**CEOS Database API Overview**

*Rational, Use Cases, and Implementation Approach*

*Last Updated 7 March 2017*

**Rational**

Enabling machine-to-machine access to CEOS Database content via an API (Application Programming Interface) would enable the content to be embedded in and leveraged by external developers. Several potential API users have been identified, including WGISS and the CEOS SEO – however, it could be expected that other users may develop applications on top of the API once it’s is published, and in addition, we may find internal uses for the API that we have not yet considered.

**Initial Functionality**

The following initial API functions are proposed.

1. API call by missionID (or name with controlled vocab) to get mission metadata (e.g. launch date, end of life date, status, etc. from a profile [like this](http://database.eohandbook.com/database/missionsummary.aspx?missionID=655))
2. API call by instrumentID (or name with controlled vocab) to get instrument metadata (e.g. type, resolution, measurements, etc. from a profile [like this](http://database.eohandbook.com/database/instrumentsummary.aspx?instrumentID=1558))
3. [*Phase 2, USGS request*] API call by wavebandID to get instruments with certain waveband properties. (Need to be confirmed with SEO and USGS.)
4. API call by measurementTypeID to get a list of instruments making a particular measurements (e.g. [this list](http://database.eohandbook.com/measurements/measurements.aspx?measurementTypeID=9))
5. API call by gcosipECVID (ECV = Essential Climate Variable) to get a list of instruments measuring a particular ECV, and/or metadata about the ECV (e.g. action item, action status, products from a profile [like this](http://database.eohandbook.com/climate/gcosecv.aspx?gcosipECVID=10)).
6. API call by agencyID (or name with controlled vocab) to get agency mission or instrument metadata.

It is expected that these would be modified, and others added, based on user feedback.

**Implementation Approach**

It is suggested that an incremental development approach is taken, leveraging to the greatest extent possible existing and established services and frameworks.

An initial assessment of the AWS (Amazon Web Services) API Gateway calling Lambda functions to access the database for our expected level of service need appears promising. The cost scales with usage, and there appears to be a significant free tier which may end up covering most of our requirements. The development process is comparatively straight forward, and allows the developer to focus directly on writing the code (Python, or C# supported, CEOS database website implemented in C#) that will power the response. Capacity with AWS is not an issue even for the largest operators, and our application will be very modest in comparison, so scaling is not an issue. Going with the AWS does lead to some platform lock-in, but does enable rapid prototyping and spin-up with minimal development effort.