\jdoe mustChangePassword flag changed from False to True
```

Create User that is an Organizational Administrator

This utility is a batch file that uses the RiskBand Java CLI to create a user that is an organizational administrator that has permissions to send action action messages. Generally, an administrator is consider to be any user in an organization that is granted additional permissions after creation.

The utility makes the following API calls:

- LoginByUser — this call returns the UUID for the user that will be making the API calls. It also returns the organization ID where the administrative user will be created. But rather manually parsing out the organization ID from the response, in this example, the LoginByUser call is made a second time with responseField set to organizationId.
- CreateUser — this call returns the UUID of the user created.
- CreateOrganizationPermission — this call grants permissions to the newly created user. There are two permissions granted, but because USERS_VIEW permission also grants the DEVICE_VIEW permission, the user ends up with three permissions.
- GetOrganizationPermissions — this call returns all the permissions granted to all the users in the organization. Consequently, the results are filter to only return the permissions that match the UUID of the user that was created.

The utility is implemented as a Windows batch file that calls the `runRbCli.bat` file. The utility needs to be run from the directory where batch file and the `RiskBand-Java-CLI-•••••.jar` file reside.

Full Listing of createOrgAmind.bat File

```
@echo off

:: As necessary, change values of _ENDPOINT, _NAME, and _PWDHASH
SET _NAME=acme\apier
SET _ENDPOINT=dev3.riskband.ws
SET _PWDHASH=vtTvodT9vZVL03Bdaip4Jw7JpS7Pv7AQxhhir1x2rxc=
SET _ID=

:: Org Admin User to Create
SET _ORGADMINNAME=orgadmin
SET _ORGADMINID=
SET _ORGID=

:: LoginByUser for acme\apier UUID
call runRbCli.bat LoginByUser -server %_ENDPOINT% -name %_NAME% -passwordHash
%_PWDHASH% ^
   -responseField id >responseField.txt
SET /p _ID=<responseField.txt
:: echo _ID         = %_ID%

:: LoginByUser for organization UUID
call runRbCli.bat LoginByUser -server %_ENDPOINT% -name %_NAME% -passwordHash
%_PWDHASH% ^
   -responseField organizationId >responseField.txt
SET /p _ORGID=<responseField.txt
:: echo _ORGID      = %_ORGID%

:: CreateUser
echo.
echo Creating User "%_ORGADMINNAME%" . . .
call runRbCli.bat CreateUser -server %_ENDPOINT% -authId %_ID% -authKey %_PWDHASH%
-authType USER^
   -name %_ORGADMINNAME% ^
   -firstName "Organization" ^
   -lastName "Admin" ^
   -organizationId %_ORGID% ^
   -responseField result-id >responseField.txt
SET /p _ORGADMINID=<responseField.txt
:: echo User %_ORGADMINNAME% (%_ORGADMINID%)

:: CreateOrganizationPermission
:: USERS_VIEW permission also grants DEVICE_VIEW permission
echo.
echo Granting permissions . . .
call runRbCli.bat CreateOrganizationPermission -server %_ENDPOINT% -authId %_ID%
-authKey %_PWDHASH% -authType USER^
   -userId %_ORGADMINID% ^
   -permission USERS_VIEW
call runRbCli.bat CreateOrganizationPermission -server %_ENDPOINT% -authId %_ID%
-authKey %_PWDHASH% -authType USER^
   -userId %_ORGADMINID% ^
   -permission USERS_CREATE_ACTION_MESSAGES

:: GetOrganizationPermissions
echo.
echo Permissions for "%_ORGADMINNAME%" (%_ORGADMINID%)
call runRbCli.bat GetOrganizationPermissions -server %_ENDPOINT% -authId %_ID%
-authKey %_PWDHASH% -authType USER ^
   -computeResultCount TRUE ^
   -filters-operator MATCHES ^
   -filters-property userId ^
   -filters-value %_ORGADMINID% ^
   -prettyJson | findstr "permission resultCount"
```

createOrgAdminUser.bat Output

```
> createOrgAdminUser.bat

Creating User “orgadmin” . . .

Granting permissions . . .
{“result”:{“permission”:”USERS_VIEW”,”userId”:”35348556-e873-4192-af50-134acaee84
c1”,”id”:”68e82217-d810-4993-907c-f3ecb0207996”},”correlationId”:”B2B0F-1-1”}
{“result”:{“permission”:”USERS_CREATE_ACTION_MESSAGES”,”userId”:”35348556-e873-41
92-af50-134acaee84c1”,”id”:”983900ac-2b2f-4421-8148-35cbb06e7d9f”},”correlationId
”:”5EC35-1-1”}

Permissions for “orgadmin” (35348556-e873-4192-af50-134acaee84c1)
    “permission” : “DEVICE_VIEW”,
    “permission” : “USERS_VIEW”,
    “permission” : “USERS_CREATE_ACTION_MESSAGES”,
  “resultCount” : 3
```

Converting UNIX Time with Perl

And if the have Perl installed on Windows, you can pipe CLI command output through Perl and use a regular expression to convert UNIX time to human time. The regular expression is contained in a file called `timeconv.pl` and looks like this:

```
s/(\d{10})/localtime($1)/e
```

Using the CLI you can pipe the API output through perl using a command like this:

```
D:\CLI> runRbCli.bat GetDevices -server test2.riskband.ws –user sysadmin -pwd
password123 -prettyJson | findstr "name lastBatteryLevel dateLastHeartbeat {" |
perl -p c:\batch\timeconv.pl


{
  "results" : [ {
    "name" : "div1-8",
    "dateLastHeartbeat" : Mon Oct 12 16:34:50 2020511,
    "lastBatteryLevel" : 30,
  }, {
    "name" : "group1-3",
    "dateLastHeartbeat" : Mon Oct 12 16:30:16 2020012,
    "lastBatteryLevel" : 60,
  }, {
    "name" : "190921-00012",
    "dateLastHeartbeat" : 0,
    "lastBatteryLevel" : 1,
```

Contacting RiskBand

Contact RiskBand customer support in the following ways:

Support Site	www.riskband.com/support
email	customercare@riskband.com
Telephone Number	877-475-2263 (87RISKBAND)
Company Website	www.riskband.com