REST API interface to H5 and HSFS servers

1. Introduction.

H5 and HSFS are web servers designed and implemented by the HDF5 groups. Both servers offer a RESTful interface to retrieve HDF5-format data from servers. For the SSBD project we store BD5-files on H5 and HSFS servers. The BD5 file format is totally compatible with the HDF5-format. We request BD5 resources by using the RESTful interface defined by the HDF5 group.

This manual describes the part of the interface used to implement the webBD5 navigator included in the SSBD home page.

The full specification is described in the document:

https://support.hdfgroup.org/pubs/papers/RESTful_HDF5.pdf (last consulted at 2023/June/12).

The interface specification starts describing the three main elements of a RESTful interface:

“To define an (HTTP-based) REST interface for HDF5 we need to define three things:

1. HDF5 resources and the activities for accessing them
2. HDF5 resource identifiers (URIs)
3. HDF5 resource representations”

2. REST Interface for BD5 files

The BD5 format is based on the HDF5 format, therefore the concepts mentioned in the above definitions are applicable by extension. From now on, we will start by describing the three basic elements of the interface:

2.1. Resources

A resource in a H5 or HSDS server is a representation of a BD5 file entity or component. The next table summarizes the resources we use in our application:

Resource	Description
BD5 domain	A BD5 domain
BD5 root	A BD5 domain root, contains a reference to the BD5 root group
BD5 group collection	The collection of all BD5 groups in an BD5 domain
BD5 group	A BD5 group
BD5 dataset collection	The collection of BD5 datasets in a BD5 domain
BD5 dataset	A BD5 dataset
BD5 dataset’s attribute collection	The collection of BD5 attributes of a BD5 dataset

2.2. Resource Identifier

The interface specification defines a resource identifier as “REST uses a resource identifier to identify the particular resource involved in an interaction between components.”

Then, the specification adds a more detailed definition: “Many HDF5 entities (datasets, groups, etc.) are identified by universally unique identifiers (UUIDs). Let {id} denote such a UUID, e.g.,

aab20368-6e9c-4b91-899d-a42c9bcce117

Think of UUIDs as ‘addresses’ in a large (128-bit), generic address space.”

2.3. Resource representations

We again rely on the definition presented in the interface specification: “A representation is a sequence of bytes, plus representation metadata to describe those bytes.”

This is a very generic definition. We can say that a resource representation is the response we get from the server presented in a specific web format like JSON or XML. We will see examples with resource representations in JSON.

3. Accessing resources in webBD5 navigator

We will show examples of the three elements described in the previous subsections. We selected easy examples to illustrate the concepts.

For learning or checking data from a H5 or HSFS server we use a web navigator, or you can use the Linux command line “curl”. Because we used this interface for developing the webBD5Viewer application we will also show how to use the fetch function for retrieving data in JavaScript.

The H5 and HSFS server share the same interface, however they also have some differences. We will describe clearly when there are differences between both.

The basic pattern to access any resource at H5 and HSDS server is:

https://HOST{:PORT}/PATH

where:

HOST: an IP address or a web server name

PORT: an optional parameter when the server defines a special port where the service is offered.

PATH: A combination of resources identifiers and special additional strings to access the resources.

From now on, in short, we define WEBADDR as https://HOST{:PORT}

3.1 Domains

We can define domain as the conceptual space where several resources belong to. In a POSIX BD5 file, you can think that a BD5-file is a domain containing collections of groups and dataset that correspond to one unit. In the SSBD the organization of data is made up of projects and files. Every project and every file correspond to a domain.

3.2 Accessing projects

In an H5 server we require to navigate for several domains before we can access a project.

First, we require to know the UUID of the root domain. In H5 you can access this information from the server main address:

https://h5serv.ssbd.riken.jp/

the reply of the server is a resource representation in JSON format like this:


```
created       "2023-06-12T06:43:19Z"
lastModified"2023-06-12T06:43:19Z"
root          "1ff0adfc-fdd0-11ed-9db5-0242ac502e05"
hrefs
0
  rel         "self"
  href        "https://h5serv.ssbd.riken.jp/"
1
  rel         "database"
  href        "https://h5serv.ssbd.riken.jp/datasets"
2
  rel         "groupbase"
  href        "https://h5serv.ssbd.riken.jp/groups"
3
  rel         "typebase"
  href        "https://h5serv.ssbd.riken.jp/datatypes"
4
  rel         "root"
  href        "https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05"
```


In JavaScript we build a query for getting the same information as:

```
let server = “https:// h5serv.ssbd.riken.jp”;
let query = server;
let result = await fetch(query).json();
```

When you execute the code above you get the same information as JavaScript objects and arrays.

