Space Details

Key:	SSDS
Name:	Shore Side Data System (600031 and 900823)
Description:	This is a project to build a data management system for the Monterey Ocean Observing System (MOOS).
Creator (Creation Date):	kgomes (Oct 17, 2006)
Last Modifier (Mod. Date):	kgomes (May 30, 2008)

Available Pages

- Welcome to the Shore Side Data System Project
  - Benthic Rover Data Management
  - Project Memos Minutes
    - MTM_2007_01_17
    - SSDS Strategy Meeting on January 22, 2007
    - Weekly Notes from January 12, 2006
    - Weekly Notes from January 5, 2006
  - SSDS Project Documentation
    - 2008 Abstract
    - 2008 Abstract Presentation
    - 2011 Proposal Notes
    - ALOHA
      - April 27, 2010
    - An example use of Graphs - FOCE
    - Data Simulator
    - Debugging quick look and contour wind stick plots
    - How to Configure Graphs
    - Ingest Architecture
      - Qpid Exploration
    - Installation and Development
      - Windows Installation
    - Instrument Swap on Oasis Mooring
    - M0 Text File Generation Scripts
    - Migration to Google Code Base
    - new-ssds.mbari.org Setup
    - ProjectRequirements
      - CIMT Requirements
        - 2004-04-20 CIMT SSDS Meeting Notes
      - MSERequirements
    - Publishing other non-SIAM data to SSDS
    - Republishing Data From SIAM Node
    - Services
      - Data Producer Services
      - Data Services

- SQL 2008 Upgrade and Move to Dione
- Testing TransmogrifyMDB and Ingest
- Transmogrify and Ingest Deployment
- UserInterfaces
  - DataContainer Importer
  - Device Inspector
  - Matlab 2008a Integration
  - Related Resources
  - RIA Technologies for the SSDS
- Tasks
  - Cleanup Configuration Management
  - Internal Application Consolidation

Welcome to the Shore Side Data System Project

This page last changed on Jan 05, 2011 by kgomes.

SSDS Products:

1. Production Web App

Project Documentation:

1. Documents
2. Memos and Minutes
3. Presentations
4. Purchase Orders

Tasks

1. Tasks **Deprecated, see Project Documentation**

Related Project Sites:

1. CIMT Web App

Related Links:

1. Alfresco Content
2. JIRA Bug Tracking

Benthic Rover Data Management

This page last changed on Mar 05, 2014 by kgomes.

This page documents the progress and status of Benthic Rover data management activities, as of the end of July, 2009.

Overview: Status of Benthic Rover Data Mgt Tasks

Right now Rover data is being copied (manually?) to shore. The intent is to have it all logged to SSDS. The proposal for doing so was previously circulated to the Rover team, and has a lot of detail that may be of interest, though some information is out of date.

The Rover data is stored in the project share for the Rover (900502.BenthicRover), under Rover.Deployment. This is not an ideal place for data; the permissions have to be set correctly to avoid accidentally losing the data, and I believe it can not be exposed for easy access via the web. The project may wish to consider moving it to a more accessible share, similar to the BIAUV share.

There are several tasks for Rover data management. The tasks and their status are as follows.

Convert most recent Rover data to new format

Some time ago we designed a new data format (changing both organization of files, and content for the data files). Past Rover data of value (through last October, the last pre-MARS mission) were converted to the new format. This data is stored under Rover.Documents/Project.Data/ReprocessedData/RoverData_2007-2008_NewFormat.

The Rover began collecting data again several weeks ago, when it was attached to the MARS node. That data is being uploaded to the Rover Project Folder on Tornado. It appears the current release of the Rover software writes data in this format. (nice, Rich!) See Converting Rover Data Formats below for more information.

Logging Rover data in SSDS

No Rover data is being logged to SSDS; this task is in progress as described below.

Originally the notion was to log Rover data in real time. Since the Rover will only be on MARS for a brief period, we no longer intend to do this, but will log all data retroactively instead.

Logging Rover images to SSDS

This task has not started. The approaches being considered are described below.

Converting Rover Data Formats

This task established a 'normalized' data file format, in which each mission (e.g., a cruise), deployment (when Rover is physically deployed off a ship, or spends a period of time doing a science or engineering project), plan (script or similar set of commands), or activity (something producing data from a particular device) had its own folder. Folder names indicate the type of folder, the sequence of the folder among others at the same level, and whatever identifier the original user wanted for that mission, deployment, plan, or activity. This provides an order and hierarchy for the data, and puts different kinds of data into different folders.

