Webification for Earth Science

Z. Xing

Jet Propulsion Laboratory

California Institute of Technology

March 18, 2018

Version 1.1.0

Table of Contents

Section 1
Introduction

1.1 Purpose

This white paper discusses our work of applying Webification (w10n) to Earth science data. The focus will be on how Webification for Science (w10n-sci) can be used to enable uniform and ubiquitous access of data products from NASA's Earth science missions. We will also compare it with relevant data technologies from the Open Geospatial Consortium (OGC) and the Open-source Project for a Network Data Access Protocol (OPeNDAP).

1.2 Scope

Section 2 introduces w10n as a generic data virtualization technology and explains how it can be applied to different data domains. Section 3 is an in-depth discussion of w10n-sci, a special application of w10n to science data. Existing implementations of w10n-sci for Earth science are discussed and evaluated in Section 4. Section 5 provides examples of w10n-sci applied to Earth science mission products. In Section 6 and 7, we compare w10n-sci with the Web Map Service (WMS) standard from OGC and the Data Access Protocol (DAP) from OPeNDAP.

Please note that discussions here are based on our experience with respective technologies and public information available at the time of writing. Please let us know if you notice any inaccuracy and error. It will help us to improve this white paper in the future.

It is assumed that reader has a basic understanding of contemporary web technologies. Familiarity with JavaScript Object Notation (JSON) is required. It would be helpful if reader has prior working experience with tools and services from OGC and OPeNDAP.

Section 2
Webification

Webification (w10n) is a data virtualization technology. It facilitates a full exposure of an arbitrary data store in a hierarchical way. It uses meaningful, i.e., user-friendly, URLs to address the data store inner components and make them directly accessible through self-describing ReSTful web service.

Over the years, w10n has been applied to a wide ranges of data domains in both private and public sectors. Examples of stores that have been webified include file system directories, document files, databases, etc. A common application of w10n is to provide access to the data store of an individual file or a collection of them. Popular Earth Science data stores in Hierarchical Data Format (HDF) and Network Common Data Form (netCDF) are among the recent and important ones. It can also be a data model or even a remote service.

The word "webification" is often abbreviated using its numeronym "w10n". We will use them interchangeably in our discussions below.

Most contents of this and next section are taken directly from the open specification of w10n, available at http://w10n.org/spec. If there is any discrepancy between this white paper and the specification, the wording in the official specification will take precedence.

By design, w10n supports both read and write. However, our current discussion will be limited to read only w10n, because it is the primary use case in Earth Science.

2.1 The Idea

W10n holds a simple abstract tree view for a data store. In this tree, there are two types of entities: (1) node, which can contain sub-nodes and leaves. (2) leaf, which can only belong to one parent node and may hold data. A leaf is terminal.

A node or leaf always has meta information, describing what it contains. A leaf can also have data information attached, holding the bulk of content.

At its core, w10n treats inner components of a data store as nodes or leaves and make their meta or data information directly addressable and accessible through meaningful URLs.

2.2 URL Syntax

W10n uses distinctive URLs to pinpoint meta or data information of an entity:

(a) a trailing forward slash "/" denotes **meta** info,
(b) a trailing pair of square brackets "[…]" denotes **data** info.

W10n **meta** info URL has a form as

http://host:port/path/store/.../entity/?query_string

and w10n **data** info URL as

http://host:port/path/store/.../entity[selector]?query_string

in which selector denotes a portion of data information. In both forms, query_string indicates how information should be retrieved.

Sometimes, it is useful to group several entities into a virtual node, using a pair of curly brackets "{…}". More specifically, the resulting virtual node will have a name as {entity0,entity1,…}. Its **meta** info URL becomes

http://host:port/path/store/.../{entity0,entity1,...}/?query_string

and **data** info URL

http://host:port/path/store/.../{entity0,entity1,…}/[selector]?query_string

Note that w10n concepts of grouping and selecting are similar to the ones of projection and selection in the Structured Query Language (SQL).

2.3 Request

W10n supports all standard Hypertext Transfer Protocol (HTTP) methods: GET, PUT, POST, etc. Since read-only is the primary use case considered here, from now on, our discussion will be limited to HTTP GET and/or POST methods.