From the server's reply, the root domain UUID corresponds to the element 4 href link. In the web navigator we can directly access the root domain "links", with:

WEBSERVER/groups/{rootUUID}/links

Replacing the data from our environment and rootUUID we build the next link:

https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05/links

In JavaScript you require to build the query, so, we add to our previous code:

```
const rootUUID = result.root;
query = server + "/groups/" + rootUUID + "/links";
result = await fetch(query).json();
```

The response in both cases will be like this (we show a portion of the full response):

links
0
class"H5L_TYPE_HARD"
title "public"

"https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05

links
0
class "H5L_TYPE_HARD"
title "public"
href "https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05/links/public"
id "1ff1428a-fdd0-11ed-9db5-0242ac502e05"
collection"groups"
target "https://h5serv.ssbd.riken.jp/groups/1ff1428a-fdd0-11ed-9db5-0242ac502e05"
hrefs
0
href rel "self"
href "https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05/links"
1
rel "root"
href "https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05"
2
rel "home"
href "https://h5serv.ssbd.riken.jp/"
3
rel "owner"
href "https://h5serv.ssbd.riken.jp/groups/1ff0adfc-fdd0-11ed-9db5-0242ac502e05"
/links/public"
id "1ff1428a-fdd0-11ed-9db5-0242ac502e05"

For accessing a list of all the projects, we require to query the group collection containing the root domain children links. From the response we got from the previous query we took the UUID corresponding to the links[0].id element, and we built our new query with the same pattern as shown before but with a different UUID. Therefore, we build the next link:

https://h5serv.ssbd.riken.jp/groups/1ff1428a-fdd0-11ed-9db5-0242ac502e05/links

In JavaScript we add to our code the next lines:

```
const links = result.links[0].id;
query = server + “/groups/” + links + “/links”;
result = await fetch(query).json();
```

Finally, you got the resource representation containing the list of projects like this:

links
0
class "H5L_TYPE_HARD"
title "1-Bao-WormEmbryo"
href "https://h5serv.ssbd.riken.jp/groups/1ff1428a-fdd0-11ed-9db5-0242ac502e05/links/1-Bao-WormEmbryo"
id "1ff18934-fdd0-11ed-9db5-0242ac502e05"
collection "groups"
target "https://h5serv.ssbd.riken.jp/groups/1ff18934-fdd0-11ed-9db5-0242ac502e05"
1
class "H5L_TYPE_HARD"
title "10-Komatsuzaki-MolDyn"
href "https://h5serv.ssbd.riken.jp/groups/1ff1428a-fdd0-11ed-9db5-0242ac502e05/links/10-Komatsuzaki-MolDyn"
id "1ff1b922-fdd0-11ed-9db5-0242ac502e05"
collection "groups"
target "https://h5serv.ssbd.riken.jp/groups/1ff1b922-fdd0-11ed-9db5-0242ac502e05"
2
class "H5L_TYPE_HARD"
title "11-Toyoshima-NeuAct"
href "https://h5serv.ssbd.riken.jp/groups/1ff1428a-fdd0-11ed-9db5-0242ac502e05/links/11-Toyoshima-NeuAct"
id "1ff1eb90-fdd0-11ed-9db5-0242ac502e05"
collection "groups"
target "https://h5serv.ssbd.riken.jp/groups/1ff1eb90-fdd0-11ed-9db5-0242ac502e05"

In HSDS the HDF5 team added a “domain” parameter to the interface that clarifies the access to the same information. In the web navigator you can access the list of projects with the next rule for building the link:

WEBADDR/domains?

In our test server the link becomes:

http://172.21.20.232:5101/domains?

In JavaScript the code is very short as well:

```
let server = “https:// hsds.ssbd.riken.jp”;
let query = server + “/domains?”;
let result = await fetch(query).json();
```

You get the next resource representation for the list of projects:

```
domains
0
class          "folder"
owner          "admin"
created        1678150015.664639
lastModified1678150015.664639
name           "/2-Kyoda-WormEmbryoRNAi"
1
class          "folder"
owner          "admin"
created        1678233660.449496
lastModified1678233660.449496
name           "/4-Keller-FlyEmbryo"
2
class          "folder"
owner          "admin"
created        1678233676.043947
lastModified1678233676.043947
name           "/5-Keller-FishEmbryo"
3
class          "folder"
owner          "admin"
created        1678234653.7726731
lastModified1678234653.7726731
name           "/tests"
hrefs          []
```

