Tethys RESTful Web Services Interface

A manual for accessing a Tethys server using a representational state transfer interface

Tethys, Antioch mosaic, 3rd century from Baltimore Museum of Art

Contents

Introduction

Tethys provides web services via a representational state transfer (REST, Fielding, 2000) interface. RESTful architectures identify resources via uniform resource locators (URLs) and provide operations to create, remove, update and delete items associated with the resource. It is assumed that the reader is familiar with Tethys and this work merely documents the interface that can be used by other programs to interact with Tethys.

Tethys provides a series of resources to the user:

- Attach – Document attachments, e.g. a spectrogram or short waveform.
- BatchLogs – Logs for batch operations, primarily used in rebuilding the database from source documents.
- XQuery – A resource to query the Tethys database.
- Collection resources – These resources represent individual Tethys collections. Current collection resources are:
    - Detections
    - Deployments
    - Events
    - Localizations
    - mediator_cache
    - SourceMaps
    - ITIS
    - SourceMaps
    - SpeciesAbbreviations

An example resource for a server running on port 9779 (the default port) accessed from the server machine would be: http://localhost:9779/XQuery. Examples throughout this document will use localhost as the server, these should be updated with the desired server's address.

Clients can use hypertext transfer protocol (HTTP) to operate on these resources via the methods GET, PUT, DELETE, and POST. In general, GET is used to access part of a resource, PUT to add information to a resource, DELETE to remove part of a resource, and POST to either update or add to the resource. PUT and POST can be somewhat confusing, as any message to the server that requires a multi-part form data (e.g. attachments) must use a POST message. In Tethys, POST is used to add documents to the database and to query. POST is used for queries as the XQuery is passed as field "XQuery" in a multipart form.

XQuery Resource

POST – Run database query

POST is the only method supported for resource XQuery. A body attachment named XQuery is expected with the post operation and should contain a valid XQuery. See the Tethys manual for information on XQuery and our extensions to support external collections (e.g. physical data such as sea surface temperature). The result is returned in XML.

Optional form variables:

- plan – When plan is present and have a value > 0, the query is not run, but instead the database manager will return an XML document describing how the query will be executed. This allows one to examine details such as whether or not indices are used. See the chapter in the *Oracle Berkeley db XML Getting started Guide* on verifying indices using query plans for details.
  CAVEATS: Does not currently support external XQuery queries using external collections.
- dataType – Type of data to return. Defaults to XML. *Jeff, please add documentation for plot and save.*

Collection Resources

GET – Retrieval of documents

There are two ways in which resources can be retrieved, via the XQuery resource (described in its own section) or via the GET operator on the resource.

GET on a resource URL (e.g. http://localhost:9779/Detections) has the following behaviors:

1. No arguments – Returns the number of XML documents in the collection

2. Path argument – Interprets the path argument as an XPath query. The namespace tethys.sdsu.edu is bound to the abbreviation ty. Example query:
   `http://localhost:9779//Detections/ty:Detections[DataSource%2FProj`

```
ect = "SOCAL" and DataSource%2FSite = "M" and
DataSource%2FDeployment= 34]
```
Note that standard URL escapes such as %2F for / must be used for /s and other characters that are for XPath instead of part of the URL.

3. Document Identifier parameter – Document identifiers are derived from the basename of documents that are submitted to Tethys. This is specified via ?DocId:
`http://localhost:9779//Detections/?DocId= SOCAL45H_HF_Gg_Lo_jst`
where SOCAL45H_HF_Gg_Lo_jst was derived from detection document SOCAL45H_HF_Gg_Lo_jst.xls that was submitted to the database.

POST – Adding/changing documents

New documents can be added with the POST operator. POST supports several methods for adding new documents that are specified as part of the path: add, ODBC, and rebuild. It is assumed that the user is familiar with importing documents in Tethys, details can be found in the Tethys manual.

