Routing Service Specification - v1.1

Release 1.1.1

Javier Quinteros and the EIDA team

November 10, 2017

CHAPTER

ONE

OVERVIEW OF THE SERVICE

Purpose

To provide the URL(s) where a client can request information to a service for a specific stream and in a particular time window.

Conceptual Architecture

This service provides routing information for distributed data centers, in the case where multiple different seismic data centres offer access to data and products using compatible types of services. Examples of the data and product objects are seismic timeseries waveforms, station inventory, or quality parameters from the waveforms. The European Integrated Data Archive (EIDA) is an example of a set of distributed data centres (the EIDA „nodes“). EIDA have offered Arclink and Seedlink services for many years, and now offer FDSN web services, for accessing their holdings. In keeping with the distributed nature of EIDA, these services could run at different nodes. Depending on the type of service, these may only provide information about a reduced subset of all the available waveforms.

To assist users or clients to locate data, we have designed a Routing Service, which could run at EIDA nodes or elsewhere. This (meta)service is queried by clients (or other services) and provides the address(es) where, within the context of EIDA, the desired information is provided.

To be effective, the Routing Service must know the locations of all services integrated into a system (e.g. EIDA) and serve this information in order to help the development of smart clients and/or services at a higher level, which can offer the user an integrated view of the entire system (EIDA), hiding the complexity of its internal structure.

The service is intended to be open and able to be queried by anyone without the need of credentials or authentication.

Services could be classified or located at different layers, depending on the type and amount of information they provide, if it is aggregated or not, etc. In the context of this document we will only take into account the services running at the "node level". Namely, services that provide partial information related to a node within EIDA: these services only offer content related to data which is local to each node. Services that provide already aggregated information are not considered herein.

General Features

We prioritize some key concepts common to any good design process in software engineering when designing the Routing Service. Some of them are:

- Simplicity: Provide an atomic, simple and reduced business process.
- Consistency with standards: Parameter names, formats and types follow similar existing services (e.g. FDSN web services) whenever possible.
- Stateless: The service will neither create nor store information related to sessions, connections or results, beyond that needed to check that it is operating correctly.

- Platform independence: The interoperability between different services/ clients is mandatory and there are no assumptions made concerning either on the underlying software implementations, or on the hardware used.
- Traffic friendly: The service should constrain to a minimum the amount of information sent/received.
- Detailed Errors: The service must return a comprehensive and detailed description of the error(s) found.

Service path and port

The following base URL pattern is to be used for the deployment of the service:

```
<site>/<relative-path>/routing/<majorversion>/
```

where relative-path is optional (it will be necessary when the service does not run at the top level inside the site) and *majorversion* is an integer value that determines the major specification version supported by the service. An example could be:

```
http://geofon.gfz-potsdam.de/eidaws/routing/1/
```

It is suggested that the service should be available for TCP/IP connections at port 80, but it could also work on non-standard ports.

CHAPTER TWO

AVAILABLE METHODS

Four methods are defined following the FDSN style: *application.wadl*, *version*, *query* and *info*. The first three methods behave in a similar way as in the standard FDSN web services (http://fdsn.org/webservices) and are described in detail in the following sections. The *info* method is intended to provide human-readable information about the content provided by the service (see Section 5.4).

Description of the service

The *application.wadl* method shall return a WADL conformant description of the interface using the MIME type *application/xml*. Any parameters submitted with the method will be ignored. The WADL shall describe all parameters supported by the interface and is primarily used to document the service and whether some optional parameters are supported.

Version of the implementation

The *version* method shall return the implementation version as a simple text string using the MIME type *text/plain*. Any parameters submitted with the method will be ignored. This scheme follows the FDSN web services approach.

