The Ocean Observatories Data Evaluation Team (the Data Team) is part of the Cyberinfrastructure group at Rutgers University. They are tasked with reviewing the oceanographic and engineering data from over 1,200 instruments deployed throughout the OOI system, ensuring that the data and metadata delivered by the OOI meets community data quality standards. They also work with the user community and marine engineers to identify, diagnose and resolve data availability and data quality issues. The Data Team is also responsible for user outreach and training regarding data access, availability, processing routines, and quality control.

1. Mission
The OOI Data Team works to ensure that the quality of data delivered by the OOI system meets the scientific needs of the oceanographic community.

2. Team Goals
The Data Team’s primary goals are:
   - To monitor the operational status of data flowing through the OOI Data system (from raw, to ingestion, uFrame, ERDDAP, and plotting)
   - To ensure the availability of OOI datasets in the system (raw, processed, derived, and cruise)
   - To ensure that data delivered by the system meets quality guidelines
   - To identify data availability and quality issues and ensure they are resolved
   - To communicate known data issues with end users
   - To report operational statistics on data availability and quality, and issue resolution

3. Key Metrics
   - How much data is available through the system?
   - How much of the data has been reviewed? How much remains?
   - How much of the data meets quality control standards?

4. Key Activities
To meet the above goals, the Data Team focuses on the following key activities.

Daily Activities
   - Review the end-to-end operational status of online instruments and investigate any outages (e.g. instrument, telemetry, parsing, or ingestion failures).
   - Review the operational status of other data archives (raw, cruise, ERDDAP)
• Look into and resolve new system alerts.
• Follow up on any issue requests from users (via Redmine)
• Add annotations to notify users of operational status changes.

Periodic Activities
• Meet with Marine IOs to discuss operational issues and data quality questions.
• Review instruments, streams, parameters and deployments provided by the system for completion.
• Conduct deep dives on datasets to review availability and quality.
• Review data from completed deployments to assess data quality. Add annotations as appropriate.
• Develop new scripts, plotting tools, and quality checks to improve quality control processes for streaming, telemetered and recovered datasets.
• Produce reports on the availability and quality of datasets.
• Review appropriateness of QC flags.
• Follow up with Marine after each cruise to ensure asset, deployment, calibration and ingestion configurations have been updated, and cruise reports have been posted.
• Prototype and test new data portal user interface and visualization features.

5. Key Questions and Workflows
The following section outlines the key data evaluation questions the team focuses on. For each question, a standard operating procedure (workflow) is provided. This may be in the form of a checklist or a process/decision tree. In many cases the results are also summarized in one or more reports the team will produce, example templates of which are also included.

5.1 Daily Reviews
Each day the Data Team will review the overall operational status of the system, and investigate any changes in status or new issues reported by users or other OOI staff. These activities are generally triggered by alerts from the system or others.

Questions
• Have any telemetered or streaming data streams stopped updating? If so, why?
  o Are there issues with the instrument?
  o Are there Issues with telemetry or raw data transfer?
  o Are there issues with ingestion?
• Is recent data still reasonable (scientifically valid)?
  o Are there any issues with the dataset (QC flags or fill values)?
  o Have any potential issues been added to Redmine for investigation?
  o Have all identified issues been annotated in the system?
  o Has the sampling rate changed?
• Are there new issues from users (via Help Desk)?
• Has the status of any operational (e.g. Redmine) issues changed?
Workflows
The first two questions above are addressed in Figure 1, which describes the review process (decision tree) the team follows when an instrument or parameter alert is raised.

![Figure 1. Daily review process diagram (prototype).](image)

In addition to reviewing recent alerts from the system, each day the team also reviews Redmine for any new issues from users or updates on previous issues (which may include operational updates from MIOs, troubleshooting of instrument issues, or CI software fixes). In general, the approach to addressing these is very issue specific. But each day the Data Team will investigate issues that have been assigned to them, and then reassign or close them as appropriate.