3.3 Accessing files

In H5 with the UUID of the project domain you can request the links of the files corresponding to the selected project. Continue with the previous example we extract the project UUID of the 2-Kyoda-WormEmbryoRNAi project and build the link as:

https://h5serv.ssbd.riken.jp/groups/1ff29fb8-fdd0-11ed-9db5-0242ac502e05/links

In JavaScript we added to the previous H5 server code:

```
query = server + “/groups/” + project.id + “/links”;
result = await fetch(query).json();
```

You get a list of the files contained in the requested UUID project like this:

links
0
class "H5L_TYPE_EXTERNAL"
title "RNAi_B0336.10_040518_01_bd5"
href "https://h5serv.ssbd.riken.jp/groups/1ff29fb8-fdd0-11ed-9db5-0242ac502e05/links/RNAi_B0336.10_040518_01_bd5"
h5path "/"
h5domain "RNAi_B0336%2E10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"
target "https://h5serv.ssbd.riken.jp/?host=RNAi_B0336%2E10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"
1
class "H5L_TYPE_EXTERNAL"
title "RNAi_B0336.10_040518_02_bd5"
href "https://h5serv.ssbd.riken.jp/groups/1ff29fb8-fdd0-11ed-9db5-0242ac502e05/links/RNAi_B0336.10_040518_02_bd5"
h5path "/"
h5domain "RNAi_B0336%2E10_040518_02_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"
target "https://h5serv.ssbd.riken.jp/?host=RNAi_B0336%2E10_040518_02_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"
2
class "H5L_TYPE_EXTERNAL"
title "RNAi_B0361.10_040518_01_bd5"
href "https://h5serv.ssbd.riken.jp/groups/1ff29fb8-fdd0-11ed-9db5-0242ac502e05/links/RNAi_B0361.10_040518_01_bd5"
h5path "/"
h5domain "RNAi_B0361%2E10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"
target "https://h5serv.ssbd.riken.jp/?host=RNAi_B0361%2E10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp"

To get the same information in an HSDS server, the files list corresponds to a new domain contained inside the project domain. For that reason, you build a query requesting the domain of a domain. In the web navigator you build a links like this:

http://172.21.20.232:5101/domains?domain=/2-Kyoda-WormEmbryoRNAi

In JavaScript you add the next code to your previously shown code:

```
query = server + "/domains?domain=" + project.name;
result = await fetch(query).json();
```

Where project.name is a domain in reference to the global domain: that means that the variable project.name contains the value:

project.name = "/2-Kyoda-WormEmbryoRNAi"

The HSDS respond with the list of file of the project 2-Kyoda-WormEmbryoRNAi like this:

```
domains
0
root          "g-fb582265-c1d82eb9-73d0-aaed49-50a631"
class         "domain"
owner         "admin"
created       1678150081.5206342
lastModified1678150081.5206342
name          "/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_01_bd5.h5"
1
root          "g-53cee935-23c68857-db46-61bdab-4e00df"
class         "domain"
owner         "admin"
created       1678162954.318698
lastModified1678162954.318698
name          "/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_02_bd5.h5"
hrefs         []
```

At this point is important to notice that responses between H5 and HSDS server have several differences. The most important is the method of creating the domain name string. In the H5 server case you build an additional string like this:

?host=FILENAME.PROJECTNAME.SERVERDOMAIN

An example clarifies how to define this string: For the file RNAi_B0336.10_040518_01_bd5

of the project 2-Kyoda-WormEmbryoRNAi the additional string will become:

?host=RNAi_B0366%2E10_040518_01_bd5. 2-Kyoda-
WormEmbryoRNAi .public.h5serv.ssbd.riken.jp

Where:

FILENAME: RNAi_B0361%2E10_040518_01_bd5 (notice that this name contains a dot in the middle, to avoid syntax problems, you should change do by its corresponding Unicode (%2E). You also must notice that we do not add the file extension)

PROJECTNAME: 2-Kyoda-WormEmbryoRNAi

SERVERDOMAIN: public.h5serv.ssbd.riken.jp (it follows a naming convention like Java packages names).

On the HSDS the additional string generally will be:

?host=/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_01_bd5.h5 (In HSDS we do not require to use Unicode for the dot, and we add the file extension .h5. Notice also that the project is referenced from the root domain "/").

In our JavaScript program we named the additional domain string as "fileHost".

3.4 Accessing groups

Data in a BD5-file has two important elements: groups and datasets. The groups are like directories in a POSIX file system. They are repositories containing other groups and datasets. Datasets are like tables on a database that store data. In this section we will describe how to access groups for BD5 domains.

To build the web link to access all the groups contained in a BD5 domain we first require the domain UUID of the file we want to access. We can get any file domain by using the next rule for building links in the web navigator:

WEBADDR/{?fileHost}

Where:

WEBADDR: we explained before that WEBADDR the h5 or HSDS webserver root link such as https://h5serv.ssbd.riken.jp

{?fileHost}: We already shown how to build the fileHost for H5 and HSDS servers.

