Chapter

9

Using the Actuate REST API

This chapter contains:

- Accessing Actuate BIRT iHub content using the Actuate REST API
- Using REST API Resources
- Getting an authentication token
- Managing folders
- Managing files
- Using visualizations
- Extracting data
- Managing user accounts
- Managing user groups

Accessing Actuate BIRT iHub content using the Actuate REST API

The Actuate REST API accesses your rich visualizations and dynamic data built with Actuate BIRT technology. The REST API integrates content into any web or mobile application that can connect to Actuate iHub. User accounts and user groups control access to BIRT iHub resources in a volume. The volume organizes content into files and folders like a file system and assigns privileges to different users and user groups. BIRT iHub also generates new visualizations using application and report designs created by BIRT designers. You can use the Actuate REST API to access all of these features using any programming languages or tools that support the REST standard.

Working with Actuate REST API

The Actuate REST API is a resource extension installed with iHub that responds to RESTful requests. The REST API adheres to the REST standard, a strategy for developing web and mobile components that are platform and language independent, require very little time to implement, and that use minimal client and server resources.

RESTful requests use a specific command set to access REST API resources, which simplifies implementations by providing access to essential functions and raw data. Actuate offers many APIs that provide broader functionality but they are implemented using specific tools or access resources in a wide array of formats and interfaces. The REST API provides maximum freedom for developers to create their own implementations.

Live Swagger-based documentation for the REST API operations is also installed with iHub and is accessible using a web browser using the following URL:

```
http://<web server>:5000/ihub/v1/ihubrestdocs/
```

The REST API employs Uniform Resource Identifiers (URIs) references to convey user requests to the iHub System. URIs access iHub functionality including generating and storing reports, browsing volume contents, extracting data from files and data sources, and managing users and credentials.

Client applications use the REST API to send REST requests to the REST Service. The REST service runs on a Node.js platform. The REST server module interprets REST requests and forwards them as SOAP requests to iHub, as shown in Figure 2-1.

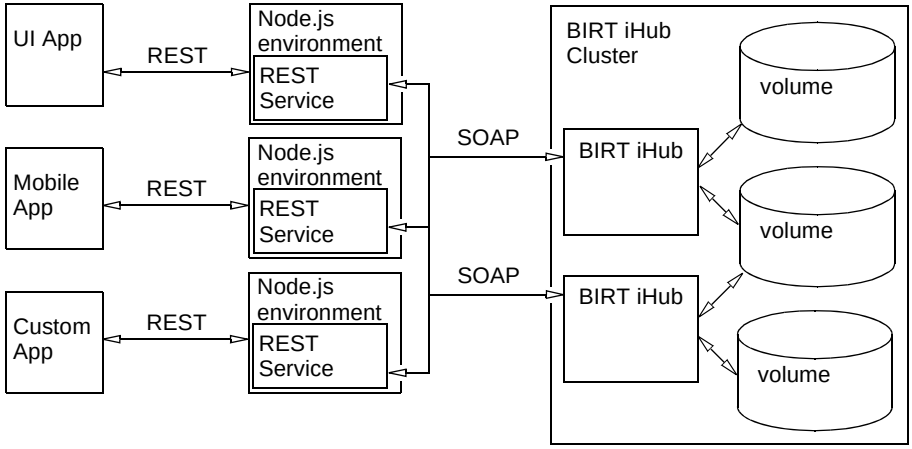

Figure 2-1 REST API tiered architecture

Actuate REST API syntax

REST requests consist of a REST method combined with a resource URI. RESTful resources are hierarchical and unique. REST requests require input parameters for some requests with complex values enclosed in braces. GET requests for a REST resource can include optional query parameters, which are not unique and are a name-value pair separated by an equal sign.

REST API URIs consist of the web server and context root of the iHub installation and port 5000, the default port for REST API traffic. To create an HTTPS request, use port 5010. REST URIs have the following syntax:

```
/ihub/v1/<resource>
   [?<parameter=value>{&<parameter=value>}]
```

- <context root>/v1 is the default context root and directory for accessing REST API resources.
- <resource> is the name of the REST API resource.
- <parameter=value> specifies the parameters and values that a GET request requires. Only GET requests use query parameters in the URL. For other requests, JSON formatted data must be passed in programmaticaly.

For example, to authenticate a user with the iHub server, use the POST method with the following resource URI:

```
http://<web server>:5000/ihub/v1/login
```

- v1/login is the REST API resource that authenticates a connection with iHub.
- The username and password parameters are passed to the server in a JSON object as part of a REST request.

UTF-8 encoding

All network communication with iHub uses UTF-8 encoding. When creating HTTP requests for a REST API application, specify UTF-8 encoding support. For example, to set the utf-8 encoding support in an HTTP header for a POST request, include charset=utf-8 in the content-type field, as shown in the following code:

```
POST /ihub/v1/folders HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded; charset=utf-8
```

When creating application code for a REST API application, provide UTF-8 encoding support using the language-specific content type settings. For example, to set the content type to UTF-8 in a Java HttpURLConnection object called conn, use the following code:

```
conn.setRequestProperty("contentType", "application/json;
   charset=utf-8");
```

Using the HTTP header

REST API transactions are configured using an HTTP header that assigns network configuration parameters to HTTP traffic with iHub. In any request, you can set the authId returned by the /login resource in the HTTP Header using the AuthId custom field. If an HTTP header includes the AuthId field, resource requests do not require an authid parameter.

You can also set a locale code using the Locale custom field, which assigns a language and country that changes the language and formatting of the content to match localized conventions. The format of the field value is a two-letter language code paired with a two letter country code separated by an underscore. The default value is set in the constants.js configuration file. Language and country codes are defined in iHub's localemap.xml configuration file.

You can set a the name for iHub volume to which to make requests using the TargetVolume custom field. The default value is set in the constants.js configuration file.

Using search functionality

REST API requests that return a list of resources from iHub, such as files, folders, jobs, bookmarks, users, or usergroups, accept a search query parameter. The search value limits the results limit the results to those resources with a name that matches the specific search value. To search for names with a substring or character, use wildcard characters. For example, a search value of 'ab*' would return a resource with the name absolutevalues but not resources with the names databank or bankstatements.

About privileges

Privileges control the visibility and actions available to users for specific files and folders. All privileges are denied unless a user or user group appears in the Access Control List (ACL) for a file or folder. The privileges granted are listed in the Permission property of

the ACL. Available privileges are Delete, Execute, Grant, Read, Secure Read, View, and Write. The Administrator user and Administrators user group do not appear in any ACL, and the Administrator user and Administrators user group have all privileges for every file and folder in the volume. Every user is a member of the All user group. If the All user group provides access to a file or folder in a volume, every user of that volume receives that access.

About response codes

All REST API requests respond with a standard set of HTTP response codes that identify the results of the request. The standard set of response codes for the Actuate REST API are as follows:

- 200: successful deletion, cancellation, retrieval of content, and updates to existing resources.
- 201: successful creation of new resources
- 304: failure to update a resource or a duplicate request.
- 400: failure to complete a request, including no privilege to perform requested operation or an incorrect value format for a parameter.
- 401: failure to delete a resource.
- 404: failure to find a resource.

The body of the response may include more information in the case of an error, similar to the following response.

```
{
   "error": {
      "description": "The specified item ID does not exist in the
   Encyclopedia volume.",
      "parameters": "123456789012",
      "code": "3073",
      "message": " Download File failed"
   }
}
```

Fetching a large number of items

REST API requests that return a list of resources from iHub, such as files, folders, jobs, users, or usergroups, accept a fetchSize query parameter that limits the number of items to return at one time. When fetchSize is not specified, lists return the number of entries set in the FETCHSIZE configuration parameter in the constants.js configuration file. When the TotalCount exceeds the fetchSize, the response object also includes a fetchHandle field that provides access to the next set of data from the returned list. For example, if fetchSize

is set to 1, and the total number of users found is 28, the first response object from a GET users request appears similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "Users": {
      "User": [
      {
         "Id": "100000000000",
         "Name": "Administrator",
         "Description": {},
         "EmailAddress": {},
         "HomeFolder": "/Home/administrator"
      }]
   },
   "FetchHandle":
   "RklMRU9SRk9MREVSICx8IHRydWUgLHwgKiAsfHwgMDoxMDowOjAgLHwgKiAsfH
   wgMDoxOjA6MA=="
}
```

To request the next 1 user from the list, use a second request for users and include the fetchHandle as a query parameter in the request, similar to the following:

```
GET /ihub/v1/
   users?fetchHandle=RklMRU9SRk9MREVSICx8IHRydWUgLHwgKiAsfHwgMDoxM
   DowOjAgLHwgKiAsfHwgMDoxOjA6MA== HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

The response includes another fetchHandle for the next set from the list. You can iterate through the list using fetchHandle in one direction. If the fetchDirection is set, the list order applies to all the results. You cannot change the fetchDirection to go backwards through a list.

Deploying the iHub REST service

The iHub installation includes the iHub REST service, which runs by default or you can deploy the iHub REST service on a separate node or machine from iHub. A web browser can send REST requests to iHub using the separate REST service or the REST service installed with iHub, as shown in Figure 2-1.

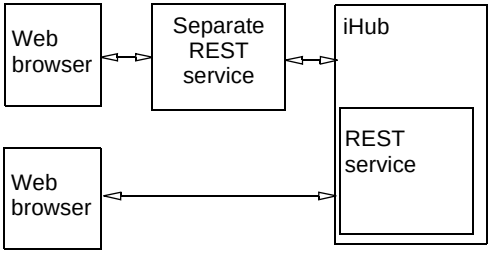

Figure 2-1 Deploying iHub REST separately

How to deploy the REST service on a separate node or machine

To deploy the iHub REST Service on a separate machine, perform the following steps.

1. Copy the server and nodejs directories and their contents to the node or machine. These directories are located in the following directory:

   ```
   <iHub installation folder>\modules\BIRTiHub\iHub\RESTAPI
   ```

2. Configure the APP_URL and HOST_NAME parameters in the constants.js configuration file to the URL and host name of the iHub server to which the REST service connects.

3. To enable the service to respond to REST requests, start the server component using the node command. For example, you can run the following command from the server directory:

   ```
   node app.js
   ```

The REST server receives requests on TCP port 5000 and forwards requests to TCP port 8000 on the iHub server.

Handling HTTPS requests using the REST API

The REST Service for the REST API runs in HTTP mode by default. To enable HTTPS support, edit the constant.js configuration file and change the value of the USEHTTPS parameter to true. You must restart the REST server service to commit the changes. For more information about configurable settings, see "Configuring the REST API using constants.js."

Once enabled, the REST service responds to HTTPS request on port 5010. The port is configurable using constants.js.

Configuring the REST API using constants.js

You can configure the behavior of several REST API resources by editing the constants.js configuration file. Constants.js is installed with iHub in the following path:

```
<iHub installation folder>\modules\BIRTiHub\iHub\RESTAPI\server\
   config
```

Table 2-2 lists and describes the parameters in the constants.js configuration file.

Table 2-2 Parameters in constants.js

Parameter	Description
APP_URL	The iHub URL for the axis service, which includes the http prefix, host and domain name, TCP port, and path to the axis resources for iHub, such as http://host.domain.com:8000/wsdl/v11/axis/all. The default value is ##REST_INSTALL_DIR##/wsdl.xml and must changed.
FETCHSIZE	The default number of entries to return for a GET request that returns a list, such as /file, /folders, /users, and /usergroups. The fetchSize parameter for these requests overrides this value.The default value is 10.
HOST_NAME	The iHub host name, which can be an IP address or a host and domain name.The default value is ##HOST_NAME##.
HTTPS_CERT_LOCATION	The path to the HTTPS certificate. By default, the REST API uses the iHub HTTPS certificate. The default value is ##REST_INSTALL_DIR##/../../shared/config/credentials/birtihub.crt.
HTTPS_PORT	The port set to respond to HTTPS requests. HTTPS support must be enabled using the USEHTTPS parameter.
HTTPS_KEY_LOCATION	The path to the HTTPS key. By default, the REST API uses the iHub HTTPS key. The default value is ##REST_INSTALL_DIR##/../../shared/config/credentials/birtihub.key.
LOCALE	Localizes content, such as currency and time format, to a particular country and language as defined in localemap.xml. The default value is en_US.
TARGETVOLUME	The default volume for REST API requests. The targetVolume parameter in the HTTP header overrides this value. The default value is Default Volume.
USEHTTPS	Enables HTTPS support. True enables HTTPS support. False disables HTTPS support. The default value is false.

Using REST API Resources

Actuate provides REST API resources using the HTTP protocol. To make a request, use a REST-enabled markup or platform to form requests containing the following elements:

- An HTTP method: GET, POST, PUT, or DELETE
- A resource identifier, as listed in the documentation
- One or more parameters: for a GET request, query parameters, or for any other request, a correctly formatted input object

The following sections describe the purpose of each of the available REST API resources and provide full reference information and examples. Each resource includes examples of the raw header and request, a curl example, and an example response. The curl examples use Windows-format system variables for the REST API host server and the authentication token. To use these examples on a Linux system, change the %variable% format to $variable.

Table 2-1 lists the Actuate Application URIs and the operations associated with particular HTTP methods applied to that resource.