5.2 Periodic Reviews (weekly/monthly/quarterly)
The following activities are generally triggered on a regular schedule. For example, while scientific reviews of instruments may occur every day, only a small subset is investigated on any given day. These reviews generally lead to operational reports on the status of the system and the activities of the data team.

Questions/Metrics
- What is the status of all telemetering data streams?
  - Majority of status should stem from automated system (TBD), which is used to help populate status timeline
  - TBD: example report summary and wireframe, frequency of delivery, and level of detail
- Have the MIOs made any operational changes?
Need upgrade to operations log and regular MIO communications

- Has the status of any flagged instrument changed?
- Are all streams and parameters available in the system?
- How complete and reasonable are all data streams?
- Have all recovered data been processed and reviewed?

Workflows
Over the past year, building on the results of the first operational end-to-end quality audit that began in 2016, the data team constructed a database of all data products, data streams, instruments, nodes, and platforms from every array (http://ooi.visualocean.net/). The OOI machine-to-machine (M2M) interface is used to download data, which are then run through several automated QC routines (https://github.com/ooi-data-review/), and then annotations are added to every data product. The goal is to produce an availability and quality timeline for every data product that every user can access.

Rest in Class Review Procedure:

1. **STEP 1:** M2M data request
   a. Data request and access can be done via a terminal.
      i. Authenticate by creating a .netrc file in your home directory set up with the following:
         1. machine ooinet.oceanobservatories.org
         2. login your_login
         3. password your_token
         4. Note: login and password are found on https://ooinet.oceanobservatories.org under User Profile
      ii. Clone M2M toolbox via a terminal as follows:
          1. $ cd your_directory
          2. $ git clone https://github.com/kerfoot/uframe-m2m.git
      iii. Build a request as shown in the example below:
          1. $ python build_instrument_requests.py -h
             a. (Note: -h to see options available to use in the request command line e.g. -s to specify start time and -e to specify end time)
          2. $ python build_instrument_requests.py -b https://ooinet.oceanobservatories.org -u leila-ocean-gmail --email leila.ocean@gmail.com --csv -r CE01ISSM-MFD35 > CE01ISSM-MFD35.csv
      iv. Send a request as follows:
          1. $ python send_requests.py -f CE01ISSM-MFD35.csv
   v. Access your request from email:
      1. Check emails from services@oceanobservatories.net with links to data files.
2. In this example, data files from instruments on CE01ISSM-MFD35 are accessed here.

   b. Data request and access can be done via a web browser.
      i. Open a web browser and modify the buildup example request given here to request different data:

2. **STEP 2:** Automated Data Evaluation Tests. Use check_data.py to iterate over the data files and execute the following tests:
   a. Instrument Level Tests
      i. Operational status test: fit the instrument operational status from GitHub ingestion-csvs folder to the deployment time range
      ii. Actions:
         1. Use instrument status list to enter annotations in the database
         2. Roll down instrument status to stream and parameter level tests
   b. Stream Level Tests
      i. Deployment Start and End Date Test: test the start and end dates against deployment information in the system asset management
      ii. Deployment Lat/Lon Test (distance less than .5 km): test the data file Latitude and Longitude against deployment information in the system asset management
      iii. Timestamps Duplicate Test: test if the data file timestamp series has no duplicates; all dates are unique.
      iv. Parameter Availability Test: test if the stream contains all expected parameters listed in QC database (based on IDD/IOS, DPS documentation, and SME input)
      v. Actions:
         1. Use the stream status list to enter annotations in the database
         2. Roll down the stream status to the parameters level tests
      vi. NOT IN AUTOMATED ROUTINE:
         1. Depth/Pressure test (distance greater than 1 m): test data file pressure record against the deployment information in asset management.
         2. Timestamps order sequence test: Pass if data are in ascending chronological order, investigate if there are jumps or out of order time stamps
   c. Parameter Level Tests
      i. Global range test: check data min and max values against the system global_range lookup values
      ii. Fill value test: check if fill values exist in the time-series
      iii. Fill values number test: check if the system software is using the expected fill value
      iv. NaN test: check if the data are all NaNs
      v. Gap > 1-day test: report gaps greater than a day in the data
      vi. Actions:
         1. Use the stream status list to enter annotations in the database
vii. **NOT IN THE AUTOMATED ROUTINE**
   1. Unit test: check if the system software is using the expected parameters’ units
   2. Sampling rate test: test data file sampling record against the deployed sampling rate.