Each data file also has a standard format in the normalized scheme, making fields like timestamp the same for all Rover files.

The organization and renaming of files from the old format to the new format was performed manually, and only for those missions that appeared to have any data of interest that could be converted. The reformatting of records was done automatically via the formatDataFiles.pl script (checked in to svn:rover/

trunk/scripts/perl/). This script leaves filenames in an intermediate state; the user must then run undoFormatFiles.pl (in the same directory; has a terribly misleading name) to either finalize or revoke the changes. Not ideal, but I don't know that you'll have to run this code ever again.

It appears the MARS Roverdeployment is writing missions largely in the new format (yay!). Only the system log and optode file formats could be confirmed; currents data was not evident yet. Metadata about device IDs is missing from the first line of the data files; either this will need to be corrected, or the processing software will need to know what to use for a device ID when none is present.

There are some minor differences between the proposed directory layout and the new layout (most of these can be addressed trivially), and some new files are present that may require additional code to submit.

Submitting Rover Data to SSDS

The strategy for submitting Rover data was to submit it record by record to the SSDS system, associating the records with the corresponding device IDs using the XML at the top of the file. Metadata will be collected from the folder names, and submitted separately to SSDS. The hierarchy of the folders can be preserved in SSDS by making each folder a deployment in SSDS, thus allowing similar navigation within SSDS.

(We considered submitting all the original files of Rover data to SSDS, the way the AUVs do. I considered the files more of a transport mechanism, and also wanted to support real-time submission at some point, so this is not the current mechanism.)

So far we have successfully validated submitting individual test records to SSDS (not Rover test records, but the difference should be trivial). And I have a Perl script checked in, roverParse.pl, that can either submit records to SSDS, or log them to a file; this appeared to be working (logging to a file) when run against the reformatted data. All that would be required to submit the data to SSDS is setting a flag to submit data into the SSDS server, instead of logging it (recommend running against a test server first though, especially if you wanted to run it on the new MARS data, as it hasn't had much testing).

I have some notional changes on where to fit metadata management into that script. Some Perl was written to parse folder names, but it is in a very preliminary state. And I started to verify the metadata submission process using SSDS testing utilities, but there were challenges I haven't overcome. All this work was continuing up until my departure.

The next steps will be to (a) try parsing and submitting all the metadata information; (b) try submitting all the data records (hmm, might be better/faster to do this first?) (c) connect the data to the metadata, or vice versa. Since data goes into SSDS buckets according to the device that generated it, it will be important for the processing scripts to either know, or have a way to determine, those device IDs. (Paul McGill keeps a spreadsheet as current as he can that contains this information; make sure he updates the one that I added a sheet to, as it has a more thorough description of device IDs on all the deployments.)

The data records were most important data elements, but more data files appear to be in the data sets now (not sure which of the additional files are generated on the Rover). A decision will have to be made about which of these should be submitted to the SSDS. The team may also prefer to keep processed data separate from the original data, to avoid confusion.

Submitting Rover Images to SSDS

The Rover collects images from multiple cameras. Some of these are JPEG images, and some are RAW (Bayer) images. The intent was to either submit the images to SSDS directly, or store them in an accessible place and index them in SSDS. Unlike the BIAUV Image Processing Strategy situation, there are relatively few images for the Rover, so storing them directly in SSDS is not out of the question.

The exact technique for storing images was still being investigated, and the metadata for the images needs consideration.

Conversion of images

The RAW images must be converted to JPEG or other suitable format. Whether this is done before or after storing the images is at the discretion of the program, given the disk space required/available. Although no Unix script existed to do this when last investigated (in 2008), one may be available now.

Metadata for images

The image metadata in the raw images is entirely non-existent. There is not even a timestamp. Fortunately timestamps are maintained in the name of the image, but this is a very weak metadata system for such a critical piece of information.

The plan and recommendation is to follow similar image metadata post-processing as is performed for the Benthic Imaging AUV images (BIAUV Image Processing Strategy). Much of the same code could be reused, but the metadata will have to come from different places or analysis. (The fastest way to do this may be to generate netCDF files for the Rover data sets, as this could probably be done readily. Then much of the BIAUV processing could apply more directly.) Rover position should be assumed at first to be the same as the MARS node, with a large error bar of course.

The most fundamental and important modification to add metadata for images will be the timestamp, as all knowledge of the image depends on correct timestamps, and putting that metadata in the name is very weak, as noted previously.