The service is versioned according the following three-digit (x.y.z) pattern:

```
SpecMajor.SpecMinor.Implementation
```

where the fields have the following meaning:

- SpecMajor: The major specification version, all implementations sharing this value will be backwards compatible with all prior releases. Values start at 1.
- SpecMinor: The minor specification version, incremented when optional parameters or behavior is added to the previous specification but backwards compatibility is maintained with the previous major versions, i.e. all 1.y.z service versions will be compatible with version 1.0. Values are integers starting at 0.
- Implementation: The implementation version, an integer identifier specific to the data center implementation. Useful to track service updates for bug fixes, etc. but with no implication on conformance to the specification.

Together the *SpecMajor* and *SpecMinor* versions imply a minimum expected behaviour of a given service. This versioning scheme allows clients to expect specific behaviour based on the *SpecMajor* version, while allowing the extension of the service with optional parameters while maintaining backwards compatibility. Each version number is service specific, there is no implication that *SpecMajor* version numbers across services (from EIDA or FDSN) are related.

Querying the service

The *query* method is how the users access the main functionality of the service. Both *GET* and *POST* methods must be supported.

Input parameters

The complete list of input parameters can be seen in Table *Table 1*. Parameter names must be in lower case, and may be abbreviated as shown in the first column, following the FDSN style. Valid input values must have the format shown in the "Format" column. All the values passed as parameters will be case-insensitive strings composed of numbers and letters. No other symbols will be allowed with the exception of:

1. wildcards ('*' and '?'), which may be used to select the streams (for parameters *network*, *station*, *location* and *channel* only),
2. the symbols ':', '-' (minus) and '.', which are specified in the ISO 8601 format for dates (*starttime* and *endtime*), and
3. the string '–' (two minus symbols), which may be used only for the *location* parameter.

For the wildcards accepted in the case of *network*, *station*, *location* and *channel*. The character '*' matches any value, while '?' matches any character. For any of these four parameters, if no value is given it will be set to a star ('*').

Blank or empty location identifiers may be specified as "–" (two dashes) if needed, which the service must translate to an empty string.

Table 2.1: Input parameters description

Parameter	Support	Format	Description	Default
starttime (start)	Required	ISO 8601	Limit results to routes valid on or after the specified start time.	Any
endtime (end)	Required	ISO 8601	Limit results to routes valid on or before the specified end time.	Any
network (net)	Required	char	Select one network code. This can be either SEED network codes or data center defined codes.	*
station (sta)	Required	char	Select one station code.	*
location (loc)	Required	char	Select one location identifier. As a special case "–" (two dashes) will be translated to an empty string to match blank location IDs.	*
channel (cha)	Required	char	Select one channel code.	*
minlatitude	Required	float	Limit to stations with a	
(minlat)			latitude larger than or equal to the specified minimum.	-90
maxlatitude	Required	float	Limit to stations with a	
(maxlat)			latitude smaller than or equal to the specified maximum.	90
minlongitude	Required	float	Limit to stations with a	
(minlon)			longitude larger than or equal to the specified minimum.	-180
maxlongitude	Required	float	Limit to stations with a	
(maxlon)			longitude smaller than or equal to the specified maximum.	180
service	Required	char	Specify which service will be queried (dataselect, station, etc).	dataselect
format	Required	char	Select the output format. Valid values for any service are: xml and json, while get and post are optional and only available for FDSN web services.	xml
alternative	Optional	boolean	Specify if the alternative routes should be also included in the answer (in the case that these are available). Route order is indicated by the *priority* attribute.	false

Output description and format

At least two different output formats must be supported by any implementation (*xml*, *json*), while two more formats are recommended (*get*, *post*). The structure of the information returned is different with each format type. In case of a successful request the HTTP status code will be 200, and the response will be as described below for each format.

Output: XML format

This is the default selection if the parameter *format* is not specified or if it is given with an *xml* value. The MIME type must be set to *text/xml*. The following is an example of the expected XML structure. The *params* element will be repeated as many times as necessary inside the *datacenter*. Each *datacenter* element must contain:

- exactly one *url* element, specifying the URL of the service at a given data centre,
- exactly one *name* element, which gives the name of the service, and
- a list of *params* elements, each describing a stream, or set of streams by using appropriate wildcarding, available using the service at that URL. The *params* element may be repeated as many times as necessary inside the *datacenter* element.