**d. System QC Flag Tests**
   i. Global Range QC Flag Test: Compare the min and max global range values in the system against the data time series, point by point.
   ii. Stuck_Value QC flags Test: Plot the data and examine the time ranges that failed stuck_value test
   iii. Spike_Test QC flags Check: Plot the data and examine the time ranges that failed spike test

3. **STEP 3:** Plot Data Files. Generate plots to visually assess data quality
   a. Use plot_timeseries.py to plot time series, depth profiles, and x-y-z plots.
      i. Data Availability:
         1. Available - Data available in uFrame
         2. Not Available - Instrument Failure
         3. Not Expected - Data lost in transmission, not recorded, or not telemetered
         4. Pending - Not currently produced, awaiting parser/driver development, or awaiting other parameter availability
         5. Missing - Not delivered to uFrame
         6. Expected - Awaiting instrument recovery
      ii. Data Quality:
         1. Pass - data passed QC tests, time-series are complete and reasonable
         2. Suspect - data failed one or more QC test and are otherwise suspect. Action: need further investigation.
         3. Fail - data failed all QC tests or are otherwise bad.
   b. Use timeline.py to generate timeline to visualize data quality flags produced by the automated test.
      i. QC Flags Quality:
         1. Pass - flags are properly applied to the time range
         2. Suspect - investigate and apply a fix
         3. Fail - fix flags

4. **STEP 4:** Summarize Output Results.
   a. Examine the output file from the automated data quality check the data plots and database notes to:
      i. Investigate data quality issues
      ii. Annotate data
      iii. Report system software bugs
      iv. Report on data metrics
   b. Tools:
      i. [Data Team Database]
ii. Data Check Tools
iii. Timeline Plotting tools
iv. Data Plotting tools
c. Redmine Parent Tickets:
i. Data Ingest for rest in Class effort
ii. Missing calibration sheets
iii. Missing uncabled raw data
iv. Issues encountered during rest-in-class data evaluation

Figure 2. Check Data Output file example table

Figure 3. Prototype timeline view of instrument availability and data quality.

Figure 4. Data availability flags at the instrument level

Figure 5. Data Quality flags at the parameter level
5. **STEP 5**: Make annotations in [Annotations GitHub repository](#).
   a. **Procedures**:  
      i. Add operational issue annotations at the subsite, node, and instrument level in the “Subsite.csv”  
         1. Add annotations with the following status only:  
            a. **NOTOPERATIONAL**  
         2. “Operational” time periods will be derived automatically.  
         3. If there are no issues at the subsite, node, and instrument level, there is no need to enter anything in this csv.  
      ii. Add issue annotations at the stream level in the “Stream.csv”  
         1. Add annotations with the following statuses only:  
            a. **NOTAVAILABLE**: time ranges that do not have issues, but are still under review  
            b. **PENDING_INGEST**  
            c. **NOT_EVALUATED**  
            d. **AVAILABLE**: time ranges that do not have issues and have been fully reviewed  
         2. Once a time range in a stream has been marked as “Available,” that means all parameters in that time range have been reviewed and “Pass”, unless there is a parameter level annotations that specifies otherwise. The state of “Pass” at the parameter level will be derived automatically.  
         3. Annotations should be added in chronological order, down to the second level and separated by one second.  
         4. Annotations made in the stream level should cover all points in time for which an instrument was deployed.  
      iii. Add annotations for data quality issues with science parameters in the “Stream-parameters.csv”  
         1. Add the following statuses only:  
            a. **SUSPECT**  
            b. **FAIL**  
         2. “Pass” time periods are derived automatically, once the review is complete and the “Available” status is set in the stream level csv.  
         3. If there are no issues with any of the expected parameters, there is no need to enter anything in this csv  
         4. Add availability status for engineering parameters, but only annotate if a parameter is missing. Availability will be derived from the stream annotations.  
      iv. Run **check_annotations** tool to check for basic errors in the csv files  
      v. Run **timeline plotting tool** and produce data quality quicklook report ([example](#)).  
      vi. Load annotation csv via GitHub to QC Database. (Need script to auto-annotate Operational, Pass in QC Database upon import into the QC database.)
vii. Push annotations from QC database to system via API