2.4 Response

By default, HTTP response of w10n meta info request is always in JSON format. Response of w10n data info can be in JSON or other formats applicable in a particular domain such as will be described in section 3.

Expressed in JSON, the response of meta or data info request will be in one of the following forms:

Meta info of a node

```
{'name':string,
 'attributes':[...],
 'nodes':[...],
 'leaves':[...],
 'w10n':[...]}
```

Meta info of a leaf

```
{'name':string,
 'attributes':[...],
 'type':string,
 'w10n':[...]}
```

Data info of a leaf

```
{'name':string,
 'data':...,
 'w10n':[...]}
```

Please note that w10n response uses a fixed set of vocabulary. Furthermore, a unique structure has been chosen for w10n response to ensure maximal flexibility in representing information.

Section 3
Webification for Science

A scientific data store (e.g., HDF) typically contains many multi-dimensional numeric arrays and is often organized in a group-based hierarchy. “Webification for Science” is a special form of w10n for scientific data stores. It is often shortened as **w10n-sci**.

W10n-sci extends the base w10n, described in Section 2 above. It introduces a few mandatory members in meta information and defines a variety of data info selectors, for convenient access and manipulation of numerical arrays.

With w10n-sci, leaves are just arrays, so we will use word “leaf” and “array” interchangeably in our discussion below.

3.1 URL Syntax

3.1.1 Node

Meta info URL takes the form

http://host:port/path/store/.../node/?output=json

3.1.2 Leaf (Array)

Meta info URL takes the form

http://host:port/path/store/.../array/?output=json

Data info URL takes the form

http://host:port/path/store/.../array[selector]?output=format

in which output format can be JSON or one of the formats common in science domain such as netCDF.

The selector can be one of the following types,

(a) An empty string “”

It indicates that the entire array is requested.

(b) A range: start0:stop0:step0, start1:stop1:step1, …

This indicates that only a subset of data points within prescribed ranges is requested. Please note that indices are 0-based.

(c) A list: [i0,i0,i2,…]

This indicates only individual data points at prescribed indices are requested.

Please note that indices are 0-based.

(d) A tile: (x0,x1,x2,…)s0*s1*s2*…

This indicates a subset of data points, starting at origin (x0,x1,x2…) and within a span of s0*s1*s2*…, is requested.

Please note that indices are 0-based.

(e) A logical condition or a combination of them

This uses array-based logical conditions to select a subset of data points.

Please see Section 5 for some examples.

3.1.3 Virtual Node

Meta info URL has the following form

http://host:port/path/store/.../{array0,array1,...}/?output=json

Data info URL has the following form

http://host:port/path/store/.../{array0,array1,…}/[selector]?output=format

where output can be JSON and one of applicable formats. And selector can be one of the types listed in Section 3.1.2 above.

3.2 Request

Since read-only is the primary use case considered here, only HTTP GET and/or POST methods are supported.

3.3 Response

For a w10n-sci node, the meta info response is

```
{'name':string,
 'attributes':[...],
 'nodes':[...],
 'leaves':[...],
 'w10n':[...]}
```

For a w10n-sci leaf (array), the meta info response is

```
{'name':string,
 'attributes':[...],
 'type':string,
 'shape':[...],
 'w10n':[...]}
```

and data info response is

```
{'name':string,
 'data':[...],
 'w10n':[...]}
```

Comparing with meta info response of base w10n leaf in Section 2.4 above, there is one additional member in w10n-sci leaf meta info response, called “shape”. Its value specifies the order and size of array dimensions. Furthermore, the value of member “type” in w10n-sci leaf (array) meta info response can be one of many ordinary numerical data types, such as “int8”, “int16”, “float32”, “float64”, etc.

Section 4
Servers and Clients

4.1 Servers

Various server-side implementations of w10n-sci exist for different scientific disciplines. In Earth science domain the known implementations are Pomegranate and Juneplum.

4.1.1 Pomegranate

It is one of our reference implementations of w10n-sci in Python and supports data formats commonly used in Earth science: HDF 4 and 5, all versions of netCDF, etc. It is implemented as a WSGI application for the Apache web server.