Let's show a practical example. Continuing with same file and project we are working with, you can get the BD5-file domain UUID with:

https://h5serv.ssbd.riken.jp/?host=2-Kyoda-WormEmbryoRNAi&file=RNAi_B0336.10_040518_01_bd5

In JavaScript you can create a function to produce the fileHost string, for future use when building a new query.

```
createFileHost = function createFileHost(project, file, domain) {
      return "?host=" + file + "." + project + "." + domain;
};
let fileHost = createFileHost("2-Kyoda-WormEmbryoRNAi",
"RNAi_B0361%2E10_040518_01_bd5", "public.h5serv.ssbd.riken.jp");
let query = server + "/" + fileHost;
let result = await fetch(query).json();
const fileUUID = result.root
```

After getting the file domain UUID you can navigate into the BD5-file components by getting the required UUID's, one after another. For example, to get the list of groups and data sets contained into the BD5-file data domain, we request the data domain UUID. We can get it with the next rule:

WEBADDR/groups/{fileUUID}/links/data/{?fileHost}

Note: The strings enclosed in {} will be data we get from the server or strings the user must build to generate the link requesting data from the server

Following our example, the link will be:

https://h5serv.ssbd.riken.jp/groups/057fb398-fddd-11ed-9dc9-0242ac503003/links/data?host=RNAi_B0336.10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp

```
query = server + "/groups/" + fileUUID + "/links/data" + fileHost
result = await fetch(query).json();
const dataUUID = result.link.id;
```

And later you can use the dataUUID to get the list all the groups and datasets contained in the data group with:

https://h5serv.ssbd.riken.jp/groups/05802300-fddd-11ed-9dc9-0242ac503003/links?host=RNAi_B0336.10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp

And the corresponding JavaScript code will be

```
query = server + "/groups/" + dataUUID + "/links" + fileHost
result = await fetch(query).json();
const groupLinks = result.links;
```

The result of the last link and JavaScript will be the complete list of groups and dataset names together with its corresponding UUIDs. With this data you can access all the elements of a BD5-file.

In HSDS server the process is simplified. First to get the BD5-file UUID, you can access directly to the data domain with the next link:

WEBADDR/{?fileHost}

In our example this link will be

http://172.21.20.232:5101/?host=/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_02_bd5.h5

And the corresponding JavaScript source code will be:

```
createFileHost = function createFileHost(project, file) {
      return "?host=" + project + "/" + file;
};

let fileHost = createFileHost("2-Kyoda-WormEmbryoRNAi",
"RNAi_B0361.10_040518_01_bd5");
let query = server + "/" + fileHost;
let result = await fetch(query).json();
const fileUUID = result.root
```

In a similar way, we perform two queries to get the list of groups and datasets contained in data domain.

http://172.21.20.232:5101/groups/g-53cee935-23c68857-db46-61bdab-4e00df/links/data?host=/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_02_bd5.h5

http://172.21.20.232:5101/groups/g-53cee935-23c68857-a494-573ed7-98a997/links?host=/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_02_bd5.h5

And the source code will become:

```
query = server + "/groups/" + fileUUID + "/links/data" + fileHost
result = await fetch(query).json();
const dataUUID = result.link.id;
query = server + "/groups/" + dataUUID + "/links" + fileHost
result = await fetch(query).json();
const groupLinks = result.links;
```

3.5 Accessing datasets

We already mentioned that datasets are like tables on a database. The objectcDef, scaleUnit, trackInfo are datasets that contains data corresponding to the object's definitions, scales units, and track information respectively. Objects groups contain datasets defining theirs geometries. Let's explain how to get the data values stored in datasets.

In both H5 and HSDS servers the rule for creating links to access a dataset is the same:

WEBADDR/datasets/{datasetUUID}/value{?fileHost}

By continuing with the same example, we request the scaleUnit UUID dataset data with:

https://h5serv.ssbd.riken.jp/datasets/0642fdbc-fddd-11ed-9dc9-0242ac503003/value?host=RNAi_B0336.10_040518_01_bd5.2-Kyoda-WormEmbryoRNAi.public.h5serv.ssbd.riken.jp

In our HSDS server the link will become:

http://172.21.20.232:5101/datasets/d-53cee935-23c68857-12e6-3b97f8-3d6ec4/value?host=/2-Kyoda-WormEmbryoRNAi/RNAi_B0336.10_040518_02_bd5.h5

And the JavaScript code is the same in both types of servers as:

```
query = server + “/datasets/” + scalesDataID + “/value” + fileHost
result = await fetch(query).json();
```