Image storing technique

As discussed in the BIAUV Image Processing Strategy, it is not clear whether images should be stored directly in SSDS at all. Kevin Gomes and I were interested in trying this out to see how well it could work, snd I did a bit of work on Rover with it. It might be advantageous to use the same method as the BIAUV; the team may want to read that document before deciding on an approach. The detailed organization of the Rover data products, where images are in many different sub-sub-directories, may be a factor.

The first attempt to store images was via hex-encoded URLs (HTTP GET), but this failed due to the length (MBs!) of the URL.

A second suggestion, not yet tried, was submitting images via HTTP POST.

It is also possible to directly submit images via a Java interface. This is awkward with the Perl scripts, but a Java tool might do so readily.

Finally, it is possible to just store the images on an appropriately accessible share, and index them in the SSDS metadata.

XML and XSLT Files

Rich Henthorn and John Graybeal spent some time figuring out how to embed descriptions of the Rover deployed instrumentation in (a) an on-board description file or directory, and (b) the data logs of a mission. The notion we came up with was to document all the devices in XML files that are on board, and create a single master XML configuration file that points to the appropriate XML files (using XPath and XPointer). In addition, the first line of data files could contain a minimal XML segment containing the device IDs; this would be written by the device driver, and is needed for post-processing to submit the data records into appropriate device bins.

I'm not sure how much of those concepts are in the Rover mission code, nor how much will be added later. The XML files have been written and are checked in at svn:rover/trunk/xml and the RoverConfiguration subdirectory.

The Rover configurations can be described in a nice format using the XML files and nice XSLT transforms to convert the XML information into web pages. Three XSLT files, to transform the XML into descriptions or checklists of the Rover configuration, can be found at svn:rover/trunk/scripts/xsl. Example outputs are also there. See the README file for details.

These transformations can be run on the Rover itself using (for example) xalan or similar UNIX-generic XSL transformation software, but this environment has not been set up on the Rover, to my knowledge.

(Note Oxygen's xsltproc does not correctly handle the XPath/XPointer, and so will not work; you must set the XSLT processor as part of the Transformation configuration settings.)

Project Memos Minutes

This page last changed on Jan 22, 2007 by kgomes.

Mintues from SSDS (or related) Meetings:

Weekly Meetings

1. Weekly Notes from January 5, 2006
2. Weekly Notes from January 12, 2006
3. Weekly Notes from January 26, 2006
4. Weekly Notes from February 2, 2006
5. Weekly Notes from February 16, 2006
6. Weekly Notes from March 2, 2006
7. Weekly Notes from March 9, 2006
8. No meeting on March 16, 2006
9. Weekly Notes from March 23, 2006
10. Weekly Notes from March 30, 2006
11. Weekly Notes from April 6, 2006
12. Weekly Notes from April 13, 2006
13. Weekly Notes from April 21, 2006
14. Weekly Notes from April 27, 2006
15. No Meeting on May 4, 2006
16. No Meeting on May 11, 2006
17. Weekly Notes from May 18, 2006
18. Weekly Notes from May 25, 2006
19. Weekly Notes from June 1, 2006
20. Weekly Notes from June 8, 2006

Other Meetings

1. OSG Meeting Notes from January 12, 2006
2. Mooring Meeting Notes from January 24, 2006
3. Mooring Meeting Notes from January 31, 2006
4. Mooring Meeting Notes from February 14, 2006
5. Mooring Meeting Notes from February 27, 2006
6. Mooring Meeting Notes from April 05, 2006
7. MOOS Test Mooring Meeting (January 17, 2007)
8. SSDS Strategy Meeting on January 22, 2007

MTM_2007_01_17

This page last changed on Jan 17, 2007 by kgomes.

- SSDS tasks for April is to set up the data processing for Vertical Profiler (McLane)
- Andy is not available for any work. Hmmm ... we need Andy to help setup the processing
- December we will be replacing the Axis BIN.

SSDS Strategy Meeting on January 22, 2007

This page last changed on Jan 22, 2007 by kgomes.