Pomegranate is open source software. The official site is http://pomegranate.nasa.gov

4.1.2 Juneplum

It is our special implementation of w10n-sci in Java. It webifies remote DAP service as a data store. It is implemented as a servlet for web service engines like Tomcat. It can be deployed as a drop-in to the host of an existing Hyrax instance and supports any data format the Hyrax supports. Hyrax is a reference implementation of DAP from opendap.org. Juneplum does not support the full specification of w10n-sci, due to current limitation of DAP. There are more detailed discussions about DAP and w10n-sci in Section 7 below.

Juneplum has been open sourced recently. An official site will be established in future.

4.2 Clients

With w10-sci, any piece of information from a science data store can be directly accessed, through a unique and meaningful URL. A user can run command line tools like *cURL* or *wget* to retrieve even a single byte of data in one HTTP call.

W10n-sci is designed to work well with all kinds of client environments, ranging from sophisticated modern web browser to less web-savvy scientific computing systems like Matlab and IDL.

A full discussion of known w10n-sci clients is beyond the scope of this white paper. However, we would like to end this section by pointing readers to xGlobe, a full-fledged, interactive, in-browser, w10n-sci web client. It can be used by scientists and enthusiasts to explore and analyze products from NASA Earth science missions as well as model outputs. This tool is capable of visualizing data in two modes: 2D map and 3D globe. Interested users are welcome to explore it at http://xGlobe.jpl.nasa.gov.

Section 5
Earth Science Examples

In this section, we will use a product file from NASA's Soil Moisture Active Passive (SMAP) mission,

```
SMAP_L3_SM_P_20150418_R11400_001.h5
```

to illustrate the application programming interface (API) of w10n-sci web service.

Server software used is Pomegranate version 2.1.5.

5.1 URL Syntax

The SMAP data file is in HDF 5 format, in which data arrays are organized in a two level hierarchy. Using w10n-sci syntax described in Section 3 above, we will demonstrate how data arrays and their subsets inside this file can be semantically addressed and directly retrieved via meaningful URLs. Please note that long URLs are forced to fold within page width and they should be read as single lines. Also, host and port portions in the URLs are omitted using ellipses (…) in order to save space.

5.1.1 Meta Information

a. *Single array*

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/soil_moisture/?output=json

This URL identifies meta info of array "soil_moisture" under group "Soil_Moisture_Retrieval_Data". Request using this URL will return meta info in JSON of the array. From that, we learn it is a 2-d array of shape [406,964]. Please check subsection 5.3.1-a below for the response.

b. *Multiple arrays*

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/{soil_moisture,surface_temperature,surface_flag}/?output=json

This URL identifies meta info of three arrays, together as a w10n virtual node "{soil_moisture,surface_temperature,surface_flag}". Request using this URL will return

combined meta info in JSON for all three arrays. Please check subsection 5.3.1-b below for the response.

5.1.2 Data Information

a. *Single array*

The following URLs illustrate various types of data info selectors.

(1) Empty string as selector

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/
/soil_moisture[]?output=json

This URL identifies data info of the entire "soil_moisture" array. Request using this URL will return all values.

(2) Range selector

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/
soil_moisture[0:406:2,0:964:3]?output=json

This ULR identifies data info of a portion of array "soil_moisture". Request using this URL will return only down-sampled values at locations defined by range 0:40:2,0:964:3.

(3) List selector

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/
soil_moisture[[0,10,123,2212,27154,194399]]?output=json

This URL identifies data info of a portion of array "soil_moisture". Request using this URL will return only data values at location [0,10,123,2212,27154,194399], as if the 2-d array is flattened to 1-d.

(4) Tile selector

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/
soil_moisture[(100,200)30x40]?output=json

This URL identifies data info of a portion of array "soil_moisture". Request using this URL will return only data values inside a slab of size 30x40 originated at (100,200).

(5) Conditional selector

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/
soil_moisture[40<=latitude<=45,95<=longitude<=100,surface_flag==34]?output=json

This URL identifies data info of a portion of array “soil_moisture”. Request using this URL will return values of data points only at locations where values of array “latitude” are between 40 and 45, values of array “longitude” are between 95 and 100, and values of array “surface_flag” are equal to 34.