For instance,

```
<service>
  <datacenter>
    <url>http://ws.resif.fr/fdsnws/dataselect/1/query</url>
      <params>
        <loc>*</loc>
        <end>2012-04-20T23:59:00</end>
        <sta>KES28</sta>
        <cha>*</cha>
        <priority>1</priority>
        <start>2011-09-15T00:00:00</start>
        <net>4C</net>
      </params>
    <name>dataselect</name>
  </datacenter>
</service>
```

Output: JSON format

If the format parameter is *json*, the information will be returned with MIME type *text/plain*. The content will be a JSON (Java Script Object Notation [1]) array, in which each element is a JSON object corresponding to a *datacenter* element in the XML format shown above. For the example response above, this would appear as[#f2]_:

```
[{"url": "http://ws.resif.fr/fdsnws/dataselect/1/query", "params": [{"loc":
"*", "end": "2012-04-20T23:59:00", "sta": "KES28", "cha": "*", "priority": 1,
"start": "2011-09-15T00:00:00", "net": "4C"}], "name": "dataselect"}]
```

It should be noted that the value associated with *params* is an array of objects and that there will be as many objects as needed for the same datacenter.

Output: GET format

[1] See *"Introducing JSON"*, http://json.org; and ECMA International, *"The JSON Data Interchange Format"*, ECMA-404, 1st edition, October 2013.

Note: This option is not mandatory, but due to practical reasons is highly recommended. It should be noted that it could not make sense for some services, but it will be very helpful for services using a similar URI construction as the FDSN-WS (e.g. dataselect and station).

When the *format* parameter is set to *get*, the output will be declared as *text/plain* and will consist of one URL per line. This option will only be available with known services where the usage of a URL via the GET method is allowed (e.g. FDSN web services). The URLs will be constructed in a way that they can be used directly by the client to request the necessary information without the need to parse them. For instance:

```
http://ws.resif.fr/fdsnws/dataselect/1/query?sta=KES28&net=4C&start=...
```

Output: POST format

Note: This option is not mandatory, but due to practical reasons is highly recommended. It should be noted that it could not make sense for some services, but it will be very helpful for services using a similar URI construction as the FDSN-WS (e.g. dataselect and station).

If *format* is *post*, the output will be declared as *text/plain* and the structure will consist of:

- a line with a URL where the request must be made,
- a list of lines with the format declared in the FDSN-WS specification to do a POST request[#f3]_.

This option will only be available with known services where the usage of a URL is allowed (e.g. FDSN web services). If the request should be split in more than one datacenter, the blocks for every datacenter will be separated by a blank line and the structure will be repeated (URL and POST body).

```
http://ws.resif.fr/fdsnws/dataselect/1/query
4C KES28 * * 2011-09-15T00:00:00 2012-04-20T23:59:00
```

Alternative routes

If the *alternative* parameter is set, the service will return all the routes that match the requested criteria without filtering them by priority. Alternative routes are indicated by the *priority* attriute. The client will be required to interpret the priority of the routes and to select the combination of routes that best fits their needs to request the information. The client needs also to take care of checking the information to detect overlapping routes, which will definitely occur when a primary and an alternative route are being reported for the same stream.

Note: As a rule of a thumb and in a normal case, the alternative addresses should only be used if there is no response from the authoritative data centre (priority=1).

It should be noted that the for *get* and *post* format outputs, *priority* cannot be returned. Additional parsing of other formats are required to interpret priority.

How to pass the parameters

GET: the parameters must be given in *key=value* pairs separated by '&'.

```
http://server_url?key1=value1&key2=value2
```

POST: The first lines could be used to pass the parameters not related to streams or time windows (*service*, *format*, *alternative*) with one *key=value* clause per line. For instance,

```
service=station
```

For the six parameters used to select streams and timewindows, one stream+timewindow is expected per line and the format must be:

```
NET STA LOC CHA START END
```

If there is no defined time window, an empty string should be given as ‘’ or “” (see the section with to get a better understanding of the details)

Abnormal responses

In addition to a *200 OK* status code for a successful request, other responses are possible, as shown in Table Table 2. These are essentially the same as for FDSN web services. Under error, maintenance or other unusual conditions a client may receive other HTTP codes generated by web service containers, and other intermediate web technology.

