

# Geostreaming API

```
# -----
# GEOSTREAMS ENDPOINT
# -----
POST      /api/geostreams/datapoints      api.Geostreams.
addDatapoint
DELETE     /api/geostreams/datapoints/:id  api.Geostreams.
deleteDatapoint(id: String)
GET        /api/geostreams/datapoints      api.Geostreams.
searchDatapoints(operator = "", since: Option[String] ?= None, until: Option[String] ?= None, geocode: Option
[String] ?= None, stream_id: Option[String] ?= None, sensor_id: Option[String] ?= None, sources: List[String] ?
= List.empty, attributes: List[String] ?= List.empty, format: String ?= "json", semi: Option[String])
GET        /api/geostreams/datapoints/averages  api.Geostreams.
searchDatapoints(operator = "averages", since: Option[String] ?= None, until: Option[String] ?= None, geocode:
Option[String] ?= None, stream_id: Option[String] ?= None, sensor_id: Option[String] ?= None, sources: List
[String] ?= List.empty, attributes: List[String] ?= List.empty, format: String ?= "json", semi: Option[String])
GET        /api/geostreams/datapoints/trends    api.Geostreams.
searchDatapoints(operator = "trends", since: Option[String] ?= None, until: Option[String] ?= None, geocode:
Option[String] ?= None, stream_id: Option[String] ?= None, sensor_id: Option[String] ?= None, sources: List
[String] ?= List.empty, attributes: List[String] ?= List.empty, format: String ?= "json", semi: Option[String])
GET        /api/geostreams/datapoints/bin/:time/:depth  api.Geostreams.
binDatapoints(time, depth: Double, raw: Boolean ?= false, since: Option[String] ?= None, until: Option[String] ?
= None, geocode: Option[String] ?= None, stream_id: Option[String] ?= None, sensor_id: Option[String] ?= None,
sources: List[String] ?= List.empty, attributes: List[String] ?= List.empty)
GET        /api/geostreams/datapoints/:id      api.Geostreams.
getDatapoint(id: String)
GET        /api/geostreams/cache              api.Geostreams.
cacheListAction
GET        /api/geostreams/cache/invalidate    api.Geostreams.
cacheInvalidateAction(sensor_id: Option[String] ?= None, stream_id: Option[String] ?= None)
GET        /api/geostreams/cache/:id          api.Geostreams.
cacheFetchAction(id)
POST       /api/geostreams/sensors            api.Geostreams.
createSensor
GET        /api/geostreams/sensors/update      api.Geostreams.
updateStatisticsStreamSensor()
GET        /api/geostreams/sensors/:id        api.Geostreams.getSensor
(id: String)
PUT        /api/geostreams/sensors/:id        api.Geostreams.
updateSensorMetadata(id: String)
GET        /api/geostreams/sensors/:id/stats  api.Geostreams.
getSensorStatistics(id: String)
GET        /api/geostreams/sensors/:id/streams  api.Geostreams.
getSensorStreams(id: String)
GET        /api/geostreams/sensors/:id/update  api.Geostreams.
updateStatisticsSensor(id: String)
GET        /api/geostreams/sensors            api.Geostreams.
searchSensors(geocode: Option[String] ?= None, sensor_name: Option[String] ?= None)
DELETE     /api/geostreams/sensors/:id        api.Geostreams.
deleteSensor(id: String)
POST       /api/geostreams/streams            api.Geostreams.
createStream
GET        /api/geostreams/streams/update      api.Geostreams.
updateStatisticsStreamSensor()
GET        /api/geostreams/streams/:id        api.Geostreams.getStream
(id: String)
PUT        /api/geostreams/streams/:id        api.Geostreams.
patchStreamMetadata(id: String)
GET        /api/geostreams/streams/:id/update  api.Geostreams.
updateStatisticsStream(id: String)
GET        /api/geostreams/streams            api.Geostreams.
searchStreams(geocode: Option[String] ?= None, stream_name: Option[String] ?= None)
DELETE     /api/geostreams/streams/:id        api.Geostreams.
deleteStream(id: String)
DELETE     /api/geostreams/dropall            api.Geostreams.deleteAll
GET        /api/geostreams/counts            api.Geostreams.counts
GET        /api/geostreams/config            api.Geostreams.getConfig
```

## Suggested Endpoints

GET /api/geostreams/parameters - Looking for a list of unique keys in the properties of all included datapoints. One thought on this one is that if we make this a proper table in the database, we would be able to store the original parameter name as found from the source, then map it dynamically in this table.

GET /api/geostreams/sources - The "path" to this data is currently stored in SENSOR -> PROPERTIES -> TYPE. Type is an object that includes "id" and "title", where "id" is the acronym for the source and title is the "pretty name".

GET /api/geostreams/datapoints/counts - some endpoint so that I can get the count of datapoints quickly without downloading all of the datapoints. This would allow us to do things like, "There are 84,000 datapoints in your request, are you sure you want to try to draw a graph?"

GET /api/geostreams/areas - right now we define this in the config, but maybe we want to consider storing these here as well

GET /api/geostreams/layers - also stored in the config, but typically stored from the Geoserver, so maybe we could have a way to integrate the two.

GET /api/geostreams/regions - this is stored on the sensor now in properties. It is pretty vague, and we may want to consider whether or not we need this or could refactor it.