Table 2-1 REST API resource and method quick reference

Resource	Supported HTTP Methods	Description
login	POST	Returns an authentication token
folders	GET, POST	Returns a list of folders or creates a new folder
folders/<folderId>	GET, DELETE, PUT	Accesses or changes a specific folder
folders/<folderId>/items	GET	Accesses the complete contents of a specific folder
folders/<folderId>/privileges	GET, POST	Accesses or changes folder privileges
files	GET	Returns a list of files
files/<fileId>	GET, DELETE, PUT	Accesses or changes a specific file
files/<fileId>/download	GET	Downloads a file from the volume
files/<fileId>/privileges	GET, POST	Accesses or changes file privileges
files/<file>/upload	POST	Uploads a file to the volume
visuals/<visualId>/bookmarks	GET	Returns a list of bookmarks from a specific report file

Table 2-1 REST API resource and method quick reference

Resource	Supported HTTP Methods	Description
visuals/<visualId>/bookmarks/<bookmarkName>	GET	Returns the contents of a report document referenced by a bookmark
visuals/<visualId>/datasets	GET	Returns the list of data sets from a report document file
GET visuals/<visualId>/datasets/<datasetname>	GET	Returns the contents of a specific data set from a report document file
visuals/<visualId>/execute	POST	Runs a report immediately and creates a transient document
visuals/<visualId>/schedule/now	POST	Runs a report design and saves the document
visuals/<visualId>/schedule/once	POST	Creates a schedule to run a job once
visuals/<visualId>/schedule/recurring	POST	Creates a recurring schedule of jobs
visuals/jobs	GET	Returns a list of jobs
visuals/<jobId>/jobs/status	GET	Returns the status for a job or schedule
visuals/<jobId>	DELETE	Removes a job or schedule
visuals/<visualId>/parameters	GET	Returns the list of parameters from a specific file
visuals/<visualId>/parameters/picklist	GET	Returns the picklist for a set of parameters from a specific file

Table 2-1 REST API resource and method quick reference

Resource	Supported HTTP Methods	Description
visuals/<visualId>/pdf	GET	Returns a page range from a report in PDF format
visuals/<visualId>/xls	GET	Returns a page range from a report in Excel format
dataobject/<dataobjectId>	GET	Returns the list of data sets from a BIRT data object
dataobject/<dataobjectId>/<dataset>	GET	Accesses a data set from a BIRT Data Object
users	GET, POST	Returns a list of users or creates a new user
users/<userId>	GET, DELETE, PUT	Accesses or changes a specific user
users/<userId>/usergroups	GET	Returns list of user groups to which a user belongs
usergroups	GET, POST	Accesses user groups or creates a new user group
usergroups/<groupId>	GET, DELETE, PUT	Accesses or changes a specific user group
usergroups/<groupId>/users	GET	Returns the list of users that belong to a user group

Getting an authentication token

Authentication is a standard security measure that verifies the identity of users requesting access to server resources by requiring a user name and password. An authId is an authentication token passed back from iHub after successful authentication and is required for all subsequent REST API requests. To generate the authId token, use a POST request for the /login resource with a username query parameter. Other parameters for /login are optional.

POST login

Authenticates a user name and password with iHub, initiated by the POST method, which sends a user name, password, and target volume to the iHub service for authentication

and, when successful, returns an authentication identifier, authId. A REST API authentication token remains valid for 24 hours by default.

Syntax /<context root>/v1/login

Parameters

locale
String. Optional. A locale code assigning a language and country that changes the language and formatting of the content to match localized conventions. The format of the input string is a two-letter language code paired with a two letter country code separated by an underscore. The default value is set in the constants.js configuration file. Language and country codes are defined in iHub's localemap.xml configuration file. Set locale in the header or as a query parameter.

targetVolume
String. Optional. The name of the iHub volume to which the authentication request is sent. The default value is set in the constants.js configuration file. Set targetVolume in the header or as a query parameter.

username
String. Required. A valid iHub user name.

password
String. Optional. The password corresponding to the user name. Provide this parameter if the user has a password.

Request examples

This HTTPS POST request example sends the administrator credentials to iHub for authentication on the Volume1 volume:

```
POST /ihub/v1/login HTTP/1.1
host: localhost:5010
content-type: application/x-www-form-urlencoded; charset=utf-8
locale: en_US
targetVolume: Volume1

username=Administrator&password=password
```

An HTTP request does not encrypt the password field so always use an HTTPS request for /login.

This curl POST request example sends the administrator credentials to iHub for authentication on the default volume:

```
curl -i https://%RESTHost%:5010/ihub/v1/login -d @loginparams.txt
```

loginparams.txt contains:

```
username=Administrator&password=password
```

Response

AuthId
String. A unique authentication identifier generated for an authenticated user. Use authId in an authid query parameter for other resources or to set the AuthId field value in the HTTP header.

AdminRights
String. The user groups to which this user belongs, which grants them the associated privileges for the iHub system. This field only appears for members of the Administrators user group.

User
A user data structure. A list of fields and values corresponding to user details, as shown in Table 2-2.

Table 2-2 The user data structure returned by /login

Field Name	Value
Id	String. A unique user ID number set by iHub.
Name	String. The username for the authenticated session.
EmailAddress	String. The user's email address.
HomeFolder	String. The user's iHub home folder.

Response example

The response contains a data object in JSON format that includes the authId, adminRights, and user data similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "AuthId": "2cwqOnGzNy6y/
   SnnFk0rU5cvuXR43zf0cfMOutp7qK+2e22RhHs5NGmZEfzBOI/
   1zq7IXQfLcY5K4rY4v6BFcYxL37ZK9fw9c902WLTVyBrFGuCOb3ORLq5GsjA6W/
   xwFOTi1a9X4O8r6qCo8Lg0QKeS6PDj6G7ybZYxEnFmUJ84uXLXzuHCk8HBkjOm9
   OUjThjLcyeczSdq/OlsTcKw72fjvY1chCsDcWbhRJwCsc0=",
   "AdminRights": "Administrator",
   "User": {
      "Id": "100000000000",
      "Name": "Administrator",
      "EmailAddress": {},
      "HomeFolder": {/Home/administrator}
   }
}
```

Response codes

Response codes for /login include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

Managing folders

A folder is a file system container for other files and folders in a volume. The iHub system identifies a folder by a file ID number, which is unique on each volume. The REST API uses file ID numbers for most requests. Users have explicit privileges to different folders. The /folders resource provides users access to files and folders, enables them to add, modify, or remove folders, and allows them to change folder privileges. iHub restricts these abilities based on a user's privileges and access rights.

GET folders

Retrieves the list of files and folders in the root directory of the volume, including their unique file ID numbers. Use these file IDs to perform other REST actions, such as navigating through the folder structure of a volume. GET folders uses the standard header fields, AuthId, Locale, and TargetVolume.

Syntax /<context root>/v1/folders

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

search
String. Optional. A search string that is compared to folder names. For example, use a value of "*Sales*" to retrieve all folder contents whose names contain Sales.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order.

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of files and folders retrieved by a previous GET /folders request. When a GET folders request returns more files and folders than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of files and folders in the files and folders list. The fetchSize parameter determines the number of files and folders in each set.

Request examples This example requests the list of files and folders in the root directory of the volume:

```
GET /ihub/v1/folders HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the contents of the root folder on the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders -H
   "authid:%RESTAuthId%"
```

Response

File

An array of File data structures. Provides details for all the files and folders in the root directory of the volume The File data structure contains the fields and values shown in Table 2-3.

Table 2-3 File data structure for folders response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type.
PageCount	Integer. The page count, which for a folder is always 0.
Size	Integer. The file size in bytes, which for a folder is always 0.
Version	Integer. The version of the file. iHub automatically generates incremental version numbers for files created with the same name in the same folder.

fetchHandle

String. A unique ID that identifies the next set of files and folders found. The fetchSize parameter determines the number of file and folders in each set. Use the fetchHandle as a query parameter with GET /folders to iterate through the complete list of files and folders found.

Response example

The response contains a data object in JSON format that includes an array of File objects and the TotalCount of files and folders, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "ItemList": {
      "File": [
      {
         "Id": "800000000000",
         "Name": "Applications",
         "FileType": "Directory",
         "PageCount": "0",
         "Size": "0"
      },
      {
         "Id": "210000000100",
         "Name": "Bookmarks.rptdocument",
         "FileType": "RPTDOCUMENT",
         "PageCount": "0",
         "Size": "365534",
         "Version": "1"
      }
      ]
   },
   "FetchHandle":
   "RklMRU9SRk9MREVSICx8IHRydWUgLHwgKiAsfHwgMDoxMDowOjAgLHwgKiAsfH
   wgMDoxOjA6MA==",
   "TotalCount": "32"
   }
}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

POST folders

Creates a new folder in the root directory of the volume or, when sent with a parentFolderId, creates a new subfolder inside a specific folder.

Syntax

/<context root>/v1/folders

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. Required. The name of the folder without the path. A folder name can be any sequence of alphanumeric characters.

parentFolderId
String. Optional. Changes the scope of the request from the root directory of the volume to a specific folder.

description
String. Optional. The description of the folder, which provides additional information about the purpose of the folder or its contents.

Request examples

This HTTP POST request example creates a new subfolder called customers in the folder with the folder ID 340000000100:

```
POST /ihub/v1/folders HTTP/1.1
Host: localhost:5000
content-type: application/x-www-form-urlencoded; charset=utf-8
AuthId: <authId>

name=customers&parentFolderId=340000000100&description=customer%20
   files
```

This curl POST request example creates a folder named Sales in the root folder of the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders -X POST -H
   "authId:%RESTAuthId%" -d name=Sales -d description="Dynamic
   Sales Information"
```

Response

The response when creating a folder includes an empty JSON object, as shown in the following HTTP response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 2

{}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure

GET folders/{folderId}

Retrieves the details for a specific folder, including the name and time stamp when it was created or last modified.

Syntax

/<context root>/v1/folders/{folderId}

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

folderId
String. Required. A current iHub folder ID number.

Request examples

This example requests the details for the folder with the folder ID 640000000100:

```
GET /ihub/v1/folders/640000000100 HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the details of the folder with the folder ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/200100000100 -H
   "authId:%RESTAuthId%"
```

Response

File
File data structure. Provides details for the requested folder. The File data structure contains the fields and values shown in Table 2-4.

Table 2-4 File data structure for folders response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type, which for folders is always "Directory."
PageCount	Integer. The page count, which for non-paginated files such as folders and report designs is always 0.
Size	Integer. The file size in bytes, which for folders is always 0.
Timestamp	String. The date and time at which this folder was created or last modified.
Owner	String. The user that owns this folder.

ACL
Array of name-value pair strings. The list of fields and values correspond to the specific access rights for each user or user group, as shown in Table 2-8.

Table 2-5 Fields of the ACL array

Field Name	Value
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively. This field precedes the UserName or RoleName to which the rights apply.

Table 2-5 Fields of the ACL array

Field Name	Value
UserName	String. A valid iHub user name
RoleName	String. A valid iHub user group name

ArchiveRules
Reserved.

Response example

The response contains a data object in JSON format that includes File details, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "File": {
      "Id": "800000000000",
      "Name": "/Applications",
      "FileType": "Directory",
      "PageCount": "0",
      "Size": "0",
      "TimeStamp": "2014-08-08T19:53:54.000Z",
      "Owner": "Administrator"
   },
   "ACL": {},
   "ArchiveRules": {}
}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

DELETE folders/{folderId}

Removes a specific folder from the volume.

Syntax

/<context root>/v1/folders/{folderId}

Parameters

folderId
String. Required. The unique ID number of the folder to delete.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

This example deletes the folder with the folder ID 70000000100:

```
DELETE /ihub/v1/folders/70000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/json
AuthId: <authId>
```

This curl DELETE request example deletes the folder with the folder ID 40010000100 from the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/400100000100 -X
   DELETE -H "authId:%RESTAuthId%"
```

Response

The response when deleting a folder contains an empty JSON object as shown in the following HTTP response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 401: Failure

PUT folders/{folderId}

Changes the name or location of a specific folder.

Syntax

/<context root>/v1/folders/{folderId}

Parameters

folderId
String. Required. The unique ID number of the folder to modify.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

newName
String. A valid iHub folder name, which replaces the previous folder name. This folder name must include the repository path. For example, to move the /Application1 folder to the /Public folder, use "/Public/Application1" for the value of newName. The server creates folders in the path if they do not already exist.

Request examples

The following HTTP request changes the name and path of the folder with folder ID 340000000100 to /Public/Application1:

```
PUT /ihub/v1/folders/340000000100 HTTP/1.1
Host: lcoalhost:5000
Content-Type: application/x-www-form-urlencoded

authid=<authId>&newName=%2FPublic%2FApplication1
```

This curl PUT request example changes the name and path the folder with the folder ID 20010000100 to /Sales/North America in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/200100000100 -X PUT
   -H "AuthId:%RESTAuthId%" -d newName=/Sales/North%20America
```

Response

The response when changing folder details contains an empty JSON object as shown in the following HTTP response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET folders/{folderId}/items

Retrieves a list of the files and folders in a specific folder.

Syntax

/<context root>/v1/folders/{folderId}/items

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

folderId
String. Required. A current iHub folder ID number.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order.

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of files and folders retrieved by a previous GET folders/{folderId}/items request. When a GET request returns more files

and folders than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of files and folders in the files and folders list. The fetchSize parameter determines the number of files and folders in each set.

Request examples

This example requests the contents of the folder with the folder ID 640000000100 using the HTTPS protocol:

```
GET /ihub/v1/folders/640000000100/items HTTP/1.1
Host: localhost:5010
AuthId=<authId>
```

This curl example requests the contents of the folder with the folder ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/200100000100/items
   -H "AuthId:%RESTAuthId%"
```

Response

File

An array of File data structures. Provides details for all the files and folders contained in the requested folder. The File data structure contains the fields and values shown in Table 2-6.

Table 2-6 File data structure for folders/{folderId}/items response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type.
PageCount	Integer. The page count, which for non-paginated files such as folders and report designs is always 0.
Size	Integer. The file size in bytes.
Version	Integer. The version of the file. iHub automatically generates incremental version numbers for files created with the same name in the same folder.

fetchHandle

String. A unique ID that identifies the next set of files and folders found. The fetchSize parameter determines the number of file and folders in each set. Use the fetchHandle as a query parameter with GET /folders/{folderId}/items to iterate through the complete list of files and folders found.

TotalCount

Integer. The total number of files and folders found, which is not limited by the fetchSize.

Response example

The response contains a data object in JSON format that includes File array and the total count of the contents of the folder, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "ItemList": {
      "File": [
      {
         "Id": "200000000100",
         "Name": "customers",
         "FileType": "Directory",
         "PageCount": "0",
         "Size": "0"
      },
      {
         "Id": "300000000100",
         "Name": "Crosstab Sample Revenue.rptdesign",
         "FileType": "RPTDESIGN",
         "PageCount": "0",
         "Size": "138761",
         "Version": "1"
      }]
   },
   "TotalCount": "2"
}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET folders/{folderId}/privileges

Returns an Access Control List (ACL) that specifies the privileges for each user and user group for a specific folder. The Administrator user and Administrators user group are sometimes not included in the ACL because the Administrator account and Administrators user group have full privileges to every file and folder in the volume.

Syntax

/<context root>/v1/folders/{folderId}/privileges

Parameters

folderId
String. Required. The folder for which to return privileges.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

The following URL requests the privileges assigned to folder ID 340000000100 when accompanied by a GET method:

```
GET /ihub/v1/folders/340000000100/privileges HTTP/1.1
Host: localhost:5010
AuthId=<authId>
```

This curl example requests the ACL for the folder with the folder ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/200100000100/
   privileges -H "AuthId:%RESTAuthId%"
```

Response

File

File data structure. Provides details for the requested folder. The File data structure contains the fields and values shown in Table 2-7.