b. Vocabulary for statuses (taken from uFrame):
   i. NOT_OPERATIONAL
   ii. NOTAVAILABLE
   iii. PENDING_INGEST
   iv. NOT_EVALUATED
   v. SUSPECT
   vi. FAIL
   vii. PASS

![Diagram of annotation workflow and roll-up rules](image)

**Figure 6. Annotation workflow and roll-up rules**

c. General Notes:
   i. Issues should be annotated at the day level
   ii. If an issue occurs more than once within a 24 hour period, it receives a single annotation, bound by a 24 hour time period
   iii. If an issue occurs more than 24 hours after the first issue occurred, it receives a separate annotation
   iv. If a time period has frequent issues (e.g. every 48 hours an issue occurs over the course of a month), the entire month receives a single annotation
   v. All points in time for which an instrument was deployed will be annotated at the stream level. Only issues will be captured at the subsite, node, instrument, and parameter level
   vi. If a stream is marked as available, because it has been fully reviewed, and no issue annotation has been made at the parameter level, all remaining time ranges will be automatically marked as “Pass” for that parameter
vii. Instrument, node, and subsite timelines will be marked as “Operational” for all points in time during which a stream is marked as “Available”
viii. Notes can be added under the “Todo” column in a given .csv file
d. Annotations are not currently being pushed to the production system. Once that is a possibility, a mechanism will be added to automatically flag annotations that are ready to be pushed and match the production system annotation schema

5.3 New Cruise
Throughout the year, several cruises take place to deploy and recover instrumentation throughout the OOI. As part of its quality assurance role, the Data Team ensures that all necessary information and documentation has been added to the system. This work occurs prior to, during, and after a cruise.

Questions
- What happened during the cruise? (cruise plan, daily logs / quick look reports, final cruise report)
- What was deployed?
  - Were there any changes from the plan?
  - Have all the appropriate uFrame sheets been updated? (cruise, deployments, assets, calibrations, ingestion)
  - Are data available in the raw data directory or via streaming?
- Were any shipboard samples taken? Have they been processed and added to the system?
- Are the data from recovered platforms available in the raw data directory?

Workflows
For each cruise, there are a number of critical pieces of information that are required for the OOI data system (uFrame) to be able to ingest and process new data. This includes asset, calibration, cruise, deployment, and ingestion information. The responsibilities and steps required to enter this information are outlined in the Data Ingestion Procedures document (2100-60001). The key steps are also noted in Figure 2.
Additionally, the Data Team reviews the activities that occurred on each cruise to ensure that all necessary information has been added to the system, all reports and other information have been appropriately posted, and that both telemetered and recovered datasets affected by the cruise are available. These cruise review activities are logged by the Data Team to ensure completion. See Appendix 1 for an example checklist.

5.4 Final Deployment Review
Following each recovery of an instrument, the Data Team can provide a thorough and final review of the available datasets (both telemetered/streamed and recovered). This provides the best opportunity to review the availability and quality of each data set, and ensuring all issues have been resolved (if possible) and that appropriate annotations are available.