POST add

The add method can be used to add Microsoft Office Excel spreadsheets, XML documents, and comma separated value documents. It supports the following variables and multipart body components:

Variables:

- overwrite – Overwrite an existing document: 0 not allowed (default if absent), 1 allowed
- import_map – Specifies the import translation map to be used. This maps from user field names to Tethys names. For Excel spreadsheets only, this can also be specified by adding the text Parser to the first row of any column on the MetaData sheet and writing the import map name in row two of the same column.
- species_map – Specifies a translation map between local abbreviations and Latin names. As with import_map, Excel files can embed this in a column called SpeciesAbbreviation on the MetaData sheet with the map name underneath it. Collections that do not report species or that use taxonomic serial numbers directly need not worry about this variable.

Attachments

- data – The document file that is to be added. Mimetype must be specified.
- Attachment – File that is to be associated with this document, the mimetype should be specified. Each attachment is expected to be referenced by name within the data. Currently, attachments are only available for detections, and are typically used to show an image (e.g. spectrogram) or a short clip of audio data. Attachment may be repeated. If many files are to be added, they can be placed in a zip archive. Image files should be in subfolder image and audio files in subfolder audio.

Resources that expect attachments will verify submissions and report any missing attachments. This can be used as a strategy to avoid having to parse the document being submitted on the client side. In our document submission client, we submit a document, check to see if missing attachments are the only problem, and if they are prepare a new submission with the needed files.

This does require the document to be transmitted twice, but saves the additional burden of writing parsing tools for anything but the XML that is returned from the server.

POST ODBC

Open database connectivity (ODBC) is designed to allow data interchange between multiple formats and ODBC libraries let Tethys import data from a wide variety of sources. See the data import section of Tethys for details.

The ODBC method is similar to the add method. Data added by ODBC can either take the form of a file that is transmitted to the Tethys server, or instructions to access a network resource. Both methods share the following variables and body parts:

- import_map – as above
- species_map – as above
- overwrite – as above
- Attachment – as above

ODBC file submission expects a data file attachment as in the add method. The ODBC connection string will be automatically determined for supported types (Excel, XML, CSV, Access). See the Tethys manual for details.

Network submissions expect a ConnectionString parameter that specifies how the data source is to be opened.

When connecting to databases, it is common to import multiple documents at once (e.g. many deployments stored in a separate database). Document names will be automatically generated for each instance. As the number of documents may decrease from one database read to another (e.g. somebody deleted records), it is not recommended to use the overwrite parameter when reading database records. Rather, clear the collection (or at least documents derived from database records within the collection), then import the database.

POST import

Import is a new post interface designed to replace POST add and POST ODBC. It is currently under development and not yet available.

POST expects a series of multipart bodies.

- specification – File containing XML that specifies the how data will be formatted:

```
<?xml version="1.0" encoding="UTF-8"?>
<import>
    <!-- Name that will be used for the associated XML document.
    If more than one document is generated, sequential numbers
    will be assigned, e.g. name1, name2, ...
    If absent, basename of first source will be used, e.g.
    if source is SOCAL32M-HF-SBP.xlsx, use SOCAL32M-HF-SBP.
    -->
    <docname>name</docname>
    <!-- Replace an existing document? -->
    <overwrite>true|false</overwrite>
    <!-- For documents that refer to species, how are species
```

```
            declared:
                TSN - ITIS taxonomic serial number (default if absent)
                Latin - scientific name of taxon
                abbreviation map name - Name of mapping in collection
                    SpeciesAbbreviations
        -->
        <species_map>TSN</species_map>
        <sources>
            <source>
                <!-- What is being sent?
                file - A file in the multipart body attachment.  It is
                assumed that the file type can be derived from the file
                extension and an appropriate ODBC connection string will
                be generated by the server.
                resource - A resource accessible by ODBC on the remote
                server.
                -->
                <type>file|resource</type>
                <!-- The name is the form name -->
                <name>sourcename</name>
                <!-- ODBC connection string.
                Optional for type file.  If specified, remote system
                    will not attempt to guess the connection string.
                Required for type resource.
                -->
                <connectionstring></connectionstring>
            </source>
            <!-- Repeat source as needed.
                Name elements must have unique values -->
        </sources>
    </import>
```

- SourceFiles – One attachment for each file source using the sourcenames. Mimetype must be specified. Example: <name>boobear.xls</name> would expect a boobear.xls bodypart.
- Attachment – File that is to be associated with this document, the mimetype should be specified. Attachments may only be specified when a single document is being added. Each attachment is expected to be referenced by name within the data. Multiple attachments may be handled in one of two ways:
    - Repeat Attachment label. Some clients do not allow this and Attachment1, Attachment2, … is also acceptable.
    - Create a zip archive of file attachments and attach it. Filepaths must match the references.

