Efficiency and Opportunities Gained with a Data Exchange API

GREG WILKE,
PSMFC, STREAMNET PROJECT
API and REST

Application Programming Interface

- I speak HTTP
- Hey!
- Me too!

= REST API

- Speak HTTP
- Basic data format
- Easy for clients to use
- Data both directions
- CRUD functions
  (Create, Read, Update, Delete)

Google API
Weather API
GitHub API
State Open Data Portal
Twitter API
Amazon Cloud API
REST APIs are Everywhere

Private sector:

Google Maps - https://developers.google.com/maps/documentation/javascript/tutorial
Twitter - https://developer.twitter.com/
Netflix - https://www.programmableweb.com/api/netflix
Facebook - https://developers.facebook.com/
Bloomberg - https://www.bloomberg.com/professional/support/api-library/
Weather Underground - https://www.wunderground.com/weather/api/
FullContact - https://www.fullcontact.com/developer/
Amazon S3 (Amazon cloud) - https://aws.amazon.com/s3/
Twilio - https://www.twilio.com/
MailChimp - https://developer.mailchimp.com/documentation/mailchimp/

Federal government:

- Data.gov - https://www.data.gov/developers/apis
- Of the roughly 250 federal agencies, approximately half now have public APIs.
- There are almost 100 developer hubs across the federal government.

State government:

Open Data Portals

Programmable Web - https://www.programmableweb.com/

Apigee.com - API services interface with list of available services

R is the preferred letter in CRUD
https://api.streamnet.org/api/v1/catables

```json
{
    "tables": [
        {
            "name": "SuperPopulations",
            "count": 211,
            "lastmodifiedon": "N/A",
            "description": "Multiple Population group reference",
            "id": "009A08FE-6479-44FC-9B6F-01C55E2C88A3",
            "lastmodifiedby": "N/A",
            "primary_key": "SuperPopID,PopID",
            "type": "List"
        },
        {
            "name": "Reference",
            "count": 19533,
            "lastmodifiedon": "N/A",
            "description": "",
            "id": "1FB86FDA-2DC0-4FCD-9B7F-E37C0C57114F",
            "lastmodifiedby": "N/A",
            "primary_key": "RefID",
            "type": "DES"
        },
        {
            "name": "PNI",
            "count": 113,
            "lastmodifiedon": "N/A",
            "description": "Proportionate Natural Influence of supplementation hatcheries",
            "id": "35F33AE9-75A1-473B-9236-C9AA17083826",
            "lastmodifiedby": "N/A",
            "primary_key": "ID",
            "type": "DES"
        }
    ]
}
```
Basic REST API Functional Components

**Client Side**
- Client System (App)
  - Written in any computer language able to negotiate HTTP calls
  - .NET
  - PHP
  - MS Access
  - PERL
  - Coldfusion, Android & iPhone

**Server Side**
- Common HTTP Request transport to send and retrieve data
- Common text-based data format: JSON or XML
- Common simple CRUD Functions: Create, Read, Update, Delete

**Common Web Server**
- IIS, NGINX, Apache

**Server App**
- Written in any computer language able to negotiate HTTP calls
- .NET
- PHP
- Coldfusion
- Java
StreamNet API Design

Design goals

- Minimal documentation necessary
- Server-side data abstraction
- Standard REST convention of collections & members
- Robust data validation

Kickoff

Let the client development begin!
StreamNet API Data Flow

1. Loop over data table, for each record send SQL Database Client App

2. Client App then processes response (i.e. log result)

**HTTP POST Request (Insert)**

Data record in JSON format

```json
{"table_id":"3613A357-258C-4615-927F-EB368C61E13E", "record_values": {
  "rperstype":"Adult recruits per adult spawners", "recruitssalpha":"
, "age4adults":
, "methodadjustments":
}
```

**HTTP Failure Return Code**

Error Data in JSON Format

```json
{"error": {"msg":"data could not be processed", "record_values": {
  "Age1Juvs": "required for DES, but does not exist in the incoming record",
  "HarvestAdj\length=9": "the length of this string value must be <= 3",
  "Age2Juvs\value=seven": "this should be a decimal value"}, "params":{}}}
```

**HTTP Success Return Code**

Success Data in JSON Format

```json
{"record": {"id":"8a36b236-3a4f-4352-bae1-16dbbb3bd5b4"}}
```

StreamNet REST API

Server App (Data Validation)

Validation Only

Post Record

SQL Database
Data Validation

- Field level
- Record level
- Table level

```
"error": {
  "msg": "data could not be processed",
  "record_values": {
    "primary_key": "The fields (TrendID) must be unique, record id: AE7EFF1-9B32-420B-9787-0A20A3EEE91F has the same values as the submitted record.",
    "ID-PE7EFF1-9B32-420B-9787-0A20A3EEE91F": "this id already exists in table: TrendEdits",
    "publish": "Must be \"Yes\" or \"No\".",
  },
  "params": {}
}
```
Efficiencies & Opportunities Gained

- Faster Data Validation
- Better Quality Data
- Easy to Share Data with Others
- Integration with Other APIs
Current Status

- High Level Indicator data all submitted through the API for roughly 2 years now.

- Primary time-series data started being submitted through the API as of a few months ago. I said earlier that this type is more complicated with normalized data, but we implemented a system that works for our project.