Questions
- Are the deployment information sheets complete and correct? (cruise, deployments, assets, calibrations, ingestion)
- Is all of the recovered data available in the raw data directory?
- Are all parameters and streams available from the system?
- How much of the data passed automated QC checks?
  - Are there any issues with the dataset (QC flags or fill values)?
- How much of the data looks reasonable?
  - How does the data compare with nearby sensors or other environmental data?
- Have any potential issues been added to Redmine for investigation?
- Have all identified issues been annotated in the system?
- Is the deployment data quality report completed?

Workflows
See Appendix 2 for an example report checklist
5.5 Deep Dive Questions

Questions
- (see Deployment Review above)
- Are there any anomalies or gaps in the dataset?
- Can paste in steps identified during the First in Class checks (algorithm, parser, etc.)
- Use instrument pages on oceanobservatories.org as a reference, and check that all information is included and correct – this needs to be systematic and consistent across all arrays

Workflows
TBD

Outputs
The team produces the following outputs.
- Data Availability Reports (% completeness, streams/parameters being reported, particles in the system)
- Data Quality Reports
- Report card for CI and MIOs (?)
- Redmine issue conversations
- Redmine report, including issues found and helpdesk open/closed
- Deep dive investigations
- Annotations (to users)
- Download statistics
- Forum statistics (?)

6. Evaluation Tools and Needs
- QC Portal
- Plotting scripts (with QC flags)
- Raw data repo check (24h change)
- Stream/Parameter availability check
- Ingestion scripts check
- Purge and reinjest tool (including for Cabled data; including partial purge that doesn’t require blowing away an entire reference designator)
- Tool to compare time stamps of port agent logs and files on the RSN archive to see where gaps can be filled (we don’t have regular access to the port agent logs)
Appendix 1: Example Cruise Report / Checklist

http://ooi.visualocean.net/cruises

<table>
<thead>
<tr>
<th>Item</th>
<th>MIO Submitted</th>
<th>Date Team Reviewed</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cruise Plan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Look</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asset Sheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calibration Sheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment Sheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ingest Sheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raw Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Live Ingestion Started</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cruise Report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cruise Photos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shipboard Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Water Sampling Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Summary
None

Notes
None
Appendix 2: Example Deployment Report / Checklist

List of documents by MIO:
Provided:
 Cruise plan
 Cruise logs
 Quick look cruise reports
 Instrument sampling settings
 Shipboard data
 Final cruise reports
Updated:
 Cruise sheet
 Assets’ OOI barcode
 Calibration sheet
Prepared:
 Deployment sheet
Communicated:
 Operation status
 Data upload to CI

List of documents by Data Team
Prepared:
 Ingestion sheet
 Annotations
Reviewed:
 Deployment sheet
 Data parsers
Reported:
 Availability issues
 Quality issues
Updated:
 Annotation
 Issues

List of documents by CI
 Aggregated data files (NetCDF format)
 Provenance data files (JSON format)
 Annotation data files (.csv format)

Data summary report
 Data information correctness
 Data availability
 Data completeness
 Data quality
 Data correction
 Data statistics
 Data visualization
 Data events
## Edit Review for CP01CNSM-MFD35-06-PHSEND000 — Deployment 1

<table>
<thead>
<tr>
<th>Status</th>
<th>Not Started</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewer</td>
<td>Lori Giazzo</td>
</tr>
<tr>
<td>Asset Sheet Reviewed</td>
<td></td>
</tr>
<tr>
<td>Calibration Sheet Reviewed</td>
<td></td>
</tr>
<tr>
<td>Deployment Sheet Reviewed</td>
<td></td>
</tr>
<tr>
<td>Ingest Sheet Reviewed</td>
<td></td>
</tr>
<tr>
<td>Raw Data Reviewed</td>
<td></td>
</tr>
<tr>
<td>Raw Data Url</td>
<td></td>
</tr>
<tr>
<td>Parameter Check</td>
<td></td>
</tr>
<tr>
<td>Availability Check</td>
<td></td>
</tr>
<tr>
<td>Quality Check</td>
<td></td>
</tr>
<tr>
<td>Environment Check</td>
<td></td>
</tr>
<tr>
<td>Percent Good</td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td></td>
</tr>
</tbody>
</table>