Please check subsection 5.3.2-a below for its response.

b. *Multiple arrays*

http://.../SMAP_L3_SM_P_20150418_R11400_001.h5/Soil_Moisture_Retrieval_Data/{soil_moisture,surface_temperature,surface_flag,@}/[40<=latitude<=45,95<=longitude<=100,surface_flag==34]?output=json

This URL identifies data info of a combined portion of three arrays “soil_moisture”, “surface_temperature” and “surface_flag” together. Request using this URL will return values of data points of all three arrays, only at locations where values of array “latitude” are between 40 and 45, values of array “longitude” are between 95 and 100, and values of array “surface_flag” are equal to 34. Please note that a special array called “@” is also sought. It identifies locations of selected data points in original array.

Please check subsection 5.3.2-b below for its response.

5.2 Request

A client simply issues an HTTP GET call using one of URLs described above.

5.3 Response

Displayed below are respective HTTP GET responses for URLs in section 5.1 above. To assist in readability, only JSON responses are used. To save space, a large portion of numerical values have been omitted in responses using ellipses (…).

5.3.1 Meta Information

a. *Single array*

The response of a request using URL in section 5.1.1-a looks like


```
{
    "attributes": [
        {
            "name": "long_name",
            "value": "Representative soil moisture measurement for the
Earth based grid cell."
        },
        {
            "name": "units",
```

```
            "value": "cm**3/cm**3"
        },
        {
            "name": "coordinates",
            "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
        },
        {
            "name": "valid_min",
            "value": 0.019999999552965164
        },
        {
            "name": "valid_max",
            "value": 0.5
        },
        {
            "name": "_FillValue",
            "value": -9999.0
        }
    ],
    "shape": [
        406,
        964
    ],
    "type": "float32",
    "name": "soil_moisture",
    "w10n": [
        {
            "name": "spec",
            "value": "w10n-sci-1.2"
        },
        {
            "name": "application",
            "value": "pomegranate-2.1.5"
        },
        {
            "name": "type",
            "value": "hdf5"
        },
        {
            "name": "path",
            "value":
"/data/product/L3_SM_P/R11400/2015/04/18/SMAP_L3_SM_P_20150418_R11400_0
01.h5"
        },
        {
            "name": "identifier",
            "value": "/Soil_Moisture_Retrieval_Data/soil_moisture/"
        }
    ]
}
```


b. ***Multiple arrays***

The response of a request using URL in section 5.1.1-b looks like

```
{
    "leaves": [
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Representative soil moisture measurement
for the Earth based grid cell."
                },
                {
                    "name": "units",
                    "value": "cm**3/cm**3"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
                    "name": "valid_min",
                    "value": 0.019999999552965164
                },
                {
                    "name": "valid_max",
                    "value": 0.5
                },
                {
                    "name": "_FillValue",
                    "value": -9999.0
                }
            ],
            "shape": [
                406,
                964
            ],
            "type": "float32",
            "name": "soil_moisture"
        },
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Temperature at land surface based on
ECMWF or NCEP."
                },
                {
                    "name": "units",
                    "value": "Kelvins"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
```

```
                    "name": "valid_min",
                    "value": 0.0
                },
                {
                    "name": "valid_max",
                    "value": 350.0
                },
                {
                    "name": "_FillValue",
                    "value": -9999.0
                }
            ],
            "shape": [
                406,
                964
            ],
            "type": "float32",
            "name": "surface_temperature"
        },
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Bit flags that record ambient surface
conditions for the grid cell"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
                    "name": "_FillValue",
                    "value": 65534
                },
                {
                    "name": "flag_masks",
                    "value": "1s, 2s, 4s, 8s, 16s, 32s, 64s, 128s,
256s, 512s, 1024s, 2048s"
                },
                {
                    "name": "flag_meanings",
                    "value": "36_km_static_water_body_flag
36_km_radar_water_body_detection_flag 36_km_coastal_proximity_flag
36_km_urban_area_flag 36_km_precipitation_flag 36_km_snow_or_ice_flag
36_km_permanent_snow_or_ice_flag 36_km_radar_frozen_ground_flag
36_km_model_frozen_ground_flag 36_km_mountainous_terrain_flag
36_km_dense_vegetation_flag 36_km_nadir_region_flag"
                }
            ],
            "shape": [
                406,
                964
            ],
            "type": "uint16",
            "name": "surface_flag"
        }
```

```
    ],
    "nodes": [],
    "name": "{soil_moisture,surface_temperature,surface_flag}",
    "w10n": [
        {
            "name": "spec",
            "value": "w10n-sci-1.2"
        },
        {
            "name": "application",
            "value": "pomegranate-2.1.5"
        },
        {
            "name": "type",
            "value": "hdf5"
        },
        {
            "name": "path",
            "value":
"/data/product/L3_SM_P/R11400/2015/04/18/SMAP_L3_SM_P_20150418_R11400_0
01.h5"
        },
        {
            "name": "identifier",
            "value":
"/Soil_Moisture_Retrieval_Data/{soil_moisture,surface_temperature,surfa
ce_flag}/"
        }
    ]
}
```


