Datastream REST API

The following document describes the REST API provided by DSAPI2, as of Feb 22 2011.

The DSAPI2 REST service provides endpoints that allow clients to fetch data and metadata about DSAPI2 streams, as well as token data from the streams. In addition, the REST service can generate simple charts for numeric data, suitable for user-facing display.

Many of the REST calls provide output in more than one format. Supported formats include the following (note that not every endpoint supports every format):

- "json": Javascript Object Notation
- "xml": XML
- "html": HTML
- "csv": Comma-Separated Values
- "chart": Data chart (as image)
- "img": Frame image data
- "sos": SOS format
- "rdf": RDF/XML
- "n3": RDF Notation 3 ("N3")
- "rss": RSS 2.0
- "zip": ZIP archive of requested data

REST endpoints are in the following syntax:

```
/{request}/{param1}/{value1}/{param2}/{value2}/.../{paramN}/{valueN}
```

where "request" identifies the type of request being made, and "param" and "value" refer to parameters and their values. For example, the following endpoint is a "list" request, with a "format" parameter whose value is "xml":

```
/list/format/xml
```

most endpoints take a "uri" parameter which identifies the stream being queried. Note that any "/" characters in the URI need to be escaped; the exception to this requirement is if "uri" is the final parameter in the request. For example, the following "window" request is targeting the stream with the URI "urn: streams/snorb7/twitter":

```
/window/start/first/end/last/format/html/uri/urn:streams/snorb7/twitter
```

Here's an equivalent call:

```
/window/start/first/end/last/format/html/uri/urn%3Astreams%2Fsnorb7%2Ftwitter
```

Supported endpoints

List

List streams. Returns the URIs of all (or selected) streams. For HTML format, returns a list with human-readable names for each stream, which are hyperlinked to calls against the "window" endpoint.

Parameters:

1. **format** - desired result format (supported: html, xml, json)
2. **site** (optional) - list only the streams at the given "site" (value should be the URI of a site). A site is any arbitrary grouping of streams. Some fetchers associate multiple streams with a single site.

Example:

```
/list/format/xml
```

Window

Return all frames (optionally, with data) from a given stream within a given time range. The time range is specified with the **start** and **end** parameters. The values of these parameters are either timestamps represented as ms since the *nix epoch, or one of several other notations:

- **start** can be specified as "first", which means the earliest frame in the stream
- **end** can be specified as "last", which means the most recent frame in the stream
- either range bound can be specified as "Fn" where n is the number of frames worth of data to fetch relative to the other bound. For example /window/start/F2/end/last... means to return the last two frames of the stream, and /window/start/1298397372/end/F10... means to return the 10 frames following Tue 22 Feb 2011 11:56:12 AM CST.

Parameters:

1. **start** - start time
2. **end** - end time
3. **format** - format (supported: html, xml, json, csv, chart, img, sos, rdf, rss, zip)
4. **data** (optional) - set to "true" to return the data of each frame (may produce very large result)
5. **sample** (optional; if included "data" must be set to "true") - set this to n and the restlet will return every nth frame
6. **uri** - the URI of the stream

Example:

```
/window/start/1297791028800/end/last/format/xml/data/true/uri/http%3A%2F%2Fiacat-enviro.ncsa.illinois.edu%
2FEBIWeatherStations%2FNE%2FAvg15%23PAR_APOGE_Avg
```

Example output:

```
<?xml version="1.0" encoding="UTF-8"?><streams>
  <stream>
    <streamURL>http://iacat-enviro.ncsa.illinois.edu/EBIWeatherStations/NE/Avg15#PAR_APOGE_Avg</streamURL>
    <frames>
      <frame>
        <time>1297791900000</time>
        <dateTime>Tue Feb 15 11:45:00 CST 2011</dateTime>
        <data>1157</data>
      </frame>
      <frame>
        <time>1297792800000</time>
        <dateTime>Tue Feb 15 12:00:00 CST 2011</dateTime>
        <data>1020</data>
      </frame>
    </frames>
  </stream>
</streams>
```

When **data** is set to true, the format and encoding of the contents of the data field will depend on the MIME type associated with the stream. If the stream's major MIME type is "text", the data field will contain unicode text; for binary MIME types, the data field will be Base64-encoded. MIME types are associated with streams in the fetching stage, via a filter called TypeRegisterFilter.

Frame

Return a single frame of data from a stream. Note that the MIME type of the response is determined by the stream. If each frame is plain text, it will be set to text/plain; if each stream is a TIFF image, it will be set to image/tiff, etc.

Parameters:

1. **time** - the timestamp of the desired frame (in ms since the *nix epoch)
2. **uri** - the URI of the stream

Depending on the server configuration, this endpoint may require authentication.

Example:

```
/frame/time/1278614700000/uri/http://iacat-enviro.ncsa.illinois.edu/EBIWeatherStations/SE/Avg15%23WindSpd_Max
```

Range

Return the temporal range of the stream (i.e., the time of the earliest frame and the time of the most recent frame).

Parameters:

1. **format** - format (supported: json, xml)
2. **uri** - the URI of the stream

Example response:

```
[
  {
    "streamURL": "urn:streams/snorb/twitter",
    "start" : 1296859710000,
    "end" : 1297101208000
  }
]
```

Formats

Return the MIME type of the given stream.

Parameters:

1. **format** - format (supported: html, xml, json, csv)
2. **uri** - the URI of the stream

Example output (CSV):

```
stream,format
http://ncsa.edu/dsapi/twitter/cufire2,text/plain
```

Triples

Return all metadata about the given stream.

Parameters:

1. **format** - format (supported: rdf, n3)
2. **uri** - the URI of the stream

Example output (rdf):

```
<rdf:RDF>
  <a:DataStream rdf:about="http://ncsa.edu/dsapi/twitter/cufire2">
    <a:characterEncoding>UTF-8</a:characterEncoding>
    <dc:format>text/plain</dc:format>
    <dc:title>Twitter search for fire near CU</dc:title>
    <rdfs:label>Twitter search for fire near CU</rdfs:label>
  </a:DataStream>
</rdf:RDF>
```