  Currently, attachments are only available for detection documents, and are typically used to show an image (e.g. spectrogram) or a short clip of audio data. Resources that expect attachments will verify submissions and report any missing attachments. This can be used as a strategy to avoid having to parse the document being submitted on the client side. In our document submission client, we submit a document, check to see if missing attachments are the only problem, and if they are prepare a new submission with the needed files. This requires the document to be transmitted twice, but saves the additional burden of writing parsing tools for anything but the XML that is returned from the server.

Important notes for disaster recovery. When only simple files (comma separated values, spreadsheets, xml) are sent, the source documents will be saved on the server, and the server's

rebuild commands will be able to reconstruct the database if it is lost or corrupted. When more complicated resources are used (e.g. a database), source documents are not stored as multiple snapshots of a database are neither practical nor desirable.

POST rebuild

This method is used to rebuild a method from the source data that was submitted to the Tethys server. This is primarily used for disaster recovery.

Note that this does not include database records. We do not store these as we assume that users backup their databases and can simply re-run a query. Two variables can be passed to this resource method:

- clear: 0 – Do not clear before rebuilding (default), 1 – clear before rebuilding.
- update: 0 – Only update if a document does not exist (default), 1 – Always update.

Collection rebuild jobs can be quite long when there is a large database. As a consequence, a batch log identifier is assigned and returned as the result of the operation. To check the result of a rebuild, access the BatchLogs resource.

DELETE – Deletion of documents

Documents may be deleted by invoking the DELETE method on a collection. A mandatory keyword argument DocID must be used, which has the same format as for the GET method. To remove all documents, the special DocID `<*clear*>` should be used. Note that this is irreversible.

Attach Resource

The attach resource is used for retrieving attachments associated with collections via a GET operation. It uses a path argument to specify the attachment. The first element of the path is the collection resource (e.g. Detections) followed by the document identifier which is usually composed of the basename of the file that was submitted. The attachment is specified with a keyword argument indicating the attachment type and its name. Currently the only valid attachments are Image and Audio.

Example: If image UknownPhenom.jpg was submitted with an Excel spreadsheet ALEUT02BD_MF_MFAOrca_ajc.xls, one would retrieve the image using a GET on URL:

http://localhost:9779//Attach/Detections/ALEUT02BD_MF_MFAOrca_ajc?Image=UnknownPhenom.jpg

BatchLogs Resource

BatchLogs are used by operations that may take longer than a client can be expected to wait. Instead, an identifier is returned to the user and this is used as the path for a BatchLogs GET request.

Example:
http://localhost:9779//BatchLogs/DetectionsRebuildX32J97.log

Tethys Resource

The Tethys resource is used for controlling the Tethys server itself.

.

Get

The following GET operations are allowed on the Tethys resource:

Tethys/ping – Returns an XML document if the server can respond:

```
<Tethys>
    <ping>alive</ping>
</Tethys>
```

Tethys/performance_monitor – Reports status of the query performance monitor (on|off)

```
<Tethys>
    <performance_monitor>on|off</performance_monitor>
</Tethys>
```

Put

Several Put commands are allowed on the Tethys resource:

Tethys/shutdown – Exit the server cleanly.

Tethys/checkpoint – Create a checkpoint in the transaction journaling system. This should be done preferably before backing up or moving the database. Tethys will automatically do this anytime the server is restarted.

Tethys/performance_monitor/ACTION – Set the performance monitor state where ACTION is:

on – Turns performance monitor on if it is not already on.

off – Turns off performance monitor if not already off

clear – Resets the performance monitor counts. If it is off, performance monitor is turned on.

Example using curl URL tool: curl -X get http://localhost:9779//Tethys/ping . Graphical user interfaces are available in tools like insomnia and postman.

References

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, The University of CA, Irvine, Irvine, CA.