5.3.2 Data Information

a. *Single array*

The response of a request using URL in section 5.1.2-a (5) looks like


```
{
    "data": [
        0.02203475683927536,
        0.01787199079990387,
        0.025918297469615936,
        0.023977292701601982,
        0.020461048930883408,
        …
        -9999,
        0.04255514591932297,
        0.029527390375733376,
        -9999,
        0.023646224290132523
    ],
    "w10n": [
        {
```

```
            "name": "spec",
            "value": "w10n-sci-1.2"
        },
        {
            "name": "application",
            "value": "pomegranate-2.1.5"
        },
        {
            "name": "type",
            "value": "hdf5"
        },
        {
            "name": "path",
            "value":
"/data/product/L3_SM_P/R11400/2015/04/18/SMAP_L3_SM_P_20150418_R11400_0
01.h5"
        },
        {
            "name": "identifier",
            "value":
"/Soil_Moisture_Retrieval_Data/soil_moisture[40<=latitude<=45,95<=longi
tude<=100,surface_flag==34]"
        }
    ]
}
```


b. ***Multiple arrays***

The response of a request using URL in section 5.1.2-b looks like


```
{
    "leaves": [
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Representative soil moisture measurement
for the Earth based grid cell."
                },
                {
                    "name": "units",
                    "value": "cm**3/cm**3"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
                    "name": "valid_min",
                    "value": 0.019999999552965164
                },
                {
```

```
                    "name": "valid_max",
                    "value": 0.5
                },
                {
                    "name": "_FillValue",
                    "value": -9999
                }
            ],
            "shape": [
                100
            ],
            "type": "float32",
            "name": "soil_moisture",
            "data": [
                0.02203475683927536,
                0.01787199079990387,
                0.025918297469615936,
                0.023977292701601982,
                0.020461048930883408,
                …
                -9999,
                0.04255514591932297,
                0.029527390375733376,
                -9999,
                0.023646224290132523
            ]
        },
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Temperature at land surface based on
ECMWF or NCEP."
                },
                {
                    "name": "units",
                    "value": "Kelvins"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
                    "name": "valid_min",
                    "value": 0
                },
                {
                    "name": "valid_max",
                    "value": 350
                },
                {
                    "name": "_FillValue",
                    "value": -9999
                }
            ],
            "shape": [
```

```
                100
            ],
            "type": "float32",
            "name": "surface_temperature",
            "data": [
                275.7823181152344,
                274.0382385253906,
                275.4599609375,
                274.9638366699219,
                274.2578125,
                …
                274.9777526855469,
                275.3252258300781,
                275.9293212890625,
                276.84619140625,
                277.36773681640625
            ]
        },
        {
            "attributes": [
                {
                    "name": "long_name",
                    "value": "Bit flags that record ambient surface
conditions for the grid cell"
                },
                {
                    "name": "coordinates",
                    "value": "/Soil_Moisture_Retrieval_Data/latitude
/Soil_Moisture_Retrieval_Data/longitude"
                },
                {
                    "name": "_FillValue",
                    "value": 65534
                },
                {
                    "name": "flag_masks",
                    "value": "1s, 2s, 4s, 8s, 16s, 32s, 64s, 128s,
256s, 512s, 1024s, 2048s"
                },
                {
                    "name": "flag_meanings",
                    "value": "36_km_static_water_body_flag
36_km_radar_water_body_detection_flag 36_km_coastal_proximity_flag
