Streamflow Prediction Tool Documentation

Release 1.0.5

Alan D. Snow

Sep 01, 2017

Contents

The Streamflow Prediction Tool provides 15-day streamflow predicted estimates by using the European Center for Medium Range Weather Forecasts (ecmwf.int) runoff predictions routed with the RAPID (rapid-hub.org) program. The connection between the predicted and hindcasted runoff are generated with GIS tools both from Esri as well as from open source contributions. Return period estimates and warning flags aid in determining the severity.

The Streamflow Prediction Tool was the result of a collaboration between:

- Brigham Young University
- Esri
- The European Centre for Medium Range Weather Forecasts
- NASA's Jet Propulsion Laboratory

The Streamflow Prediction Tool was developed at Brigham Young University with support from the National Science Foundation (NSF) under Grant No. 1135483. The Streamflow Prediction Tool is part of a larger effort known as CI-Water (http://ci-water.org). The purpose of CI-Water is to develop cyber infrastructure for water resources decision support.

Currently, the Streamflow Prediction Tool is being developed and maintained by the U.S. Army Engineer Research and Development Center.

CHAPTER 1

Publications

Snow, Alan D., Scott D. Christensen, Nathan R. Swain, E. James Nelson, Daniel P. Ames, Norman L. Jones, Deng Ding, Nawajish S. Noman, Cedric H. David, Florian Pappenberger, and Ervin Zsoter, 2016. A High-Resolution National-Scale Hydrologic Forecast System from a Global Ensemble Land Surface Model. *Journal of the American Water Resources Association (JAWRA)* 1-15, DOI: 10.1111/1752-1688.12434. http://onlinelibrary.wiley.com/doi/10.1111/1752-1688.12434/abstract

Snow, Alan Dee, "A New Global Forecasting Model to Produce High-Resolution Stream Forecasts" (2015). All Theses and Dissertations. Paper 5272. http://scholarsarchive.byu.edu/etd/5272

CHAPTER 2

Contents

Setup: Stream Network Generation

ArcHydro Tools

1. RAPID Tools (Esri Version): https://github.com/Esri/python-toolbox-for-rapid
2. RAPID Tools (ERDC Version): https://github.com/erdc-cm/python-toolbox-for-rapid

TauDEM & RAPIDpy Tools

1. RAPIDpy: http://rapidpy.readthedocs.io/en/latest/gis_stream_network.html

Setup: ECMWF-RAPID Input Generation

Tip: If you have not already generated your stream network, see these instructions *Setup: Stream Network Generation*.

GitHub Repo List

1. RAPID Tools (Esri Version): https://github.com/Esri/python-toolbox-for-rapid
2. RAPID Tools (ERDC Version): https://github.com/erdc-cm/python-toolbox-for-rapid
3. RAPIDpy: http://rapidpy.readthedocs.io/en/latest/gis_tools.html

Setup: Historical Land Surface Model Process

RAPIDpy

1. RAPIDpy: http://rapidpy.readthedocs.io/en/latest/lsm_inflow.html

Setup: Forecast Framework

GitHub Repo

The github repo for the forecast framework is located at:

https://github.com/erdc-cm/spt_ecmwf_autorapid_process

Setup: Web Application

tethysapp-streamflow_prediction_tool

This app requires you to have the ECMWF AutoRAPID preprocessing completed separately. See:

- *Setup: Stream Network Generation*
- *Setup: ECMWF-RAPID Input Generation*
- *Setup: Historical Land Surface Model Process*
- *Setup: Forecast Framework*

Publications:

Snow, Alan D., Scott D. Christensen, Nathan R. Swain, E. James Nelson, Daniel P. Ames, Norman L. Jones, Deng Ding, Nawajish S. Noman, Cedric H. David, Florian Pappenberger, and Ervin Zsoter, 2016. A High-Resolution National-Scale Hydrologic Forecast System from a Global Ensemble Land Surface Model. *Journal of the American Water Resources Association (JAWRA)* 1-15, DOI: 10.1111/1752-1688.12434. http://onlinelibrary.wiley.com/doi/10.1111/1752-1688.12434/abstract

Snow, Alan Dee, "A New Global Forecasting Model to Produce High-Resolution Stream Forecasts" (2015). All Theses and Dissertations. Paper 5272. http://scholarsarchive.byu.edu/etd/5272

Prerequisites:

- Tethys Platform 2.0 (CKAN, PostgresQL, GeoServer): See: http://docs.tethysplatform.org
- RAPIDpy (Python package).
- Geoserver needs CORS enabled.

Note: Before installing the Streamflow Prediction Tool, RAPIDpy, or the spt_dataset_manager, activate your Tethys Platform python environment:

```
$ t
```

Install RAPIDpy:

For instructions, go to: https://github.com/erdc-cm/RAPIDpy.

Install spt_dataset_manager:

For instructions, go to: https://github.com/erdc-cm/spt_dataset_manager.

Installation:

Clone the app into the directory you want:

```
$ git clone https://github.com/erdc-cm/tethysapp-streamflow_prediction_tool.git
$ cd tethysapp-streamflow_prediction_tool
```

Then install the app in Tethys Platform.