Table 2.2: HTTP status codes returned by the Routing service

Code	Description
200	OK, Successful request, results follow.
204	Request was properly formatted and submitted but no data matches the selection.
400	Bad request due to improper specification, unrecognized parameter, parameter value out of range, etc.
413	Request would result in too much data being returned or the request itself is too large. Returned error message should include the service limitations in the detailed description. Service limits should also be documented in the service WADL.
414	Request URI too large
500	Internal server error
503	Service temporarily unavailable, used in maintenance and error conditions

Information about the content of service

When the method *info* is invoked, a description about the information handled by the Routing Service should be returned. The answer must be of MIME type *text/plain* and is actually a text-free output. However, in the first lines it is expected to be specified which information we can find by querying the service. For instance,

```
All Networks from XYZ institution
Stations in Indonesia
Stations in San Francisco

Other comments and descriptions that could be of interest of the user.
```

Any parameter passed to this method will be ignored.

CHAPTER

THREE

EXAMPLES

In the first five examples, the *service*, *starttime* and *endtime* parameters take their default values. Namely, *dataselect* for the service and *Any* for start and end time. Here we show requests to a hypothetical Routing Service deployment running at *server.org* [4].

1. Where do I get waveforms via dataselect from the station APE belonging to network GE?

http://server.org/eidaws/routing/query?net=GE&sta=APE

Answer:

```
<service>
  <datacenter>
    <url>http://geofon.gfz-potsdam.de/fdsnws/dataselect/1/query</url>
    <params>
      <loc>*</loc>
      <end/>
      <sta>APE</sta>
      <cha>*</cha>
      <priority>1</priority>
      <start>1993-01-01T00:00:00</start>
      <net>GE</net>
    </params>
    <name>dataselect</name>
  </datacenter>
</service>
```

Waveforms from station APE are available at GFZ. Wildcards are set for location and channel parameters. The *end* attribute is undefined because this is a permanent station.

2. Where do I get waveforms via dataselect from the HHZ channel of the station LIENZ belonging to network CH?

http://server.org/eidaws/routing/query?net=CH&sta=LIENZ&cha=HHZ

Answer:

```
<service>
  <datacenter>
    <url>http://eida.ethz.ch/fdsnws/dataselect/1/query</url>
    <params>
      <loc>*</loc>
      <end/>
      <sta>LIENZ</sta>
      <cha>HHZ</cha>
      <priority>1</priority>
      <start>1980-01-01T00:00:00</start>
      <net>CH</net>
    </params>
    <name>dataselect</name>
```

[4] In the server responses below, white space and blank lines have been added for readability, but servers do not need to include them.

```
  </datacenter>
</service>
```

Wildcard is set for *location*. HHZ stream from station LIENZ at network CH is available at ETH.

3. Where do I get waveforms via dataselect from the BHZ channel of the station LIENZ belonging to network CH?

http://server.org/eidaws/routing/query?net=CH&sta=LIENZ&cha=BHZ

Answer:

```
<service>
  <datacenter>
    <url>http://www.orfeus-eu.org/fdsnws/dataselect/1/query</url>
    <params>
      <loc>*</loc>
      <end/>
      <sta>LIENZ</sta>
      <cha>BHZ</cha>
      <priority>2</priority>
      <start>1980-01-01T00:00:00</start>
      <net>CH</net>
    </params>
    <name>dataselect</name>
  </datacenter>
</service>
```

Wildcard is set for *location*. BHZ stream from station LIENZ at network CH is available at ODC.