36_km_urban_area_flag 36_km_precipitation_flag 36_km_snow_or_ice_flag
36_km_permanent_snow_or_ice_flag 36_km_radar_frozen_ground_flag
36_km_model_frozen_ground_flag 36_km_mountainous_terrain_flag
36_km_dense_vegetation_flag 36_km_nadir_region_flag"
                }
            ],
            "shape": [
                100
            ],
            "type": "uint16",
            "name": "surface_flag",
            "data": [
                34,
                34,
```

```
                34,
                34,
                34,
                …
                34,
                34,
                34,
                34,
                34
            ]
        },
        {
            "data": [
                [
                    59,
                    740
                ],
                [
                    59,
                    741
                ],
                [
                    60,
                    740
                ],
                [
                    60,
                    741
                ],
                [
                    60,
                    742
                ],
                …
                [
                    71,
                    745
                ],
                [
                    71,
                    746
                ],
                [
                    71,
                    747
                ],
                [
                    71,
                    748
                ],
                [
                    71,
                    749
                ]
            ],
            "shape": [
                100,
```

```
                2
            ],
            "type": "int32",
            "name": "at_sign"
        }
    ],
    "nodes": [],
    "name": "{soil_moisture,surface_temperature,surface_flag,@}",
    "w10n": [
        {
            "name": "spec",
            "value": "w10n-sci-1.2"
        },
        {
            "name": "application",
            "value": "pomegranate-2.1.5"
        },
        {
            "name": "type",
            "value": "hdf5"
        },
        {
            "name": "path",
            "value":
"/data/product/L3_SM_P/R11400/2015/04/18/SMAP_L3_SM_P_20150418_R11400_0
01.h5"
        },
        {
            "name": "identifier",
            "value":
"/Soil_Moisture_Retrieval_Data/{soil_moisture,surface_temperature,surfa
ce_flag,@}/[40<=latitude<=45,95<=longitude<=100,surface_flag==34]"
        }
    ]
}
```

Section 6
Comparison with OGC Web Service

The Open Geospatial Consortium produces many open standards for geospatial web services, data processing and sharing. They cover a wide range of technical topics. Since w10n-sci is primarily about web-based data service, one shall only compare it with relevant OGC standards.

Furthermore, due to the limitation of space, this white paper will only discuss similarities and differences between w10n-sci and the Web Map Service (WMS), OGC's leading standard.

6.1 Similarities

In this section, after main capabilities of WMS are presented, we will discuss how to use w10n-sci to achieve the same functionalities.

6.1.1 WMS

Listed below are the two mostly used WMS requests:

a. *GetCapabillity request*

http://host:port/wms?REQUEST=GetCapabilites&servie=WMS&VERSION=1.1.1

This request will return information about what the wms instance can do.

b. *GetMap request*

http://host:port/wms?REQUEST=GetMap&SERVICE=WMS&VERSION=1.1.1&LAYERS=layer0,layer1&STYLES=style0,style1&SRS=srs&BBOX=minX,minY,maxX,maxY&WIDTH=w&HEIGHT=h&FORMAT=format

This request will return a map that is a combination of two layers, each of which is drawn in its associated style. It has a spatial span [minX, maxX] and [minY, maxY], using specified spatial reference system (SRS). The generated map is of size [w, h] and delivered in requested format.

6.1.2 W10n-sci

In the view of w10n-sci, a map store exposed by a WMS instance can be considered as a tree. There are different ways to webify such as a store. For the convenience of our discussion below, we will use this tree view: the store has a hierarchy of three levels. At root level is a container that holds a list of map layers as w10n nodes. Under each node (layer), there are a few w10n

leaves, one of which holds the map data information for that layer and has a literal name called “data”, and the others keep information of associated styles.