Table 2-7 File data structure for folders response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type, which for folders is always "Directory."
Timestamp	String. The date and time at which this folder was created or last modified.

ACL

Array of name-value pair strings. The list of fields and values correspond to the specific access rights for each user or user group, as shown in Table 2-8.

Table 2-8 Fields of the ACL array

Field Name	Value
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively. This field precedes the UserName or RoleName to which the rights apply.
UserName	String. A valid iHub user name
RoleName	String. A valid iHub user group name

ArchiveRules

Reserved.

Response example

The response contains a data object in JSON format that includes File details and the ACL array, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
"File": {
    "Id": "200100000100",
    "Name": "/Sales",
    "FileType": "Directory",
    "TimeStamp": "2014-08-08T23:43:39.000Z"
  },
  "ACL": {
    "Permission": [
      {
        "AccessRight": "VSRWEDG",
        "UserName": "Maria Castillo"
      },
      {
        "AccessRight": "VSE",
        "RoleName": "All"
      }
    ]
  },
  "ArchiveRules": {}
}
```

Response codes

Response codes for /folders include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.

POST folders/{folderId}/privileges

Changes folder privileges by assigning access rights for users and user groups.

Syntax

/<context root>/v1/folders/{folderId}/privileges

Parameters

folderId
String. Required. Identifies the folder for which to change privileges.

authid
String. Required. The unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

permission
Array of name-value pair strings. A JSON object containing access rights for any set of users or user groups, as shown in Table 2-9.

Table 2-9 Permission data structure for changing folder permissions

Field Name	Value
UserName	String. A valid iHub user name, which must be followed by an AccessRight field
UserGroupName	String. A valid iHub user group name, which must be followed by an AccessRight field
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively.

To assign multiple privileges using the same request, use an array for the Permission parameter, as shown in the following example:

```
{"Permission":
   [{"UserName":"JLee", "AccessRight":"RVW"},
   {"UserGroupName":"Sales", "AccessRight":"RV"}]
}
```

Request examples

The following HTTP request changes the privileges on the folder with folder ID 340000000100 for the mrfox user and the dahl user group:

```
POST /ihub/v1/folders/340000000100/privileges HTTP/1.1
Host: localhost:5000
Content-Type: application/json; charset=utf-8
AuthId: <authId>

[{"UserName":"mrfox", "AccessRight":"DEGRVW"},
   {"UserGroupName":"dahl", "AccessRight":"RE"}]
```

This curl POST request example changes the ACL of the folder with the folder ID 20010000100 to the ACL in foldersprivsparams.txt:

```
curl -i http://%RESTHost%:5000/ihub/v1/folders/200100000100-X POST
   -H "AuthId:%RESTAuthId%" --header "Content-Type: application/
   json" -d @foldersprivsparams.txt
```

foldersprivsparams.txt contains an ACL in the following format:

```
{"Permission":
[{"UserGroupName":"All", "AccessRight":"ESV"},
{"UserName":"Maria Castillo", "AccessRight":"GRESVWD"}]
}
```

Response The response when changing folder permissions includes an empty JSON object, as shown in the following HTTP response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes Response codes for POST /folders/{folderId}/privileges include the following:

- 0: No response from the server.
- 200: Success.
- 400: Exception

Managing files

The /files resource accesses and controls files and file permissions. A file is stored on a volume, and contains the data and visualizations for the iHub system. Files are identified by a unique file ID number assigned by the iHub system and include report designs, documents in raw and formatted forms, and data objects. Additionally, the /files resource can upload and download files to and from a volume.

GET files

Retrieves the list of files in the root directory of the volume or in a specific folder.

Syntax /<context root>/v1/files

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

folderId
String. Optional. Changes the scope of the request from the root directory of the volume to a specific folder.

search
String. Optional. A search string that is compared to file names. For example, use a value of "*Sales*" to retrieve all files with names that contain Sales.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order.

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of files retrieved by a previous GET /files request. When a GET files request returns more files than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of files in the files list. The fetchSize parameter determines the number of files in each set.

Request examples

This example requests the list of files that are not directories in the root directory of the volume:

```
GET /ihub/v1/files HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This example requests the list of files within the folder with the folder ID 340000000100:

```
GET /ihub/v1/files?folderId=340000000100 HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the list of files within the folder with the folder ID 400000000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files?folderId=40000000100
   -H "AuthId:%RESTAuthId%"
```

Response

File
An array of File data structures. A list of fields and values corresponding to file details, as shown in Table 2-10.

Table 2-10 File data structure for files response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type. Common file types include: ■ PDF ■ DATA ■ RPTDESIGN ■ RPTDOCUMENT
PageCount	Integer. The page count.
Size	Integer. The file size in bytes.
Version	Integer. The version of the file.

TotalCount
Integer. The total number of files found which is not limited by the fetch size.

Response example

The response includes a data object in JSON format that includes an array of File objects and the TotalCount of files found, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8{

"ItemList": {
   "File": [
      {
         "Id": "40000000100",
         "Name": "Sales and Profit Analysis by Country.RPTDESIGN",
         "FileType": "RPTDESIGN",
         "PageCount": "0",
         "Size": "267519",
         "Version": "1"
      },
      {
         "Id": "140000000100",
         "Name": "Unshipped Orders 1H2013.RPTDESIGN",
         "FileType": "RPTDESIGN",
         "PageCount": "0",
         "Size": "234673",
         "Version": "1"
      }
   ]
},
"TotalCount": "2"
}
```

Response codes

Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET files/{fileId}

Retrieves the details for a specific file.

Syntax

/<context root>/v1/files/{fileId}

Parameters

fileId
String. Required. A current iHub file ID number.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

This example requests the details for the file with the file ID 640000000100:

```
GET /ihub/v1/files/640000000100 HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the details of the file with the file ID 500100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/500100000100 -H
   "AuthId:%RESTAuthId%"
```

Response

File

File data structure. A list of fields and values corresponding to file details, as shown in Table 2-11.

Table 2-11 File data structure for files response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type.
PageCount	Integer. The page count, which for non-paginated files such as report designs is always 0.
Size	Integer. The file size in bytes.
Timestamp	String. The date and time at which this file was created or last modified.
Owner	String. The user that owns this file.

ACL

Array of name-value pair strings. The list of fields and values correspond to the specific access rights for each user or user group, as shown in Table 2-8.

Table 2-12 Fields of the ACL array

Field Name	Value
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively. This field precedes the UserName or RoleName to which the rights apply.
UserName	String. A valid iHub user name
RoleName	String. A valid iHub user group name

ArchiveRules

Reserved.

Response example

The response includes a data object in JSON format that includes File details, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
   "File": {
      "Id": "410100000100",
      "Name": "/Classic Models.DATA",
      "FileType": "DATA",
      "PageCount": "0",
      "Size": "1154880",
      "TimeStamp": "2014-08-11T20:07:06.000Z",
      "Version": "1",
      "Owner": "Administrator"
   },
   "ACL": {
      "Permission": [{
         "AccessRight": "VRE",
         "RoleName": "All"
      }]
   },
   "ArchiveRules": {}
}
```

Response codes

Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

DELETE files/{fileId}

Removes a specific file from the iHub server.

Syntax

/<context root>/v1/files/{fileId}

Parameters

fileId
String. Required. The unique ID number of the file to delete.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

This example deletes the file with the file ID 860000000100:

```
DELETE /ihub/v1/files/860000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/json
AuthId: <authId>
```

This curl DELETE request example deletes the file with the file ID 500100000100 from the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/500100000100
   -X DELETE -H "AuthId:%RESTAuthId%"
```

Response

The response when deleting a file is an empty JSON object.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

PUT files/{fileId}

Changes the name and location of a specific file.

Syntax

/<context root>/v1/files/{fileId}

Parameters

fileId
String. Required. The file to modify.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

newName
String. A valid iHub file name, which replaces the previous file name. This file name must include the repository path and the file extension. For example, to move Unshipped Orders 1H2013.RPTDESIGN to the /Public folder, use "/Public/Unshipped Orders 1H2013.RPTDESIGN" for the value of newName. If a folder in the path does not exist, iHub creates a new folder.

Request examples

The following HTTP request changes the name of the file with file ID 640000000100 to Shipped Orders.rptdesign:

```
PUT /ihub/v1/folders/640000000100 HTTP/1.1
Host: lcoalhost:5010
Content-Type: application/x-www-form-urlencoded

authid=<authId>&newName=Shipped%20Orders.rptdesign
```

This curl PUT request example changes the name and path of the file with the file ID 50010000100 to /Public/Sales Analysis 2014-12.PDF in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/500100000100 -X PUT
   -H "AuthId:%RESTAuthId%" -d newName=/Public/
   Sales%20Analysis%202014-12.PDF
```

Response

The response when changing the file name contains an empty JSON object.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET files/{fileId}/download

Downloads a file's contents from the volume.

Syntax

/<context root>/v1/files/{fileId}/download

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

fileId
String. Required. A current iHub file ID number.

Request examples

This example downloads the content of the file with the file ID 640000000100:

```
GET /ihub/v1/files/640000000100/download HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example downloads the content of the file with the file ID 70010000100 from the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/700100000100/download
   -H "AuthId:%RESTAuthId%"
```

Response The response is the contents of the file. The format of the file determines the format of the text response. For example, a report design is formatted as XML, which is portable to BIRT Designer Professional.

Response example The download response that requests an rptdesign file contains the XML markup for the file, similar to the following:

```
HTTP/1.1 200 OK
Content-disposition: attachment; filename="Shipped
   Orders.rptdesign"
Content-Type: Application/Octet-Stream
Content-Transfer-Encoding: binary

<?xml version="1.0" encoding="UTF-8"?>

<report xmlns="http://www.eclipse.org/birt/2005/design"
   version="3.2.23" id="1">
   <property name="comments">Copyright (c) 2006 Actuate
   Corporation.</property>
   <property name="createdBy">Eclipse BIRT Designer Version
   2.1.0.N20060526-0938 Build &lt;20060526-0938></property>
   <html-property name="description"
   key="Template.SingleTable.Description">Single Table Layout</
   html-property>

      <simple-property-list name="includeResource">
         <value>StandardTemplates</value>
   ...
   <cell id="154">
   <property name="style">table_detail_cell_visible</property>
   <property name="textAlign">left</property>
      <data id="155">
         <property name="resultSetColumn">Status</property>
      </data>
      </cell>
      </row>
      </detail>
   </table>
   </body>
</report>
```

Response codes Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET files/{fileId}/privileges

Returns an Access Control List (ACL) that specifies the privileges for each user and Administrators user group for a specific file. The Administrator user and Administrators user group are sometimes not included in the ACL because the Administrator account and Administrators user group have full privileges to every file and folder in the volume.

Syntax /<context root>/v1/files/{fileId}/privileges

Parameters

fileId
String. Required. The file for which to return privileges.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples The following HTTP request returns the privileges assigned to file ID 640000000100:

```
GET /ihub/v1/files/640000000100/privileges HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the ACL for the file with the file ID 500100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/500100000100
   -H "AuthId:%RESTAuthId%"
```

Response

File
File data structure. A list of fields and values corresponding to file details, as shown in Table 2-13.

Table 2-13 File data structure for files response

Field Name	Value
Id	String. A unique file ID number set by iHub.
Name	String. The file name.
FileType	String. The file type.
Timestamp	String. The date and time at which this file was created or last modified.

ACL
Array of name-value pair strings. The list of fields and values correspond to the specific access rights for each user or user group, as shown in Table 2-14.

Table 2-14 Fields of the ACL array

Field Name	Value
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively.
UserName	String. A valid iHub user name.
RoleName	String. A valid iHub user group name.

ArchiveRules
Reserved.