4. Where do I get waveforms via dataselect from the station LIENZ belonging to network CH, whose channel names match ”?HZ”?

http://server.org/eidaws/routing/query?net=CH&sta=LIENZ&cha=?HZ

Answer:

```
<service>
  <datacenter>
    <url>http://www.orfeus-eu.org/fdsnws/dataselect/1/query</url>
    <params>
      <loc>*</loc>
      <end/>
      <sta>LIENZ</sta>
      <cha>BHZ</cha>
      <priority>2</priority>
      <start>1980-01-01T00:00:00</start>
      <net>CH</net>
    </params>
    <name>dataselect</name>
  </datacenter>
  <datacenter>
    <url>http://eida.ethz.ch/fdsnws/dataselect/1/query</url>
    <params>
      <loc>*</loc>
      <end/>
      <sta>LIENZ</sta>
      <cha>HHZ</cha>
      <priority>1</priority>
      <start>1980-01-01T00:00:00</start>
      <net>CH</net>
    </params>
    <params>
      <loc>*</loc>
      <end/>
      <sta>LIENZ</sta>
      <cha>LHZ</cha>
      <priority>1</priority>
          <start>1980-01-01T00:00:00</start>
          <net>CH</net>
      </params>
      <name>dataselect</name>
  </datacenter>
</service>
```

Wildcards are set for *location* and *channel*. BHZ stream is available at ODC, while HHZ and LHZ are available at ETH.

5. Where do I get waveforms via dataselect from the BHZ channel of the station BZS belonging to network RO? The response should be provided in GET format.

http://server.org/eidaws/routing/query?net=RO&sta=BZS&cha=BHZ&format=get

Answer:

```
http://eida-sc3.infp.ro/fdsnws/dataselect/1/query?sta=BZS&cha=BHZ&net=RO
```

This stream can be obtained from NIEP and the response can be directly used to request the data without the need to parse or interpret it.

6. Where do I access the "generic" service for the BHZ channel of the station BZS belonging to network RO? The response should be provided in JSON format.

http://server.org/eidaws/routing/query?net=RO&sta=BZS&cha=BHZ&format=json&service=generic

Answer:

```
[{"url": "http://eida-sc3.infp.ro/fdsnws/dataselect/1/query", "params":
 [{"loc": "*", "end": "", "sta": "BZS", "cha": "BHZ", "priority": 1,
   "start": "1980-01-01T00:00:00", "net": "RO"}], "name": "generic"}]
```

The "generic" service for this stream can be accessed at NIEP.

7. Which address has the dataselect server who serves locally waveforms of the first hour of 2014 from the 5E network?

http://server.org/eidaws/routing/query?net=5E&service=dataselect&start=2014-01-01T00:00:00&end=2014-01-01T01:00:00

Answer:

```
Error 204 - No Content
```

No routes can be found, because the network was operational between 2011 and 2013.

8. Where do I get waveforms via dataselect from 2012-02-02 to 2012-03-02 from the 4C network? Give me the results ready to do the request via a POST method.

http://server.org/routing/query?net=4C&start=2012-02-02T00:00:00&end=2012-03-02T00:00:00&format=post

Answer [5]:

```
http://ws.resif.fr/fdsnws/dataselect/1/query
4C KES20 * HHE 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KES20 * HHN 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KES20 * HHZ 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KEA00 * * 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KEA01 * * 2012-02-02T00:00:00 2012-03-02T00:00:00

http://geofon.gfz-potsdam.de/fdsnws/dataselect/1/query
4C KES20 * HNE 2012-02-02T00:00:00 2012-03-02T00:00:00
```

[5] Only some lines are shown as an example for each data center. The real answer is much longer.

```
4C KES20 * HNN 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KES20 * HNZ 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KEB10 -- HHZ 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KEB10 -- HHN 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KEB10 -- HHE 2012-02-02T00:00:00 2012-03-02T00:00:00

http://webservices.rm.ingv.it/fdsnws/dataselect/1/query
4C KER02 * * 2012-02-02T00:00:00 2012-03-02T00:00:00
4C KES02 * * 2012-02-02T00:00:00 2012-03-02T00:00:00
```

The data from this network is distributed among the three institutions.