Expressed in JSON, the tree looks like

```
{
    "attributes":[...],
    "nodes":[
        {
            "name":layer0,
            "attributes":[...],
            "nodes":[],
            "leaves":[
                {
                    "name":"data",
                    "attributes":[...]
                },
                {
                    "name":style0,
                    "attributes":[...]
                },
                {
                    "name":style1,
                    "attributes":[...]
                },
            ]
        },
        {
            "name":layer1,
            "attributes":[...],
            "nodes":[],
            "leaves":[
                {
                    "name":"data",
                    "attributes":[...]
                },
                {
                    "name":style0,
                    "attributes":[...]
                },
                {
                    "name":style1,
                    "attributes":[...]
                },
            ]
        },
        ...
    ],
    "leaves":[],
    "w10n":[...]
}
```

Listed below are sample URLs illustrating how individual components of the map store can be accessed.

(1) http://host:port/mapStore/?output=json

Returns top meta info about the store, similar to WMS's GetCapbilities call in section 6.1.1-a above.

(2) http://host:port/mapStore/layer0/?output=json

Returns meta info about a particular layer, namely, layer0.

(3) http://host:port/mapStore/layer0/{data,style0}/[minX<=srs:x<=maxX,minY<=srs:y<=maxY]?width=w&height=h&output=format

Returns data info for layer0, in style0, for a spatial area defined by minX, maxX, minY, maxY of a spatial reference system, of size w by h and in requested format.

(4) http://host:port/mapStore/{layer0/{data,style0},layer1/{data,style1}}/[minX<=srs:x<=maxX,minY<=srs:y<=maxY]?width=w&height=h&output=format

Returns a map that is a combination of two layers, each of which is drawn in its associated style. It has a spatial span [minX, maxX] and [minY, maxY], using specified spatial reference system (SRS). The generated map is of size w by h and delivered in requested format. This is exactly what 6.1.1-b does above.

6.2 Differences

Standards from OGC are mostly about geo-spatial data, service and processing. W10n-sci is not limited to the application of geo-spatial information. It can handle equally well all types of data from different scientific disciplines. Though w10n-sci is not geo specific, it is geo capable, as we have demonstrated in previous section.

A map is two-dimensional. That means OGC data standards inevitably have a focus on 2-d arrays. In comparison, multi-dimensional array handling is native to w10n-sci. This makes it more suitable for use in situations where data are not necessarily geo-spatial and of higher dimensions. The advantages of w10n-sci in this aspect have been shown in our work conducted under commonly used scientific computation environments such as Matlab and IDL.

As a technology for data virtualization, w10n-sci attempts to close the gap between domain specificities of data and their potential uses. Furthermore, w10n is primarily concerned with the exposure of one class of digital resources, i.e., data. The other class of digital resources such as tools and libraries is handled by servicification (serv10n), a sister technology of w10n, which should be comparable to some OGC standards dealing with data processing.

Section 7
Comparison with OPeNDAP DAP Service

In Earth science, one frequently used remote data access technology is the Data Access Protocol (DAP) from OPeNDAP. Hyrax is the leading implementation of DAP. To help our discussion here, we would like to make a clear distinction among the following words: (a) “DAP”, a data service protocol (b) Hyrax, an implementation of DAP (c) OPeNDAP, the organization behind both. From time to time, they have been used interchangeably, causing unnecessary confusions in the community.

In this section, we will examine similarities and differences between DAP and w10n-sci.

7.1 Similarities

7.1.1 DAP

Listed below are DAP requests for accessing various parts of a science data file *sst.mnmean.nc.gz*, taken directly from the QuickStart guide at http://docs.opendap.org.

(0) http://host:port/dap/data/nc/sst.mnmean.nc.gz.dds

It returns the data file’s Dataset Descriptor Structure (DDS), which provides a description of the “shape” and “type” of the arrays inside. From that, one learns that, the file, as a dataset, consists of two different pieces:

- A “Grid” containing a three-dimensional array of integer values (Int16) called sst, and three "Map" vectors: a 89-element vector called “lat”, a 180-element vector called “lon” and 1857-element evtor called “time”.
- A 1857 by 2 array called “time_bnds”.