Response example

The response includes a data object in JSON format that includes File details and the ACL array, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8{

{
   "File": {
      "Id": "920000000100",
      "Name": "/Applications/BIRT Sample App/Report Designs/Client
   Investment Portfolio.rptdesign",
      "FileType": "RPTDESIGN",
      "TimeStamp": "2014-08-16T00:44:12.000Z",
   },
   "ACL": {
      "Permission": [{
         "AccessRight": "VRE",
         "RoleName": "All"
      }]
   },
   "ArchiveRules": {}
}
```

Response codes

Response codes for /files include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.

POST files/{fileId}/privileges

Changes file privileges by assigning access rights for users and user groups.

Syntax /<context root>/v1/files/{fileId}/privileges

Parameters

fileId
String. Required. Identifies the file for which to change privileges.

authid
String. Required. The unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

permission
Array of name-value pair strings. A JSON Object containing access rights for any set of users or user groups, as shown in Table 2-15.

Table 2-15 Permission data structure for changing file permissions

Field Name	Value
UserName	String. A valid iHub user name
UserGroupName	String. A valid iHub user group name
AccessRight	Letters assign privileges to a user or user group using any combination of the letters D, E, G, R, S, V, and W, which correspond to the privileges Delete, Execute, Grant, Read, Secure Read, View, and Write, respectively.

To assign multiple privileges using the same request, use and array for the Permission parameter, as shown in the following example:

```
{"Permission":
   [{"UserName":"mrfox", "AccessRight":"RVW"},
   {"UserGroupName":"dahl", "AccessRight":"RV"}]
}
```

Request examples The following HTTP request changes the privileges for the file with file ID 640000000100:

```
POST /ihub/v1/files/6400000000100/privileges HTTP/1.1
Host: localhost:5000
Content-Type: application/json; charset=utf-8
AuthId: <authId>

[{"UserName":"mrfox", "AccessRight":"DEGRVW"},
   {"UserGroupName":"dahl", "AccessRight":"RE"}]
```

This curl POST request example changes the ACL of the file with the file ID 60010000100 to the ACL in the escaped JSON string:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/600100000100/
   privileges -X POST -H "AuthId:%RESTAuthId%" -H "Content-Type:
   application/json" -d "{\"Permission\": [{\"UserGroupName\":\
   "All\", \"AccessRight\":\"VS\"}, {\"UserName\":\"Maria
   Castillo\", \"AccessRight\":\"GRESVWD\"},{\"UserGroupName\":\
   "Sales\", \"AccessRight\":\"VSEW\"}]}"
```

Response

The response when changing file permissions includes an empty JSON object, as shown in the following HTTP response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for POST /files/{fileId}/privileges include the following:

- 0: No response from the server.
- 200: Success.
- 400: Exception

POST files/{file}/upload

Uploads a file to the volume or a specific folder in the volume.

Syntax

/<context root>/v1/files/{file}/upload

Parameters

file
Octet-stream. Required. The file to upload.

authid
String. Optional. The unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. A valid iHub file name, which becomes the file name on the volume. This file name must include the repository path and the file extension. For example, to upload Unshipped Orders 1H2013.RPTDESIGN to the /Public folder, use "/Public/Unshipped Orders 1H2013.RPTDESIGN" for the value of name.

Request examples

The following HTTP request uploads the file Crosstab Sample Revenue.rptdesign:

```
POST /ihub/v1/files/
   C%3A%5Cfakepath%5CCrosstab%20Sample%20Revenue.rptdesign/upload
   HTTP/1.1
Host: localhost:5000

------WebKitFormBoundary9eYER82g4Ifn9w8F
Content-Disposition: form-data; name="authId"

<authId>

------WebKitFormBoundary9eYER82g4Ifn9w8F
Content-Disposition: form-data; name="name"

Crosstab Sample Revenue.rptdesign
------WebKitFormBoundary9eYER82g4Ifn9w8F
Content-Disposition: form-data; name="file"; filename="Crosstab
   Sample Revenue.rptdesign"
Content-Type: application/octet-stream

<?xml version="1.0" encoding="UTF-8"?>
<report xmlns="http://www.eclipse.org/birt/2005/design"
   version="3.2.23" id="1">
   <property name="createdBy">Eclipse BIRT Designer Version
   2.6.1.v20100926-0604 Build &lt;2.6.1.v20100926-0604></property>
   <property name="units">in</property>
   <property name="iconFile">/templates/blank_report.gif</
   property>
   ...
</body>
</report>

------WebKitFormBoundary9eYER82g4Ifn9w8F--
```

This curl POST request example uploads the file DynamicSales.rptdesign from the current folder on disk to the /Sales folder in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/files/
   DynamicSales.rptdesign/upload -X POST -H "AuthId:%RESTAuthId%"
   -F name=/Sales/DynamicSales.rptdesign -F
   file=@DynamicSales.rptdesign
```

Response

The response when uploading a new file contains a JSON object with the file ID for the uploaded file.

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8

{
  "fileId": "600000000100"
}
```

Response codes

Response codes for POST /file/{file}/upload include the following:

- 0: No response from the server.
- 201: Success.
- 400: Exception

Using visualizations

The REST API accesses rich visualizations and dynamic report content using the /visuals resource. Rich visualizations include interactive charts, data tables, cross-tabs, images, text, and web applications. Visualizations can be exported to documents and spreadsheets for download, web and mobile applications. You can also access the data displayed in a visualization and pass that data to another application.

Most /visuals requests use a unique file ID number obtained by the /files and /folders resources to access .rptdesign or .rptdocument files. Requests that use the /visuals resource can generate content from dynamic data sources, schedule recurring jobs, and convert output to different file formats. The /visuals resource can access the contents of a rich visualization including formatted data, bookmarked objects, and parameters. The /visuals resource can also create more rich visualizations from existing executables on a scheduled or immediate basis.

GET visuals/{visualId}/bookmarks

Returns a list of bookmarks from the rptdocument file with a file ID of visualId. Bookmarks are strings that identify specific elements in an rptdocument file. The name of the bookmark is defined in the report design from which the document was created. Typical bookmarks include links to individual sections such as orders for a particular customer or sales for a particular country. Bookmarked content can be used in web applications as individual components, and a complete list of these resources provides you with bookmark names you can reference with the /visuals/{visualId}/bookmarks/{bookmarkName} resource.

Syntax /<context root>/v1/visuals/{visualId}/bookmarks

Parameters

visualId
String. Required. The file ID of a report document.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples The following URL requests the bookmarks from the file with file ID 580000000100 when accompanied by a GET method:

```
GET /ihub/v1/visuals/580000000100/bookmarks HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the bookmarks from the file with the file ID 20100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/20100000100/
   bookmarks -H "AuthId:%RESTAuthId%"
```

Response

BookmarkList

An array of Bookmark data structures. A list of fields and values corresponding to the bookmarks in the target file, as shown in Table 2-16.

Table 2-16 Bookmark data structure for bookmarks response

Field Name	Value
BookmarkValue	String. The name of a bookmark.
ElementType	String. The type of report item to which the bookmark refers
BookmarkType	String. The type of bookmark

Response example

The response includes a data object in JSON format that includes bookmarks as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
"BookmarkList": {
   "BookMark": [
   {
      "BookMarkValue": "_recreated__bookmark__1",
      "ElementType": "Data",
      "BookMarkType": "CONSTANT"
   },
   {
      "BookMarkValue": "SampleRevenue",
      "ElementType": "Crosstab",
      "BookMarkType": "CONSTANT"
   }]
}
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 304. Not modified or duplicate request.
- 400: Failure.
- 404. File not found.

GET visuals/{visualId}/bookmarks/{bookmarkName}

Returns a specific element contained in the file with a file ID of visualId referenced by the bookmark with the name bookmarkName. Any element in a rptdocument file can be bookmarked, such as charts, tables, labels, images, text, or links.

Syntax

/<context root>/v1/visuals/{visualId}/bookmarks/{bookmarkName}

Parameters

visualId
String. Required. The file ID of a report document.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

bookmarkName
String. Required. A bookmark within the file. A list of bookmarks is returned by the /visuals/{visualId}/bookmarks resource.

Request examples

The following URL requests the content referenced by the bookmark table 1 from the file with file ID 580000000100 when accompanied by a GET method:

```
GET /ihub/v1/visuals/580000000100/bookmarks/table1 HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests content referenced by the bookmark Revenue Bubble Chart from the file with the file ID 20100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/20100000100/
   bookmarks/Revenue%20Bubble%20Chart -H "AuthId:%RESTAuthId%"
```

Response

data
Array of data rows. The content of the report item referenced by the bookmark in JSON format

Response example

The response contains an array of data values from the element referenced by bookmarkName, similar to the following:

```
HTTP/1.1 200 OK
content-type: application/csv; charset=utf-8

{
   "data": [
   {
      "LeaseID": "1089",
      "LeaseAddressStreet": "123 Main Street",
      "LeaseAddressCity": "Los Angeles",
      "LeaseAddressState": "California",
      "LeaseAddressPostal": "90909",
      "Lessor": "ABC REIT",
      "Area": "38000",
      "Nominal": "",
      "LeaseBeginDate": "Jan 1, 1999 12:00 AM",
      "LeaseCurrentTermEndDate": "Dec 31, 2011 12:00 AM",
      "LeaseOptionTermsEndDate": "Dec 31, 2018 12:00 AM"
   },
   {
      "LeaseID": "562",
      "LeaseAddressStreet": "555 Random Avenue",
      "LeaseAddressCity": "San Diego",
      "LeaseAddressState": "California",
      "LeaseAddressPostal": "93939",
      "Lessor": "Nominal Franchise",
      "Area": "36000",
      "Nominal": "",
      "LeaseBeginDate": "Nov 1, 1999 12:00 AM",
      "LeaseCurrentTermEndDate": "Oct 31, 2007 12:00 AM",
      "LeaseOptionTermsEndDate": "Oct 31, 2007 12:00 AM"
   }
   ]
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.
- 404. Bookmark or file not found.

GET visuals/{visualId}/datasets

A data set is the set of records available to a particular section of a report document. GET visuals/{visualId}/datasets returns information about all the data sets in the document. The information provided includes the name, display name, and the set of columns in the data set. To retrieve the data in a data set, use the GET visuals/{visualId}/datasets/{datasetname} resource. Data sets cannot be assigned a bookmark.

Syntax /<context root>/v1/visuals/{visualId}/datasets

Parameters

visualId
String. Required. The file ID of a report document.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples The following URL requests the datasets from the file with file ID 650000000100 when accompanied by a GET method:

```
GET /ihub/v1/visuals/650000000100/datasets HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the datasets from the file with the file ID 20100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/20100000100/
   datasets -H "AuthId:%RESTAuthId%"
```

Response

ResultSetSchema
An array of result set schema data structures. A list of fields and values corresponding to result set schema details, as shown in Table 2-17.

Table 2-17 ResultSet data structure for datasets response

Field Name	Value
ResultSetName	String. The name of a data set.
ResultSetDisplayName	String. The name displayed in the report for the data set.
ArrayOfColumnSchema	Array of column schema. The columns defined for the data set.

ColumnSchema
An array of column schema data structures. A list of fields and values corresponding to column schema details, as shown in Table 2-18.

Table 2-18 ColumnSchema data structure for datasets response

Field Name	Value
Name	String. A column name.
Alias	String. A column alias.
DataType	Integer. A number identifying the data type
TypeName	String. The text identifying the data type
Label	String. The column label.
Visibility	Boolean. Indicates whether the column is visible or hidden
AllowExport	Boolean. Indicates whether the column is accessible or protected

Response example

The response contains an array of schema result set schemata in JSON format as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
"ArrayOfResultSetSchema": {
   "ResultSetSchema": [
   {
      "ResultSetName": "Company Information",
      "ResultSetDisplayName": "Company Information",
      "ArrayOfColumnSchema": {
      "ColumnSchema": [
      {
         "Name": "CONTACTLASTNAME",
         "Alias": "CONTACTLASTNAME",
         "DataType": 5,
         "TypeName": "String",
         "Label": "CONTACTLASTNAME",
         "Visibility": true,
         "AllowExport": true
      }]
   },{
      "ResultSetName": "Order Information",
      "ResultSetDisplayName": "Order Information",
      "ArrayOfColumnSchema": {
      "ColumnSchema": [
      {
         "Name": "PRODUCTNAME",
         "Alias": "PRODUCTNAME",
         "DataType": 5,
         "TypeName": "String",
         "Label": "PRODUCTNAME",
         "Visibility": true,
         "AllowExport": true
      }]
   }]
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.

GET visuals/{visualId}/datasets/{datasetname}

Returns the contents of a data set from a .rptdocument file. Retrieve a list of data set names using the /visuals/{visualId}/datasets resource.

Syntax

/<context root>/v1/visuals/{visualId}/datasets/{datasetname}

Parameters

visualId
String. Required. The file ID of a report document.

dataset
String. Required. The name of a data set from the target file.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

The following URL requests the Order Information data set from the file with file ID 650000000100 when accompanied by a GET method:

```
GET /ihub/v1/visuals/650000000100/datasets/Order%20Information
   HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the ELEMENT_1281_5 data set from the file with the file ID 20100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/20100000100/
   datasets/ELEMENT_1281_5 -H "AuthId:%RESTAuthId%"
```

Response

data
String. An array of rows from the data set.

Response example

The response includes a data object in JSON format consisting of an array of rows from the data set, as shown in the following code:

```
HTTP/1.1 200 OK
content-type: application/csv; charset=utf-8{
"data": [
   {
      "PRODUCTNAME": "1996 Moto Guzzi 1100i",
      "PRODUCTCODE": "S10_2016",
      "ORDERDATE": "Apr 29, 2011",
      "ORDERNUMBER": "10120",
      "QUANTITYORDERED": "29",
      "PRICEEACH": "118.94",
      "OrderTotalAgg": "3449.2599999999998",
      "Aggregation": "45864.03",
      "Aggregation_1": "180585.07"
   },
   {
      "PRODUCTNAME": "1940s Ford truck",
      "PRODUCTCODE": "S18_4600",
      "ORDERDATE": "Nov 29, 2012",
      "ORDERNUMBER": "10347",
      "QUANTITYORDERED": "45",
      "PRICEEACH": "115.03",
      "OrderTotalAgg": "5176.35",
      "Aggregation": "41995.62",
      "Aggregation_1": "180585.07"
   }
   ]
}
```

Response codes

Response codes for /visuals/{visualId}/datasets/{dataSetName} include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure
- 404: data set or file not found

POST visuals/{visualId}/execute

Runs a report design immediately, initiated by the POST method. The output is a temporary report referenced by an object ID number. Use the object ID and the connection handle to access the temporary file. For example, the JSAPI can view a temporary report document file in the BIRT viewer using code similar to the following:

```
viewer.setReportDocument(<ObjedtId>.<OutputFileType>,
   <connectionHandle>);
```

For information on how to use the JSAPI, see Chapter 14, “Creating a custom web page using the Actuate JavaScript API.”

Report designs and documents often have parameters that filter or alter the content of the output, and are required if the BIRT designer set them as required. To set parameter values for a POST visuals/{visualId/execute request, organize a JSON-formatted string of name-value pairs in the paramValues input parameter. A GET /visuals/{visualId}/parameters request returns the parameter names and the restrictions on their values.

Syntax /<context root>/v1/visuals/{visualId}/execute

Parameters

visualId
String. Required. A unique iHub file ID number for a report design file.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

paramValues
String in JSON format. Optional or required, as designated in the report design. The parameters for the report design or document. The format is a set of JSON name-value pairs enclosed in individual braces with the "ParameterValue:" prefix. For example, to set the country and city parameters, use a string similar to the following:

```
{ "ParameterValue" : [{"Name" : "Country","Value":
   "Japan"},{"Name" : "City","Value": "Kyoto"}]}
```

If the report has any required parameters without default values, this paramValues must contain them.

Request examples This example sends a request to iHub to run the report design with the file ID 100000000100:

```
POST /ihub/v1/visuals/100000000100/execute HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded

authid=<authId>&paramValues={ "ParameterValue" : [{"Name" :
   "param1","Value": "3"},{"Name" : "param2","Value":
   "33"},{"Name" : "param3","Value": "331"},{"Name" :
   "parama","Value": "32"},{"Name" : "paramb","Value": "321"}]}
```

This curl example sends a POST request to iHub to run the report design with the file ID 530000000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/530000000100/
   execute -X POST -H "AuthId:%RESTAuthId%"
```

Response

Status
String. The job status, which can be Done, Failed, FirstPage, or Pending.