- Some data with GIS information embedded do not go live immediately because they must still be processed and validated by a person using GIS tools and StreamNet’s maps and layers. We have no plans to automate this at present.

- Our long-toiling Data Manager was finally able to retire this year at the age of 67
Publishing Data with a REST API

Most REST APIs are used for PUBLISHING data. That is, the API is made available over the internet for clients to consume the data. This is the context you mostly hear about. State open data initiatives are about making data available to the public or specific users. Companies have APIs that allow custom software to have authorized access to data.

But, a REST API can also be developed to receive data.

The StreamNet REST API publishes fisheries data compiled by state and tribal partners. The API is also the mechanism for partners to submit and validate data.

https://api.streamnet.org

A significant efficiency was gained by establishing a mechanism whereby compilers validate their own data using the API. Compilers can submit their data to the API as much as they want with the “validate=true” parameter with no changes to the regional database. We also use a Boolean field called Publish that defaults to “No” on every record. This allows data to be submitted and the database updated, but no record with Publish=No will show in any Web query tools or public/anonymous API calls.

For the purpose of getting data updated in a timely manner, this method actually works ok because the bulk of time involved with getting fisheries data from the stream to the regional database is in the field and in Fish & Wildlife agency regional offices where data is compiled and checked, and indicators calculated. The “last-mile” is getting the data submitted to a regional database like StreamNet.
Basic REST API Data Formats

**JSON**

```json
{
    "record": {
        "table_id": "def09e8-2a8-4c98-8083-a272c2c3e3",
        "record_values": {
            "AgeProp": "",
            "PepID": "1024",
            "TS41": "",
            "Age10PropUpperLimit": "",
            "NullRecord": false,
            "Age10PropLowerLimit": "",
            "MetricLocation": "",
            "ContactPhone": "208-465-8404",
            "Age10Prop": "",
            "AgePropAlpha": "",
            "pH551LnewLimit": "",
            "Age10PropLnewLimit": "",
            "ContactAgency": "Idaho Department of Fish and Game",
            "Age10PropLowerLimit": "",
            "Age10PropUpperLimit": "",
            "TS41LnewLimit": "",
            "SpawningYear": "2017",
            "Age10PropLnewLimit": "",
            "pH551LnewLimit": "1139",
            "Age10PropLowerLimit": "",
            "Comments": "",
            "ContactPersonFirst": "Carlos",
            "Age10PropUpperLimit": "",
            "pH551LnewLimit": "",
            "PopSet": "Multiple",
            "Age10PropLnewLimit": "",
            "pH551LnewLimit": "",
            "RefID": "46994",
            "Waterbody": "Lower and Upper Middle Fork Salmon River",
        }
    }
}
```

**XML**

```xml
<record>
    <table_id>def09e8-2a8-4c98-8083-a272c2c3e3</table_id>
    <record_values>
        <ID/>
        <TrendID/>
        <BeginDate/>
        <EndDate/>
        <SampleDate/>
        <SampleID/>
        <Age10Prop/>
        <AgeProp/>
        <AgePropAlpha/>
        <pH551LnewLimit/>
        <Age10PropLnewLimit/>
        <ContactAgency/>
        <Age10PropLowerLimit/>
        <Age10PropUpperLimit/>
        <TS41LnewLimit/>
        <SpawningYear/>
        <Age10PropLnewLimit/>
        <pH551LnewLimit/>
        <Comments/>
        <ContactPersonFirst/>
        <Age10PropUpperLimit/>
        <pH551LnewLimit/>
        <PopSet/>
        <Age10PropLnewLimit/>
        <pH551LnewLimit/>
        <RefID/>
        <Waterbody/>
    </record_values>
</record>
```
StreamNet REST API Documentation

The StreamNet REST API allows for simple access to data sets using HTTP calls and exchanges data in JSON or XML format.

The base URI for the REST endpoint:
https://api.streamnet.org

Here is an example usage:
https://api.streamnet.org/api/v1/users

Coordinated Assessments

Create, access, update, delete data records collected for the Coordinated Assessments data sets.

<table>
<thead>
<tr>
<th>TABLES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /api/v1/tables</td>
<td>Fetch the list of Coordinated Assessments data tables.</td>
</tr>
<tr>
<td>GET /api/v1/tables/id</td>
<td>Fetch the metadata for a single table (field names &amp; types). This is the table schema.</td>
</tr>
</tbody>
</table>
REST and HTTP

Now, it doesn't have to be the case, but usually a REST API works pretty much the same way a website does.

When a client browser like Chrome or Firefox makes a call from your client computer system to a server, you get data back over the HTTP protocol. The HTTP protocol is one of the primary mechanisms or specifications for internet communication. In the case of the browser, the server delivers data which the browser formats into a web page.

In the case of a REST API, a software client requests and receives data from a server (or sends data to the server) using the same underlying HTTP protocol as the browser, but what the client software does with the response depends on the application.

In the case of a browser, it displays information on a web page, but in the case of a GIS client application, a REST API could deliver a GIS layer that the client displays as a polygon on a map. Or an API could deliver tabular data to be imported into a SQL table or shown in a grid for the user.
REST Data Formats

JSON

XML