(1) http://host:port/dap/data/nc/sst.mnmean.nc.gz.das

It returns the data file’s Data Attribute Structure (DAS), which is similar to the DDS above, but contains more information about arrays, such as units, long name, etc.

(2) http://host:port/dap/data/nc/sst.mnmean.nc.gz.ascii?lat

It returns all values of array lat in ASCII as comma separated values.

(3) http://host:port/dap/data/nc/sst.mnean.nc.gz.ascii?sst[0:1:1][13:1:16][103:1:105]

It returns a subset of array sst, also in ASCII as comma separated values.

(4) http://host:port/dap/data/nc/sst.mnean.nc.gz.ascii?geogrid(sst,62,206,56,210,”19722<time 19755”)

It returns a subset of array sst, within bounding box [62,206,56,210] and for a time period from 19722 to 19755, in ASCII as comma separated values.

In (2) – (4) above, replacing string “ascii” in URL with “nc” or “dods” would return data in netCDF or DAP binary format.

7.1.2 W10n-sci

Using w10n-sci, the file would be treated as a tree with a list of arrays as w10n leaves. Listed below are example request URLs for addressing and accessing various inner components.

(1) http://host:port/dap/data/nc/sst.mnmean.nc.gz/?output=json

It returns meta information of the file as a tree. The content contains the same information as 7.1.1 (1) above, but in a tree response form and in JSON format.

(2) http://host:port/dap/data/nc/sst.mnmean.nc.gz/lat[]?output=json

It returns all values of array lat in JSON format, comparable to 7.1.1 (2) above.

(3) http://host:port/dap/data/nc/sst.mnean.nc.gz/sst[0:1:1,13:16:1,103:105:1]?output=json

It returns a subset of array sst in JSON format, comparable to 7.1.1 (3) above.

(4) http://host:port/dap/data/nc/sst.mnean.nc.gz/sst[56<=lat<=62,206<lon<=210,19722<=time<=19755]?output=json

It returns a subset of array sst, within spatial region lat:[56,62], lon:[206,210] and for a time period from 19722 to 19755, in JSON format, comparable to 7.1.1 (4) above.

In (2) – (4) above, replacing string “json” in URL with “nc” or “be” or “le” would return data in netCDF or pure big/little-endian binary.

7.2 Differences

For Earth science, both DAP and w10n-sci can be used for remote piece-wise array retrieval. However they differ greatly in design philosophy and technical approach. DAP has a focus on data types or structures common in geo science. W10n-sci, as a special application of w10n, emphasizes on the hierarchical organization of data and has the flexibility for uniform representing of both geo and non-geo science data.

Summarized in the table below are some technical differences.

Technology	DAP/Hyrax	W10n-sci/Pomegranate
data organization	flat	hierarchical

data model	common data types in geo science	arrays in a hierarchy
service ReSTfulness	*	emphasis on resource
URL syntax	less user-friendly	semantic and user-friendly
client requirement	library needed for each language or environment	no special requirement
HTML5 readiness	no	yes
applicability	geo science	multi-disciplinary

Please note that there is usually a gap between what is defined in specification and what is implemented. For this reason, our comparison is conducted using leading implementations for both technologies that have been deployed in real operations. For DAP, it is Hyrax. For w10n-sci, it is Pomegranate.

A. ACRONYMS

Acronym	Expansion
API	Application Programing Interface
ASCII	American Standard Code for Information Interchange
DAP	Data Access Protocol
DAS	Data Attribute Structure
DDS	Dataset Descriptor Structure
ESDIS	Earth Science Data and Information System
HDF	Hierarchical Data Format
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
netCDF	network Common Data Form
OGC	Open Geospatial Consortium
OPeNDAP	Open-source Project for a Network Data Access Protocol
ReST	Representation of State Transfer
SMAP	Soil Moisture Active Passive
URL	Uniform Resource Locator
w10n	Webification
w10n-sci	Webification for science
WMS	Web Map Service

B. ACKNOWLEDGEMENTS

This white paper was developed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.