ObjectId
Integer. A unique ID number that identifies the temporary output file. This ID number can be used by the JSAPI

OutputFileType
String. The file type of the temporary output file. When running a report design, the output type is rptdocument. When running a report document, the output type is either xls or pdf.

ConnectionHandle
String. A string to identify the transient report on disk.

Response example

The response contains a data object in JSON format, as shown in the following code:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8

{
  "Status": "FirstPage",
  "ObjectId": "1",
  "OutputFileType": "RPTDOCUMENT",
  "ConnectionHandle": "bAq+y9HJrv6+ASENxVq8RL21WvOI/
   w6jmbZwg7PYF19Cq3057d+JTeUKSSPpu8CJEdtWledo6r5Fb4yX4Ucvyt84/
   3qNdvlyIqVUaheXoxBYlM/
   8wc4bAAQBSAulXtsr2DvON58eYK9UFLLwuiIIjA12d+A6VRbqDUEh4QXsTxDX9o
   te2naiNdq8z7y1oAAx2wtbdi+6KSCTQeG5TUZNAJCkoE1hMX9gyXXRgxg+A4EYN
   77EWXbNIEqEHJjhzEpMZuVMXQMrUiUPRiOlIM5s2g=="
}
```

Response codes

Response codes for /execute include the following:

- 0: No response from the server.
- 201 Success.
- 400: Failure.
- 404: File not found.

POST visuals/{visualId}/schedule/now

This resource creates a job that runs immediately. iHub runs the job and saves the output to the volume. The response includes a job ID number that you can use with a GET /visuals/{jobId}/jobs/status request to check on the job status and retrieve the file ID of the output file. The job type depends on the file referenced by the input. A job that uses a rich visualization file converts the output to another format, such as PDF or Excel. A job that uses a report design generates a file in a standard format such as PDF or Excel, or a rich visualization in rptdocument format.

Syntax /<context root>/v1/visuals/{visualId}/schedule/now

Parameters

visualId
String. Required. A unique iHub file ID number, which must correspond to a report design or report document file.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

jobName
String. Optional. A name for the job assigned to the job in the job queue.

fileType
String. Required. The file type of the input file, which is either rptdocument or rptdesign. If the input file type is rptdocument, schedule converts the file into another format and saves the output to the volume. If the input file type is rptdesign, schedule executes the report design and saves the output as a PDF, Excel, or rptdocument file.

outputFileName
String. Required. A unique iHub file name without any file extension.

outputFileFormat
String. Required. The output file extension. When fileType is rptdocument, the valid file extensions for the output file are PDF, XLS, and XLSX. Valid output types for an rptdesign include RPTDOCUMENT.

paramValues
String in JSON format. Optional or required, as designated in the report design or document. Contains the parameters and their values for the report design or document. The format is a set of JSON name-value pairs enclosed in individual braces with the "ParameterValue:" prefix. For example, to set the country and state parameters, use a string similar to the following:

```
{ "ParameterValue" :[{"Name" : "Country","Value": "USA"},{"Name" :
   "State","Value": "CA"}]}
```

If the report has any required parameters without default values, this paramValues must contain them.

priority
Integer. Optional. The priority to place on the job, which must be a number between 0 and 1000, with 0 being the lowest priority. The default value is 500.

Request examples

This example sends a request to iHub to run a low priority job and save the output of the report design with the file ID 640000000100 to the output file name crosstab.pdf:

```
POST /ihub/v1/visuals/640000000100/schedule/now HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded

authid=<authId>&fileType=rptdocument&outputFileName=crosstab&outpu
   tFileFormat=pdf&priority=100
```

This curl example sends a POST request to iHub to run the report design with the file ID 240000000100 as a job and save the output as a rptdocument file in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/240000000100/
   schedule/now -X POST -H "AuthId:%RESTAuthId%"
```

```
   -d fileType=rptdesign -d outputFileName="/Sales/
   Sales and Profit Analysis by Country"
   -d outputFileFormat=RPTDOCUMENT
```

Response

JobId
Integer. The ID number for the job for further operations.

__links
Reserved.

Response example

The response contains an object in JSON format, as shown in the following code:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
{
   "JobId": "100000000100",
   "_links": {
      "status": {
         "href": "/ihub/v1/100000000100/schedule/status",
         "rel": "job.status",
         "method": "GET"
      },
      "delete": {
         "href": "/ihub/v1/100000000100/schedule",
         "rel": "job.delete",
         "method": "DELETE"
      }
   }
}
```

Response codes

Response codes for /schedule/now include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure
- 404: File not found

POST visuals/{visualId}/schedule/once

This resource creates a schedule that runs a job at a specific time on a specific date. The response includes a job ID number for the new schedule that you can use with a GET /visuals/{jobId}/jobs/status request to check on the schedule status and retrieve the file ID of the output file. The type of job the schedule generates depends on the file referenced by the input. A job that uses a rich visualization file converts the output to another format, such as PDF or Excel. A job that uses a report design generates a file in a standard format such as PDF or Excel, or a rich visualization in rptdocument format.

Syntax

/<context root>/v1/visuals/{visualId}/schedule/once

Parameters

visualId
String. Required. A unique iHub file ID number, which must correspond to a report design or report document file.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

jobName
String. Optional. A name for the job assigned to the job in the job queue.

fileType
String. Required. The file for the input file, which is either rptdocument or rptdesign. If the input file type is rptdocument, schedule converts the file into another format and saves the output to the volume. If the input file type is rptdesign, schedule executes the report design and saves the output as an rptdocument file.

outputFileName
String. Required. A unique iHub file name without any file extension.

outputFileFormat
String. Required. The output file extension. When fileType is rptdocument, the valid file extensions for the output file are PDF, XLS, and XLSX. Valid output types for an rptdesign include RPTDOCUMENT.

scheduleDate
String. Optional. The day of the week or month to run the job. The format is yyyy-mm-dd, where yyyy is the year, mm is the number of the month, and dd is the number of the day.

scheduleTime
String. Required. The time to run the job on the day specified by scheduleDate. The format is hh:mm:ss where hh is the hour, mm is the number of minutes, and ss is the number of seconds.

priority
Integer. Optional. The priority to place on the job, which must be a number between 0 and 1000, with 0 being the lowest priority. The default value is 500.

paramValues
String in JSON format. Optional or required, as designated in the report design or document. Contains the parameters and their values for the report design or document. The format is a set of JSON name-value pairs enclosed in individual braces with the "ParameterValue:" prefix. For example, to set the country and state parameters, use a string similar to the following:

```
{ "ParameterValue" :[{"Name" : "Country","Value": "USA"},{"Name" :
   "State","Value": "CA"}]}
```

If the report has any required parameters without default values, this paramValues must contain them.

timeZone
String. Optional. The time zone to calculate date stamps and time-based data in a visualization. The default value is the current time zone set for the iHub server. For example, one Pacific time zone is America/Los_Angeles.

Request examples

This example sends a request to iHub to schedule a report job to run the report design identified by the file ID number 640000000100 once at 8:45 exactly:

```
POST /ihub/v1/visuals/640000000100/schedule/once HTTP/1.1
Host: localhost.com:5000
Content-Type: application/x-www-form-urlencoded

authid=<authId>&priority=1000&fileType=rptdesign&outputFileName=cr
   osstab&outputFileFormat=rptdocument&scheduleTime=08%3A45%3A00
```

This curl example sends a POST request to iHub to run the report design with the file ID 530000000100 as a job at 0:30 a.m. on December 1, 2014 and save the output as a PDF file in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/530000000100/
   schedule/once -X POST -H "AuthId:%RESTAuthId%"
   -d fileType=rptdesign -d outputFileName="/Sales/
   Revenue History by Product Line" -d outputFileFormat=PDF
   -d scheduleDate="2014-12-01" -d scheduleTime="00:30:00"
```

Response

JobId
Integer. The ID number for the schedule for further operations.

__links
Reserved.

Response example

The response includes an object in JSON format with the Job ID for the new job, as shown in the following code:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
{
   "JobId": "200000000100",
   "_links": {
      "status": {
         "href": "/ihub/v1/200000000100/schedule/status",
         "rel": "job.status",
         "method": "GET"
      },
      "delete": {
      "href": "/ihub/v1/200000000100/schedule",
      "rel": "job.delete",
      "method": "DELETE"
      }
   }
}
```

Response codes

Response codes for /schedule/once include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure.
- 404: File not found.

POST visuals/{visualId}/schedule/recurring

This resource creates a job that runs on a recurring schedule. The response includes a job ID number for the new schedule that you can use with a GET /visuals/{jobId}/jobs/status request to check on the schedule status and retrieve the file ID of the output file. The type of job the schedule generates depends on the file referenced by the input. A job that uses a rich visualization file converts the output to another format, such as PDF or Excel. A job that uses a report design generates a file in a standard format such as PDF or Excel, or a rich visualization in rptdocument format.

Syntax

/<context root>/v1/visuals/{visualId}/schedule/recurring

Parameters

visualId
String. Required. A unique iHub file ID number, which must correspond to a report design or report document file.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

endDate
String. Optional. The date to end the schedule. The format is yyyy-mm-dd, where yyyy is the year, mm is the number of the month, and dd is the number of the day.

frequency
String. Required. How often to run the job. Valid values are Daily, Weekly, and Monthly.

jobName
String. Optional. A name for the job in the job queue.

fileType
String. Required. The file for the input file, which is either rptdocument or rptdesign. If the input file type is rptdocument, schedule converts the file into another format and saves the output to the volume. If the input file type is rptdesign, schedule executes the report design and saves the output as an rptdocument file.

outputFileName
String. Required. A unique iHub file name without any file extension.

outputFileFormat
String. Required. The output file extension. When fileType is rptdocument, the valid file extensions for the output file are PDF, XLS, and XLSX. Valid output types for an rptdesign include RPTDOCUMENT.

paramValues
String in JSON format. Optional or required, as designated in the report design or document. Contains the parameters and their values for the report design or document. The format is a set of JSON name-value pairs enclosed in individual braces with the "ParameterValue:" prefix. For example, to set the country and state parameters, use a string similar to the following:

```
{ "ParameterValue" :[{"Name" : "Country","Value": "USA"},{"Name" :
   "State","Value": "CA"}]}
```

If the report has any required parameters without default values, this paramValues must contain them.

priority
Integer. Optional. The priority to place on the job, which must be a number between 0 and 1000, with 0 being the lowest priority. The default value is 500.

scheduleDay
String. Optional. The day of the week or month to run the job, depending on the value of the frequency parameter shown in Table 2-17.

Table 2-19 runOnDay value based on the value of frequency

Value of frequency	Value of runOnDay
Daily	No value.
Weekly	String. The first three letters of the English name of the day of the week upon which to run the job.
Monthly	Integer. The day of the month upon which to run the job.

scheduleTime
String. Required. The time to run the job on the day specified by runOnDay. The format is hh:mm:ss where hh is the hour, mm is the number of minutes, and ss is the number of seconds.

startDate
String. Optional. The date to start the schedule. The format is yyyy-mm-dd, where yyyy is the year, mm is the number of the month, and dd is the number of the day.

timeZone
String. Optional. The time zone to calculate date stamps and time-based data in a visualization. The default value is the current time zone set for the iHub server. For example, one Pacific time zone is America/Los_Angeles.

Request examples

This example sends a request to iHub to run the report design with the file ID 640000000100 every week on Thursday at midnight from January 1, 2014 to December 31, 2014:

```
POST /ihub/v1/visuals/640000000100/schedule/recurring HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
AuthId: <authId>

fileType=rptdesign&outputFileName=crosstab&outputFileFormat=RPTDOC
   UMENT&frequency=Weekly&scheduleDay=Thu&scheduleTime=00%3A00%3A0
   0&startDate=2014-01-01&endDate=2014-12-31
```

This curl example sends a POST request to iHub to run the report design with the file ID 530000000100 as a job at 3:30 a.m. every Tuesday, Thursday, and Saturday from the current date through December 31, 2014 and save the output as a rptdocument file in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/530000000100/
   schedule/recurring -X POST -H "AuthId:%RESTAuthId%"
   -d fileType=rptdesign -d outputFileName="/Sales/
   Revenue History by Product Line"
   -d outputFileFormat=RPTDOCUMENT -d frequency="Daily"
   -d scheduleDay="Tue,Thu,Sat" -d scheduleTime="03:30:00"
   -d endDate="2014-12-31"
```

Response

JobId
Integer. The ID number for the schedule for further operations.