Source Code Setup:

A. App Development:

```
$ t
(tethys) $ cd tethysapp-streamflow_prediction_tool
(tethys) $ python setup.py develop
```

B. Production:

See: http://docs.tethysplatform.org/en/stable/installation/production/app_installation.html

```
$ t
(tethys) $ cd tethysapp-streamflow_prediction_tool
(tethys) $ python setup.py install
(tethys) $ tethys syncstores streamflow_prediction_tool
(tethys) $ tethys manage collectall
(tethys) $ tethys_server_own
```

Restart the server. See: http://docs.tethysplatform.org/en/stable/installation/production/app_installation.html#restart-uwsgi-and-nginx

Configure App Settings:

Go to _"http://localhost:8000/admin/tethys_apps/tethysapp/" and select 'Streamflow Prediciton Tool'. Update required settings.

Setup the Database:

```
$ t
(tethys) $ tethys syncstores streamflow_prediction_tool
```

Updating the App:

Update the local repository and Tethys Platform instance.

```
$ t
(tethys) $ cd tethysapp-streamflow_prediction_tool
(tethys) $ git pull
(tethys) $ tethys_server_own
```

Reset the database if changes are made to the database (this will delete your old database):

```
$ tethys syncstores streamflow_prediction_tool -r
```

The last step is to restart the server. See: http://docs.tethysplatform.org/en/stable/installation/production/app_installation.html#restart-uwsgi-and-nginx

Crontab Errors

Check if your server has crontab permissions: Ex:

```
# su -s /bin/bash apache
bash-4.2$ crontab -e
You (apache) are not allowed to use this program (crontab)
See crontab(1) for more information
```

If not, add the permissions in the cron.allow file.

```
# echo apache >>/etc/cron.allow
```

SELinux

If you are using a drive/folder not associated with your normal apache server locations, you may need to set SELinux to allow it. In this example, I am using a folder named /tethys

```
# semanage fcontext -a -t httpd_sys_content_t '/tethys(/.*)?'
# restorecon -Rv /tethys
```

SPT REST API

A REST API is a web service or a set of methods that can be used to produce or access data without a web interface. REST APIs use the http protocol to request data. Parameters are passed through a URL using a predetermined organization. A REST API has been developed to provide access to the Streamflow Prediction Tool (SPT) forecasts without the need to access the web app interface. This type of service facilitates integration of the SPT with third party web apps, and the automation of forecast retrievals using programing languages like Python, or R. The available methods and a description of how to use them are shown below.

GetForecast for Forecasts Statistics

Parameter	Description	Example
watershed_name	The name of watershed or main area of interest.	Nepal
subbasin_name	The name of the sub basin or sub area.	Central
reach_id	The identifier for the stream reach.	5
forecast_folder	The date of the forecast (YYYYMMDD.HHHH)*[0]. (Optional)	20170110.1200
stat_type	The selected forecast statistic. (high_res, mean, std_dev_range_upper, std_dev_range_lower, max, min).	mean
units	Set to 'english' to get ft3/s. (Optional)	english
return_format	Set to 'csv' to get csv file. (Optional)	csv

*

Example

```
>>> import requests
>>> request_params = dict(watershed_name='Nepal', subbasin_name='Central', reach_id=5,
↪ forecast_folder='most_recent', stat_type='mean')
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/GetForecast/
↪', params=request_params, headers=request_headers)
```

GetHistoricData (1980 - Present)

Parameter	Description	Example
watershed_name	The name of watershed or main area of interest.	Nepal
subbasin_name	The name of the sub basin or sub area.	Central
reach_id	The identifier for the stream reach.	5
units	Set to 'english' to get ft3/s. (Optional)	english
return_format	Set to 'csv' to get csv file. (Optional)	csv

Example

```
>>> import requests
>>> request_params = dict(watershed_name='Nepal', subbasin_name='Central', reach_id=5)
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/
↪GetHistoricData/', params=request_params, headers=request_headers)
```

[0] forecast_folder=most_recent will retrieve the most recent date available.

GetReturnPeriods (2, 10, and 20 year return with historical max)

Parameter	Description	Example
watershed_name	The name of watershed or main area of interest.	Nepal
subbasin_name	The name of the sub basin or sub area.	Central
reach_id	The identifier for the stream reach.	5
units	Set to 'english' to get ft3/s. (Optional)	english

Example

```
>>> import requests
>>> request_params = dict(watershed_name='Nepal', subbasin_name='Central', return_
↪period=2)
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/
↪GetReturnPeriods/', params=request_params, headers=request_headers)
```

GetAvailableDates

Parameter	Description	Example
watershed_name	The name of watershed or main area of interest.	Nepal
subbasin_name	The name of the sub basin or sub area.	Central
reach_id	The identifier for the stream reach.	5

Example

```
>>> import requests
>>> request_params = dict(watershed_name='Nepal', subbasin_name='Central', reach_id=5)
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/
↪GetAvailableDates/', params=request_params, headers=request_headers)
```

GetWatersheds

This method takes no parameters and returns a list of the available watersheds.

Example

```
>>> import requests
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/
↪GetWatersheds/', headers=request_headers)
```

GetWarningPoints

Parameter	Description	Example
watershed_name	The name of watershed or main area of interest.	Nepal
subbasin_name	The name of the sub basin or sub area.	Central
return_period	The return period that the warning is based on.	(2,10, or 20)
forecast_folder	The date of the forecast (YYYYMMDD.HHHH). (Optional†[0])	20170110.1200

†

Example

```
>>> import requests
>>> request_params = dict(watershed_name='Nepal', subbasin_name='Central', return_
↪period=20, forecast_folder='20170802.0')
>>> request_headers = dict(Authorization='Token asdfqwer1234')
>>> res = requests.get('[HOST Portal]/apps/streamflow-prediction-tool/api/
↪GetWarningPoints/', params=request_params, headers=request_headers)
```

[0] If you don't include forecast_folder, it will retrieve the most recent date available.