__links
Reserved.

Response example

The response includes a job ID with which to reference the schedule, as shown in the following code:

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8
{
   "JobId": "500000000100",
   "_links": {
      "status": {
      "href": "/ihub/v1/500000000100/schedule/status",
      "rel": "job.status",
      "method": "GET"
   },
   "delete": {
      "href": "/ihub/v1/500000000100/schedule",
      "rel": "job.delete",
      "method": "DELETE"
   }
}
```

Response codes

Response codes for /schedule/recurring include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure.
- 404: File not found.

GET visuals/jobs

Returns a list of completed jobs, which includes succeeded, failed, and canceled jobs. The status of a job also returns details like the output file ID number, usable in /files resource requests. The visuals/jobs resource does not access schedules.

Syntax

/<context root>/v1/visuals/jobs

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

jobName
String. Optional. A search string to compare to the job names.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order.

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of jobs retrieved by a previous GET /visuals/jobs request. When a GET jobs request returns more jobs than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of jobs in the jobs list. The fetchSize parameter determines the number of jobs in each set.

Request examples

The following URL requests the status of all completed jobs:

```
GET /ihub/v1/visuals/status HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the status of all completed jobs in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/jobs
   -H "AuthId:%RESTAuthId%"
```

Response **Jobs**

An array of JobProperties data structures. JobProperties contains a list of fields and values corresponding to job details, as shown in Table 2-21.

Table 2-20 JobProperties fields for jobs status response

Field Name	Value
JobId	Integer. A unique ID number assigned to the job.
Priority	Integer. A number between 0 and 1000, with 0 being the lowest priority. The default value is 500.
JobType	String. A valid iHub operation, such as ConvertReport, RunReport, or PrintReport.
State	String. Status result of the job, which is succeeded, canceled, or failed.
InputFileName	String. The name of the report design or report document assigned to this job
RunLatestVersion	Boolean.
ActualOutputFileId	Integer. The output file ID. This field has no value until the job has finished running.
ActualOutputFile Name	String. The output file name, with the version number. This field has no value until the job has finished running.
RequestOutputFile Name	String. The output file name assigned in the job request.
SubmissionTime	String. The date and time of the job request.
CompletionTime	String. The date and time iHub completed or stopped the job.
StartTime	String. The date and time iHub started the job.

fetchHandle

String. A unique ID that identifies the next set of jobs to retrieve found. The fetchSize parameter determines the number of jobs in a set. Use the fetchHandle as a query parameter with GET /visuals/jobs to iterate through the complete list of jobs found.

TotalCount

Integer. The number of jobs found, which is not limited by the fetchSize.

Response example

The response contains a Jobs object in JSON format, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "Jobs": {
      "JobProperties": [
      {
         "JobId": "431100000100",
         "JobName": "2014-07-
   29T13:39:58_3sec_nullrpt.RPTDESIGN_PDF",
         "Priority": "500",
         "JobType": "RunReport",
         "State": "Succeeded",
         "InputFileName": "/ScheduleRecurringR14783d7415c/
   scheduleRecurringWeeklyR14783d7415c/3sec_nullrpt.RPTDESIGN;1",
         "RunLatestVersion": true,
         "ActualOutputFileId": "253200000100",
         "ActualOutputFileName": "/ScheduleRecurringR14783d7415c/
   scheduleRecurringWeeklyR14783d7415c/3sec_nullrpt.PDF;1",
         "RequestedOutputFileName": "ScheduleRecurringR14783d7415c/
   scheduleRecurringWeeklyR14783d7415c/3sec_nullrpt.PDF",
         "SubmissionTime": "2014-07-29T20:39:58.000Z",
         "CompletionTime": "2014-07-29T20:40:01.000Z",
         "StartTime": "2014-07-29T20:39:58.000Z"
      },
      {
         "JobId": "731100000100",
         "JobName": "2014-07-
   29T13:40:14_3sec_nullrpt.RPTDESIGN_PDF",
         "Priority": "500",
         "JobType": "RunReport",
         "State": "Succeeded",
         "InputFileName": "/ScheduleRecurringR14783d7415c/
   scheduleRecurringWithParametersR14783d7415c/
   3sec_nullrpt.RPTDESIGN;1",
         "RunLatestVersion": true,
         "ActualOutputFileId": "653200000100",
         "ActualOutputFileName": "/ScheduleRecurringR14783d7415c/
   scheduleRecurringWithParametersR14783d7415c/
   3sec_nullrpt.PDF;1",
         "RequestedOutputFileName": "ScheduleRecurringR14783d7415c/
   scheduleRecurringWithParametersR14783d7415c/3sec_nullrpt.PDF",
         "SubmissionTime": "2014-07-29T20:40:14.000Z",
         "CompletionTime": "2014-07-29T20:40:19.000Z",
         "StartTime": "2014-07-29T20:40:14.000Z"
      }
      ]
```

```
   },
   "TotalCount": "2"
}
```

Response codes

Response codes for /visuals/jobs include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET visuals/{jobId}/jobs/status

Returns the status of the job or schedule with the ID number jobId. Jobs are created by visuals/{visualId}/schedule/now resource requests or generated by schedules. Schedules are created by visuals/{visualId}/schedule/once and visuals/{visualId}/schedule/recurring resource requests. The status of a job also returns details like the output file ID number, usable in /files resource requests. Some of the information returned by this request is missing until a job has finished running.

Syntax

/<context root>/v1/visuals/{jobId}/jobs/status

Parameters

jobId
String. Required. A job ID is a unique number iHub assigns to every job and schedule.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

The following request gets the status of job 100000000100:

```
GET /ihub/v1/visuals/100000000100/jobs/status HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the status of the job with the job ID 310100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/310100000100/jobs/
   status -H "AuthId:%RESTAuthId%"
```

Response

JobAttributes
Job attributes data structure. A list of fields and values corresponding to job details, as shown in Table 2-21.

Table 2-21 JobAttributes fields for schedule status response

Field Name	Value
JobId	Integer. A unique ID number assigned to the job.
JobName	String. The name of the job.

Table 2-21 JobAttributes fields for schedule status response

Field Name	Value
Priority	Integer. A number between 0 and 1000, with 0 being the lowest priority. The default value is 500.
Owner	String. User account that initiated the job request.
JobType	String. A valid iHub operation, such as ConvertReport, RunReport, or PrintReport.
State	String. Final status of the job, which can be succeeded, canceled, or failed, or the status of a schedule, which can be scheduled, expired or pending.
InputFileId	Integer. The file ID of the report design assigned to this job.
InputFileName	String. The name of the report design assigned to this job
RunLatestVersion	Boolean.
ParameterFileName	String. The name of the parameter data file generated by iHub when creating a job. This file contains the values set by the schedule resource.
ActualOutputFileId	Integer. The output file ID. This field has no value until the job has finished running.
ActualOutputFile Name	String. The output file name, with the version number. This field has no value until the job has finished running.
RequestOutputFile Name	String. The output file name assigned in the job request.
SubmissionTime	String. The date and time of the job request.
CompletionTime	String. The date and time iHub completed or stopped the job.
PageCount	Integer. The page count of the output file.
OutputFileSize	Integer. The size in bytes of the output file.
RoutedToNode	String. The volume in which to store the output file.
DurationSeconds	Integer. The duration of the job in seconds.
StartTime	String. The date and time iHub started the job.
NotifyCount	Integer. The number of notifications sent.

Status
The status of the job, which can be pending, running, stopped, or completed. If the job is scheduled but has not run, iHub does not return this parameter.

Response example

The response includes a JobAttributes object in JSON format, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "JobAttributes": {
      "JobId": "100000000100",
      "JobName": "Customer Order History",
      "Priority": "1000",
      "Owner": "Administrator",
      "JobType": "RunReport",
      "State": "Succeeded",
      "InputFileId": "0",
      "InputFileName": {},
      "RunLatestVersion": false,
      "ParameterFileName": "/$$$TempROVs/tempRov/055.dat",
      "ActualOutputFileId": "650000000100",
      "ActualOutputFileName": "/Demo/Customer Order
   History.rptdocument;1",
      "RequestedOutputFileName": "/Demo/Customer Order
   History.rptdocument",
      "SubmissionTime": "2014-07-16T20:40:05.000Z",
      "CompletionTime": "2014-07-16T20:40:25.000Z",
      "PageCount": "5",
      "OutputFileSize": "344474",
      "RoutedToNode": "vmcip2",
      "DurationSeconds": "20",
      "StartTime": "2014-07-16T20:40:05.000Z",
      "NotifyCount": "0"
   },
   "Status": "Job completed."
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

DELETE visuals/{jobId}

Cancels or deletes a job or schedule. If the job has already run or failed, this request deletes the record of the job. If the job is running or has yet to run, this request cancels the job or schedule.

Syntax

/<context root>/v1/visuals/{jobId}

Parameters

jobId
String. Required. A job ID is a unique number iHub assigns to every job or schedule.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

type
String. The target for the delete which corresponds to the following input values:

- delete: Deletes a job entry that has already ran.
- cancel: Cancels a job entry that has not run or is running.
- deleteSchedule: Cancels and removed a recurring schedule.

Request examples

The following HTTP request deletes job 100000000100:

```
DELETE /ihub/v1/visuals/100000000100?type=delete HTTP/1.1
Host: localhost:5000
Content-Type: application/json
AuthId: <authId>
```

This curl DELETE example deletes the job with the job ID 120100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/120100000100/jobs
   -X DELETE -H "AuthId:%RESTAuthId%"
```

Response

The response when deleting a job is an empty JSON object.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET visuals/{visualId}/parameters

Returns a list of parameters from the file with a file ID of visualId, which must be a report design file. Parameters filter the output of a report job, which can be created by the execute and schedule resources. For example, if a revenue chart has a parameter for year, then setting a value for year restricts the output to data of the specified year. Parameters are created in a report design by a developer and can be set to a number of specific types:

- Required: Makes a parameter required to run a report design. Any schedule or execute requests must include all required parameters.

- Password: Hides the parameter value in all displays and transmissions.
- Hidden: Hides the parameter from regular parameter lists and displays. It can be set only if explicitly requested.
- Dynamic: Accepts any value and filters data output exactly like a SQL WHERE clause.
- Cascading: Changes the parameter based on the value set in a previous parameter, called the parent parameter.

Syntax /<context root>/v1/visuals/{visualId}/parameters

Parameters

visualId
String. Required. The file ID of a report design.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples The following URL requests the parameters from the file with file ID 940000000100 when accompanied by a GET method:

```
GET /ihub/v1/visuals/940000000100/parameters HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the parameters from the file with the file ID 30000000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/30000000100/
   parameters -H "AuthId:%RESTAuthId%"
```

Response

ParameterList
An array of ParameterDefinition data structures. A list of fields and values corresponding to parameter details, as shown in Table 2-22.

Table 2-22 ParameterDefinition structure for parameters response

Field Name	Value
Group	String. The group name, which only appears if the parameter is a member of a parameter group.
Name	String. A parameter name
Position	Integer. The display rank of the parameter, with 0 being the first parameter.
DataType	String. The data type of the parameter value
DefaultValue	The default value of the parameter. The data type for this field is set by the DataType field
IsRequired	Boolean. Indicates whether the parameter is required

Table 2-22 ParameterDefinition structure for parameters response

Field Name	Value
IsPassword	Boolean. Indicates whether the parameter is hidden from the interface like a password.
IsHidden	Boolean. Indicates whether the parameter is visible or hidden
DisplayName	String. The displayed name of the parameter
IsAdHoc	Boolean. Indicates whether the parameter is a dynamic filter, which accepts any value and filters data output exactly like a SQL WHERE clause.
DataSourceType	String. The data type in the data source the parameter filters.
CascadingParent Name	String. Optional. The name of the parent if this is a cascading parameter.
HelpText	String. Optional. Text for the tooltip.
IsViewParameter	Boolean. Reserved.
IsDynamicSelection List	Boolean. Indicates whether a dynamic filter parameter is formatted as a drop-down list

Response example

The response contains a ParameterList object in JSON format that includes parameter definitions, as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
"ParameterList": {
   "ParameterDefinition": [
   {
      "Group": "Date",
      "Name": "currentYear",
      "Position": 0,
      "DataType": "Integer",
      "DefaultValue": "2012",
      "IsRequired": true,
      "IsPassword": true,
      "IsHidden": true,
      "DisplayName": "currentYear",
      "IsAdHoc": false,
      "DataSourceType": "ABInfoObject",
      "CascadingParentName": {},
      "HelpText": {},
      "IsViewParameter": false,
      "IsDynamicSelectionList": false
   }]
}
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.
- 404: File not found.

GET visuals/{visualId}/parameters/picklist

Returns the picklist for the parameters from the file with a file ID of {visualId}. Picklists identify the valid values for a cascading parameter, which change depending on previous values entered in a parent parameter. To create a job with the /execute or /schedule resources using paramValues for a report design with cascading parameters, you need to get the picklist values that are valid to set a cascading parameter with a picklist.

Syntax

/<context root>/v1/visuals/{visualId}/parameters

Parameters

visualId
String. Required. The file ID of a report design.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

cascadingGroupName
String. Required. The name of a parameter group. Must contain the parameter set in paramName. This information is only available from the report design, so examine the report design fist.

paramName
String. Required. The name of the parameter for which get the picklist.

precedingParamValues
List of name-value pair data structures. Optional. The values of preceding parameters in JSON format with each name value set delimited by commas and preceded by the ParameterValue prefix. If the report contains required parameters, they must be included whether or not they affect the picklist. Each name-value pair has two fields, Name and Value, as shown in the following example:

```
{"ParameterValue": [
   {"Name" : "Country","Value": "USA"},
   {"Name" : "State","Value": "CA"}
]}
```

fileType
String. Required. The file type for which the parameter picklist is acquired, which is either rptdesign or rptdocument.

Request examples

The following URL requests the parameter picklist for the parameter CustomName from the file with file ID 940000000100:

```
GET /ihub/v1/visuals/940000000100/parameters/
   picklist?cascadingGroupName=CascadingParametersSingleDataSetLis
   tBox&paramName=CustomName&precedingParamValues={"ParameterValue
   ":[{"Name":"Language","Value":"0"},{"Name":"ForecastOrderDate",
   "Value":"4/24/1999 12:00:00 AM"}]}&fileType=RPTDESIGN HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the parameter picklist for the parameter pCustomer from the file with the file ID 430000000100 in the default volume. pCustomer is in the cascading parameter group Customer Selection. The value of the preceding parameter in the group, pCountry, is Canada:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/430000000100/
   parameters/picklist -X GET -H "AuthId:%RESTAuthId%" -d
   cascadingGroupName="Customer Selection" -d
   paramName="pCustomer" -d precedingParamValues="{ \
   "ParameterValue\" : [{\"Name\":\"pCountry\",\"Value\":\"Canada\
   "}]}" -d fileType=RPTDESIGN
```

Response

ParameterList
An array of NameValuePair data structures. A list of fields and values corresponding to parameter details, as shown in Table 2-22.

Table 2-23 NameValuePair structure for parameters response

Field Name	Value
Name	String. A parameter name
Value	The value of the parameter. The data type matches the data type of the parameter.

TotalCount
Reserved.

Response example

The response includes a data object in JSON format with parameter name value pairs as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "ParameterPickList": {
      "NameValuePair": [
      {
         "Name": "Computer Engineering",
         "Value": "Computer Engineering"
      }
      ]
   },
   "TotalCount": "1"
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure.
- 404: File not found.

GET visuals/{visualId}/pdf

Returns the current page or a range of pages of a report document in Adobe PDF format.

Syntax /<context root>/v1/visuals/{visualId}/pdf

Parameters

visualId
String. Required. The file ID of a report document.

authid
String. Optional. A unique authentication identifier generated by /login. Set authid in the header or as an input parameter.

pageRange
String. Optional. Two integers separated by a dash that indicate a range of pages for the output. For example, to return pages five through 10, set the pageRange value to 5-10.

Request examples

The following URL requests a PDF of the current page from the file with file ID 580000000100:

```
GET /ihub/v1/visuals/580000000100/pdf HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests a PDF of pages 2 to 3 from the file with the file ID 340100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/340100000100/pdf
   -X GET -H "AuthId:%RESTAuthId%" -d pageRange="2-3"
```

Response

PageRef
PageRef data structure. A list of fields and values corresponding to the PDF output from the target file, as shown in Table 2-24.

Table 2-24 PageRef data structure for pdf response

Field Name	Value
ContentId	String. The name of the output content.
ContentType	String. The content type, which for pdf is always application/octet-stream.
Locale	String. A locale code to identify the country and language for the output.
ContentData	octet-stream. The PDF output in Base64-encoded format.

ConnectionHandle
String. A string to identify the transient document on disk.

Response example

The response includes a PageRef object in JSON format that includes an octet-stream attachment of the PDF output, as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
  "PageRef": {
    "ContentId": "Attachment",
    "ContentType": "application/octet-stream",
    "Locale": "en_US",
    "ContentData": "<Content Data>"
  },
  "ConnectionHandle": "uYs7lPvcDkDUpyHTKPnFaraJpp2sl2/
   6UKssU7Xu5YimaPjlFg36tWZPpxE22MBcO7dxt9Ye8NGKHTpWon9M9y7fkS11F5
   2tjs1ABSnVgdlCKHEf/d2On1ma+PoZ3ALd"
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 304. Not modified or duplicate request.
- 400: Failure.
- 404. File not found.

GET visuals/{visualId}/xls

Returns the current page or range of pages of a report document in Excel format.

Syntax

/<context root>/v1/visuals/{visualId}/xls

Parameters

visualId
String. Required. The file ID of a report document.

authid
String. Optional. A unique authentication identifier generated by /login. Set authId in the header or as an input parameter.

pageRange
String. Optional. Two integers separated by a dash that indicate a range of pages for the output. For example, to return pages five through 10, set the pageRange value to 5-10.

Request examples

The following URL requests an Excel spreadsheet of the current page from the file with file ID 580000000100:

```
GET /ihub/v1/visuals/580000000100/xls HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests an Excel spreadsheet of page 1 from the file with the file ID 340100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/visuals/340100000100/xls
   -X GET -H "AuthId:%RESTAuthId%" -d pageRange="1"
```

Response

PageRef

PageRef data structure. A list of fields and values corresponding to the Excel output from the target file, as shown in Table 2-25.

Table 2-25 PageRef data structure for xls response

Field Name	Value
ContentId	String. The name of the output content.
ContentType	String. The content type, which for xls is always application/octet-stream.
Locale	String. A locale code to identify the country and language for the output.
ContentData	octet-stream. The Excel output in Base64-encoded format.

ConnectionHandle

String. A string to identify the transient document on disk.

Response example

The response includes a PageRef object in JSON format that includes an octet-stream attachment of the Excel output, as shown in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
  "PageRef": {
    "ContentId": "Attachment",
    "ContentType": "application/octet-stream",
    "Locale": "en_US",
    "ContentData": "<Content Data>"
  },
  "ConnectionHandle": "uYs7lPvcDkDUpyHTKPnFaraJpp2sl2/
   6UKssU7Xu5YimaPjlFg36tWZPpxE22MBcO7dxt9Ye8NGKHTpWon9M9y7fkS11F5
   2tjs1ABSnVgdlCKHEf/d2On1ma+PoZ3ALd"
}
```

Response codes

Response codes for /visuals include the following:

- 0: No response from the server.
- 200: Success.
- 304. Not modified or duplicate request.
- 400: Failure.

- 404. File not found.

Extracting data

BIRT reports access data stored in one or more BIRT data objects. A BIRT data object is a .data file in a volume and contains one or more data sets. The REST API can access the data from a data object store directly using the /dataobject resource. However, you must provide the names of the schemas and tables within a data object store to use the /dataobject resource.

GET dataobject/{dataobjectId}

Returns a list of data sets and their contents from a .data file, initiated by the GET method. The default output format for the data is JSON format.

Syntax /<context root>/v1/dataobject/{dataobjectId}

Parameters

dataobjectID
String. Required. A unique iHub file ID number, which must correspond to a BIRT data object store file, which has the .DATA file extension.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples This example sends a request to iHub to retrieve the data set list from the data file with file ID 640000000100:

```
GET /ihub/v1/dataobject/640000000100 HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the data set list from the data file with file ID 910000000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/dataobject/910000000100
   -H "AuthId:%RESTAuthId%"
```

Response

ResultSetSchema
An array of result set schema data structures. A list of fields and values corresponding to result set schema details, as shown in Table 2-17.

Table 2-26 ResultSet data structure for dataobjects/{objectId} response

Field Name	Value
ResultSetName	String. The name of a data set.
ArrayOfColumnSchema	Array of column schema. The columns defined for the data set.

ColumnSchema
An array of column schema data structures. A list of fields and values corresponding to column schema details, as shown in Table 2-18.

Table 2-27 ColumnSchema data structure for dataobjects/{objectId} response

Field Name	Value
Name	String. A column name.
Alias	String. A column alias.
DataType	Integer. A number identifying the data type
TypeName	String. The text identifying the data type
Label	String. The column label.
Visibility	Boolean. Indicates whether the column is visible or hidden
AllowExport	Boolean. Indicates whether the column is accessible or protected

Response example

The response contains an array of schema result set schemata in JSON format as shown in the following example:

```
HTTP/1.1 200 OK
content-type: application/json; charset=utf-8
{
   "ArrayOfResultSetSchema": {
      "ResultSetSchema": [
         {
            "ResultSetName": "Customers",
            "ArrayOfColumnSchema": {
               "ColumnSchema": [
                  {
                     "Name": "CUSTOMERNUMBER",
                     "Alias": "Customer #",
                     "DataType": 2,
                     "TypeName": "Integer",
                     "Label": "Customer #",
                     "Visibility": true,
                     "AllowExport": true
                  },
                  {
                     "Name": "CUSTOMERNAME",
                     "Alias": "Customer Name",
                     "DataType": 5,
                     "TypeName": "String",
                     "Label": "Customer Name",
                     "Visibility": true,
                     "AllowExport": true
                  }
               ]
            }
         },
         {
            "ResultSetName": "Products",
            "ArrayOfColumnSchema": {
               "ColumnSchema": [
                  {
                     "Name": "PRODUCTCODE",
                     "Alias": "Product Code",
                     "DataType": 5,
                     "TypeName": "String",
                     "Label": "Product Code",
                     "Visibility": true,
                     "AllowExport": true
                  },
                  {
                     "Name": "PRODUCTNAME",
```

```
                    "Alias": "Product name",
                    "DataType": 5,
                    "TypeName": "String",
                    "Label": "Product name",
                    "Visibility": true,
                    "AllowExport": true
                }
            ]
        }
    }
  ]
 }
}
```

Response codes

Response codes for /dataobject include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET dataobject/{dataobjectId}/{dataset}

Returns the contents of a data set in a .data file, initiated by the GET method. The default output format for the data is JSON format.

Syntax

/<context root>/v1/dataobject/{dataobjectId}/{dataset}

Parameters

dataobjectID
String. Required. A unique iHub file ID number, which must correspond to a BIRT data object store file, which has the .DATA file extension.

dataset
String. Required. The name of a data set from the file associated with dataobjectId. The data set names must be known before making this request.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

column
String. Optional. The name of a column or names of multiple columns from the data set. This parameter restricts the output to data from the specified column or columns. If no columns are specified, all columns are returned.

sortColumns
String. Optional. The name of a column in the data file. This parameter sorts the data in ascending alphabetical order by the data in the named column. To sort in descending order, prefix the column name with a '-'. This parameter is case-sensitive. By default, none of the content is sorted and the default order is the same as in the data object.

filterList
String. Optional. A set of filters in JSON format. String values in the operand fields require a set of single quotes (') inside the quotation marks. For example:

```
{"DataFilterCondition": {"ColumnName": "LASTNAME", "Operation":
   "=","Operand1": "'Murphy'"}}
```

Table 2-28 lists the valid filter Operations and the number of Operands to include.

Table 2-28 filterList operations

Operation	Description	Number of operands
BETWEEN	Between an inclusive range	2
=	Equal	1
>	Greater than	1
>=	Greater than or equal	1
<	Less than	1
<=	Less than or equal	1
NOT_BETWEEN	Not between an inclusive range	2
<>	Not equal	1
NOT_NULL	Is not null	0
NULL	Is null	0

format
String. Optional. The desired format of a dataset contained in the data file, csv or json. The default value is json.

Request examples

This example sends a request to iHub to retrieve the rows from the Customers data set with Year equal to 2012 from the data file with file ID 640000000100:

```
GET /ihub/v1/dataobject/640000000100/
   Customers?filterList={"DataFilterCondition":{"ColumnName":"Year
   ","Operation":"=","Operand1":"2012"}} HTTP/1.1
Host: localhost:5000
AuthId: <authId>
```

This curl example requests the rows from the Employees data set with JOBTITLE equal to Sales Rep from the data file with the file ID 910000000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/dataobject/910000000100/
   Employees -X GET -H "AuthId:%RESTAuthId%" -d filterList="{\
   "DataFilterCondition\": {\"ColumnName\": \"JOBTITLE\", \
   "Operation\": \"=\",\"Operand1\": \"'Sales Rep'\"}}"
```

Response

data
String. An array of rows from the data set.

Response example

The response includes a data object in JSON format consisting of an array of rows from the data set, as shown in the following code:

```
HTTP/1.1 200 OK
content-type: application/json; charset=utf-8
Transfer-Encoding: chunked"data": [
   {
      "ORDERDATE": "Jan 6, 2011",
      "REQUIREDDATE": "Jan 13, 2011",
      "SHIPPEDDATE": "Jan 10, 2011",
      "STATUS": "Shipped",
      "CUSTOMERNUMBER": "363",
      "PRODUCTCODE": "S18_1749",
      "QUANTITYORDERED": "30",
      "PRICEEACH": "136",
      "REVENUE": "4080",
      "PROFIT": "1479"
   },
   {
      "ORDERDATE": "May 30, 2013",
      "REQUIREDDATE": "Jun 5, 2013",
      "SHIPPEDDATE": "",
      "STATUS": "In Process",
      "CUSTOMERNUMBER": "314",
      "PRODUCTCODE": "S18_2949",
      "QUANTITYORDERED": "10",
      "PRICEEACH": "89.15",
      "REVENUE": "891.5",
      "PROFIT": "283.70000000000005"
   }
   ]
}
```

Response codes

Response codes for /dataobject include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

Managing user accounts

Every user of an iHub volume has an account on that volume. The REST API can manage user accounts using the /users resource. Only members of the Administrators user group can use this resource.

GET users

Returns the list of users. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/users

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

search
String. Optional. A search string that is compared to user names.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of users retrieved by a previous GET /users request. When a GET users request returns more users than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of users in the user list. The fetchSize parameter determines the number of users in each set.

Request examples This example requests the list of users:

```
GET /ihub/v1/users HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the list of users whose names begin with the letter j in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users?search=j*
   -H "AuthId:%RESTAuthId%"
```

Response

Users
An array of User data structures. A list of fields and values corresponding to user details, as shown in Table 2-29.

Table 2-29 User data structure for users response

Field Name	Value
Id	String. A unique user ID number set by iHub
Name	String. The user name
Description	String. The user's description

Table 2-29 User data structure for users response

Field Name	Value
EmailAddress	String. The user's email address
HomeFolder	String. The user's home folder

fetchHandle
String. A unique ID that identifies the next set of users found. The fetchSize parameter determines the number of users in each set. Use the fetchHandle as a query parameter with GET /users to iterate through the complete list of users found.

TotalCount
Integer. The number of users found, which is not limited by the fetchSize.

Response example

The response includes a data object in JSON format that includes an array of User objects and the TotalCount of users, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "Users": {
      "User": [
      {
         "Id": "100000000000",
         "Name": "Administrator",
         "Description": {},
         "EmailAddress": {},
         "HomeFolder": "/Home/administrator"
      },
      {
         "Id": "100000000100",
         "Name": "mrfox",
         "Description": {},
         "EmailAddress": {},
         "HomeFolder": {}
      }]
   },
   "FetchHandle":
   "VVNFUiAsfCB0cnVlICx8ICogLHx8IDA6MTowOjAgLHwgKiAsfHwgMDoxOjA6MA
   ==",
   "TotalCount": "2",
}
```

Response codes

Response codes for /users include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

POST users

Creates a new iHub user. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/users

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. Required. A name to assign to this user.

description
String. Optional. A description to assign to this user.

email
String. Optional. An email address to assign to this user.

homeFolder
String. Optional. A home folder to assign to this user, which is the default location the user's saved files are stored. If this folder does not already exist, iHub creates the folder when the user first logs in.

password
String. Optional. A password to assign to this user

Request examples This example creates a new user named user1, with a description of "user account for testing user requests":

```
POST /ihub/v1/users HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
AuthId: <authId>

name=user1&description=A%20user%20account%20for%20testing%20user%2
   0requests
```

This curl example sends a POST request to iHub to create the user Maria Castillo in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users
   -H "AuthId:%RESTAuthId%" -d name="Maria Castillo"
   -d email="mcastillo@mycompanynamehere.com"
   -d description="VP of North American Sales"
```

Response The response when creating a new user includes an empty JSON object.

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /users include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure

GET users/{userId}

Returns the details for a specific user. Only members of the Administrators user group can use this resource.

Syntax

/<context root>/v1/users/{userId}

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

userId
String. Required. A unique number identifying a specific user. A list of user IDs is provided by a GET /users request.

Request examples

This example requests the user details for the user with user ID 200000000100:

```
GET /ihub/v1/users/200000000100 HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the user details for the user with user ID 100100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users/100100000100
   -H "AuthId:%RESTAuthId%"
```

Response

Users
An array of User data structures of length 1. A list of fields and values corresponding to user details, as shown in Table 2-30.

Table 2-30 User data structure for users response

Field Name	Value
Id	String. A unique user ID number set by iHub
Name	String. The user name
Description	String. The user's description
EmailAddress	String. The user's email address
HomeFolder	String. The user's home folder

Response example

The response includes a data object in JSON format that includes an array of User objects containing only the one requested user account, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "Users": {
      "User": [
      {
         "Id": "200000000100",
         "Name": "user1",
         "Description": "A user account for testing user requests",
         "EmailAddress": {},
         "HomeFolder": "/Home/user1"
      }]
   },
}
```

Response codes

Response codes for /users/{userId} include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

DELETE users/{userId}

Deletes a specific user. Only members of the Administrators user group can use this resource.

Syntax

/<context root>/v1/users/{userId}

Parameters

userId
String. Required. A unique number identifying a specific user. A list of user IDs is provided by a GET /users request.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples

This example deletes the user with user ID 200000000100:

```
DELETE /ihub/v1/users/200000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/json; charset=utf-8
AuthId: <authId>
```

This curl example sends a DELETE request to iHub to delete the user with user ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users/200100000100
   -X DELETE -H "AuthId:%RESTAuthId%"
```

Response The response when deleting a user includes an empty JSON object.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes Response codes for /users include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

PUT users/{userId}

Changes the details for a specific user. Details are only changed for the parameters included with the request. All other details remain unchanged. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/users/{userId}

Parameters

userId
String. Required. A unique number identifying a specific user. A list of user IDs is provided by a GET /users request.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. Required. A name to assign to this user.

description
String. Optional. A description to assign to this user.

email
String. Optional. An email address to assign to this user.

homeFolder
String. Optional. A home folder to assign to this user.

password
String. Optional. A password to assign to this user

Request examples This example changes the description of the user with user ID 200000000100:

```
PUT /ihub/v1/users/200000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
AuthId: <authId>

description=Anr%20account%20for%20testing%20user%20requests
```

This curl example sends a PUT request to iHub to modify the home folder and description of the user with the user ID 100100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users/100100000100 -X PUT
   -H "AuthId:%RESTAuthId%" -d homeFolder=/Home/mcastillo
   -d description="Senior VP of Worldwide Sales"
```

Response The response when changing user details includes an empty JSON object.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes Response codes for /users include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET users/{userId}/usergroups

Returns the list of user groups to which the user with ID number userId belongs. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/users/{userId}/usergroups

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

userId
String. Required. A unique number identifying a specific user. A list of user IDs is provided by a GET /users request.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of user groups retrieved by a previous GET /users/{userId}/usergroups request. When a GET request returns more user groups than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of user groups in the user list. The fetchSize parameter determines the number of user groups in each set.

Request examples

This example requests the user groups to which the user with user ID 200000000100 belongs:

```
GET /ihub/v1/users/200000000100/usergroups HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the list of user groups of which the user with the user ID 100100000100 is a member in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/users/100100000100/
   usergroups -H "AuthId:%RESTAuthId%"
```

Response

UserGroups

An array of UserGroup data structures. A list of fields and values corresponding to user group details, as shown in Table 2-30.

Table 2-31 UserGroup data structure for users/{userId}/usergroups response

Field Name	Value
Id	String. A unique user group ID number set by iHub
Name	String. The user group name
Description	String. The user group's description

fetchHandle

String. A unique ID that identifies the next set of user groups found. The fetchSize parameter determines the number of user groups in each set. Use the fetchHandle as a query parameter with GET /users/{userId}/usergroups to iterate through the complete list of user groups found.

TotalCount

Integer. The number of user groups found, which is not limited by the fetchSize.

Response example The response includes a data object in JSON format that includes an array of UserGroup objects, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "UserGroups": {
      "UserGroup": [
         {
            "Id": "300000000100",
            "Name": "group1",
            "Description": {}
         },
         {
            "Id": "400000000100",
            "Name": "group2",
            "Description": {}
         },
         {
            "Id": "500000000100",
            "Name": "group3",
            "Description": {}
         }
      ]
   },
   "TotalCount": "3"
}
```

Response codes Response codes for /users/{userId}/usergroups include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

Managing user groups

A user group is a logical grouping of users that can be assigned privileges for files and folders. The REST API can manage user groups using the /usergroups resource. Only members of the Administrators user group can use this resource.

GET usergroups

Returns the list of user groups. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/usergroups

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

search
String. Optional. A search string that is compared to user group names.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of user groups retrieved by a previous GET /usergroups request. When a GET user groups request finds more user groups than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of user groups in the complete user list. The fetchSize parameter determines the number of users in each set.

Request examples

This example requests the list of user groups:

```
GET /ihub/v1/usergroups HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the list of user groups in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups
   -H "AuthId:%RESTAuthId%"
```

Response

UserGroups
An array of UserGroup data structures. A list of fields and values corresponding to user group details, as shown in Table 2-32.

Table 2-32 UserGroup data structure for usergroups response

Field Name	Value
Id	String. A unique user group ID number set by iHub
Name	String. The user group name
Description	String. The user group's description

fetchHandle
String. A unique ID that identifies the next set of user groups found. The fetchSize parameter determines the number of user group in each set. Use the fetchHandle as a query parameter with GET /folders to iterate through the complete list of user groups found.

TotalCount
Integer. The number of user groups found which is not limited to fetchSize.

Response example

The response includes a data object in JSON format that includes an array of UserGroup objects and the TotalCount of users groups, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "UserGroups": {
      "UserGroup": [
      {
         "Id": "100000000000",
         "Name": "Administrators",
         "Description": {}
      },
      {
         "Id": "200000000000",
         "Name": "All",
         "Description": {}
      },
      {
         "Id": "100000000100",
         "Name": "dahl",
         "Description": {}
      }]
   },
   "FetchHandle":
   "Uk9MRSAsfCB0cnVlICx8ICogLHx8IDA6MTowOjAgLHwgKiAsfHwgMDoxOjA6MA
   ==",
   "TotalCount": "3",
}
```

Response codes

Response codes for /usergroups include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

POST usergroups

Creates a new iHub user group. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/usergroups

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. Required. A name to assign to this user group.

description
String. Optional. A description to assign to this user group.

Request examples

This example creates the new user group dahl:

```
POST /ihub/v1/usergroups HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
AuthId: <authId>

name=dahl
```

This curl example sends a POST request to iHub to create the user group Sales in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups
   -H "AuthId:%RESTAuthId%" -d name=Sales
   -d email="salesadmin@mycompanynamehere.com"
   -d description="Sales Department"
```

Response

The response when creating a new user group contains an empty JSON object.

Response example

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /usergroups include the following:

- 0: No response from the server.
- 201: Success.
- 400: Failure

GET usergroups/{groupId}

Returns the details for a specific user group. Only members of the Administrators user group can use this resource.

Syntax

/<context root>/v1/usergroups/{groupId}

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

groupId
String. Required. A unique number identifying a specific user group. A list of group IDs is provided by a GET /usergroups request.

Request examples

This example requests the user group details for the user group with group ID 100000000100:

```
GET /ihub/v1/usergroups/100000000100 HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the user group details for the user group with user group ID 100100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups/100100000100
   -H "AuthId:%RESTAuthId%"
```

Response

UserGroups

An array of UserGroup data structures of length 1. A list of fields and values corresponding to user group details, as shown in Table 2-33.

Table 2-33 UserGroup data structure for usergroups response

Field Name	Value
Id	String. A unique user group ID number set by iHub
Name	String. The user group name
Description	String. The user group's description

Response example

The response includes a data object in JSON format that includes an array of UserGroup objects containing only the one requested user group, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "UserGroups": {
      "UserGroup": [
      {
         "Id": "100000000100",
         "Name": "dahl",
         "Description": {}
      }]
   },
}
```

Response codes

Response codes for /usergroups include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

DELETE usergroups/{groupId}

Deletes a specific user group. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/usergroups/{groupId}

Parameters

groupId
String. Required. A unique number identifying a specific user group. A list of group IDs is provided by a GET /usergroups request.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

Request examples This example deletes the group with group ID 100000000100 when combined with a DELETE method:

```
DELETE /ihub/v1/users/100000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/json; charset=utf-8
AuthId: <authId>
```

This curl example sends a DELETE request to iHub to delete the user group with user group ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups/200100000100
   -X DELETE -H "AuthId:%RESTAuthId%"
```

Response The response when deleting a user group is an empty JSON object.

Response example

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes Response codes for /usergroups include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

PUT usergroups/{groupId}

Changes the details for a specific user group. Details are only changed for the parameters included with the request. All other details remain unchanged. Only members of the Administrators user group can use this resource.

Syntax /<context root>/v1/usergroups/{groupId}

Parameters

groupId
String. Required. A unique number identifying a specific user group. A list of group IDs is provided by a GET /usergroups request.

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

name
String. Required. A name to assign to this user group.

description
String. Optional. A description to assign to this user group.

Request examples

This example he description for the user group with group ID 100000000100:

```
PUT /ihub/v1/usergroups/100000000100 HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
AuthId: <authId>

description=A%20user%20group%20for%20testing%20user%20group%20requ
   ests
```

This curl example sends a PUT request to iHub to modify the name, e-mail address, and members of the user group with the user group ID 200100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups/200100000100
   -X PUT -H "AuthId:%RESTAuthId%" -d name="North America Sales"
   -d email="NAsalesadmin@mycompanynamehere.com"
   -d addUser=100100000100
```

Response

The response when changing user group details includes an empty JSON object.

Response example

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{}
```

Response codes

Response codes for /usergroups include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure

GET usergroups/{groupId}/users

Returns the list of users that belong to the user group with ID number groupId. Only members of the Administrators user group can use this resource.

Syntax

/<context root>/v1/usergroups/{groupId}/users

Parameters

authid
String. Optional. A unique authentication identifier generated by /login. Provide this parameter if AuthId is not set in the header.

groupId
String. Required. A unique number identifying a specific user group. A list of user group IDs is provided by a GET /usergroups request.

fetchSize
Integer. Optional. The number of entries to return at one time. This parameter overrides the default value set in the constants.js configuration file.

fetchDirection
Boolean. Optional. The sort direction of the output. True fetches entries in ascending alphabetic order. False fetches entries in descending alphabetic order

fetchHandle
String. Optional. Use fetchHandle to iterate through sets of user groups retrieved by a previous GET /usergroups/{groupId}/users request. When a GET request returns more users than the fetch size, the response provides fetchHandle as a unique number. Use fetchHandle to access the next set of users in the user list. The fetchSize parameter determines the number of users in each set.

Request examples

This example requests the list of users that are members of the user group with user group ID 200000000100:

```
GET /ihub/v1/usergroups/200000000100/users HTTP/1.1
host: localhost:5000
AuthId: <authId>
```

This curl example requests the list of users that are members of the user group with the user group ID 100100000100 in the default volume:

```
curl -i http://%RESTHost%:5000/ihub/v1/usergroups/100100000100/
   users -H "AuthId:%RESTAuthId%"
```

Response

Users
An array of User data structures. A list of fields and values corresponding to user details, as shown in Table 2-29.

Table 2-34 User data structure for users response

Field Name	Value
Id	String. A unique user ID number set by iHub
Name	String. The user name
Description	String. The user's description
EmailAddress	String. The user's email address
HomeFolder	String. The user's home folder

fetchHandle
String. A unique ID that identifies the next set of users found. The fetchSize parameter determines the number of users in each set. Use the fetchHandle as a query parameter with GET /users to iterate through the complete list of users found.

TotalCount
Integer. The number of users found, which is not limited by the fetchSize.

Response example

The response includes a data object in JSON format that includes an array of User objects, similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{
   "Users": {
      "User": [
         {
            "Id": "100000000000",
            "Name": "Administrator",
            "Description": {},
            "EmailAddress": {}
         },
         {
            "Id": "300000000100",
            "Name": "user1",
            "Description": {},
            "EmailAddress": {}
         },
         {
            "Id": "200000000100",
            "Name": "user2",
            "Description": {},
            "EmailAddress": {}
         }
      ]
   },
   "TotalCount": "3"
}
```

Response codes

Response codes for /usergroups/{groupId}/users include the following:

- 0: No response from the server.
- 200: Success.
- 400: Failure