Agenda

1. Current SSDS Status
    a. MSE
    b. M1/M2
    c. CIMT
        i. Funded another year (June 2008)
    d. AUVCTD
    e. Post processing of all of the above
2. Documentation (WIKI migration)
3. To dos in JIRA
4. Projects involving SSDS, their status, and how SSDS contributes
    a. MOOS/MSE
        i. Post processing of profiler data
        ii. Metadata and other user interfaces
            i. HOOVES and metadata editing
        iii. Camera integration (data stream)
        iv. Watch circle alarm
        v. Processes and procedures development (metadata is inconsistent) on MSE - operations
        vi. Andrew's ACE stuff (Tom) User interface and business logic.
    b. OASIS
        i. M! Turn
        ii. M2 Turn
        iii. NDBC Cleanup (CIMT mooring)
        iv. Ops procedures and people/tools/training (XML and turns)
    c. CIMT
        i. 2 turns
    d. AUVCTD
        i. ssdsLoads is broken
        ii. HOOVES
    e. SENSORS
        i. Connect the instrument interface prototype up to SSDS on MARS (adapter or alternate ingest mechanism)
    f. Others?
        i. ROVCTD
        ii. LOBO
        iii. BOG
        iv. do we put more in SSDS?
        v. CO2 Initiative?
        vi. CN Initiative?

    ---

    g. Data Aggregation
        i. Meetings
        ii.
    h. UW to Microsoft Proposal (RCO)
        i. "We (TBD) will bring SSDS up to UW and operate". SSDS collects and then Triton (WorldWind++) visualizes it. It will tie into the .NET 3.0 Workflow Framework.
    i. UCSD Response to CI IO
        i. FIREWALLED
    j. MBARI Response to CI IO
        i. FIREWALLED
    k. CSO IO
        i. Possibly a temporary system to serve as a CI proxy.
    l. LOOKING (20 days)
        i. Jim wants to merge AOSN Data System and SSDS and then connect COOP and MoQUA to that.
        ii. Federated observatory prototype include components of SSDS as repository (catalog, processing and preserving data)
    m. AOSN

        i. If Jim uses looking for merge, AOSN will use that. Dependent on LOOKING.
    n. SNMP is an activity
        i. ESB at NCSA
        ii. ESB at MBARI
        iii. Sending packets back and forth.
        iv. To get diagram
        v. SENSORS build instrument interface and connect to ESB.
        vi. SSDS is on ESB to collect and catalog data.
    o. NCSA Proposal to NSF to extend SNMP work (keeps it going)
        i. Product are offered to community (OpenSource)
    p. Nekton Research/NOAA
        i. MBARI share SSDS code
        ii. Nekton will package SSDS with 2 AUVs delivered.
        iii. Get Nekton engineers to be self reliant with SSDS.
        iv. Would they be interested in AUV science data processing?
        v. Would need something like AUVCTD portal code to convert Nekton AUV data to XML/SSDS Data model.
        vi. Conference call with them.
    q. NOAA Proposal (Francisco)
        i. 
    r. CenCOOS  Related to NOAA Proposal and Data Aggregation?
    s. Dalhousie 
        i. Waiting on us to deliver exportable code (they can compile and run).
        ii. Preconfigured properties for building for them
        iii. Tracking metadata and data for their moorings.
        iv. They might provide some user interfaces for SSDS.
        v. Haven't talked to them in a while.
5. What are our goals for SSDS?
    a. Internally
    b. Externally
    c. Short Term
    d. Long Term
6. How do we transition to support and what still needs to be done?
    a. Metadata editing/management (and other UIs)
    b. Plan for the transition (written) and resources needed
    c. Move core to support or vice versa.
    d. Documentation
    e. Move to Subversion (get rid of branches)
    f. Plan and agreement for future work/development
    g. Operational interfaces (component health/statistics/log crawlers)
    h. Improve data stream monitoring system (NAGIOS a possibility)
    i. Controlled vocabularies (instrument types) overlap with MMI?
    j. Load all OASIS data.
    k. Workflow for instrument data streams.
7. How do we best open source SSDS?
    a. Current licensing website 
        i. Did not formalize license yet
        ii. Not utilized
    b. What is still to be done to open source?
        i. Pick a license
        ii. Decide on distribution mechanism (talk to Brian and Luis)
        iii. Clean up code/copyright.

Weekly Notes from January 12, 2006

This page last changed on Dec 28, 2009 by kgomes.

Weekly Meeting Agenda

Updates Around The Table

Mike M

1. New HOOVES/RCP
   - On hold
   - Property Change listeners on Metadata classes
2. Perl Module
   - Writing unit tests for the generated module
3. M0 Data processing
   - On schedule
   - Talked to Reiko, still out sick, but she responded with some stuff.

Andrew

1. ACE update (waiting on jar issue for SIAM)
   - SIAM has asked for more of Andrew's time
2. XML Editor
   - On hold tll after architecture roll out
   - if rollout happens soon, might be able to get done for John to use for XML for MSE Surface node.
3. Caress data
   - Half way finished, will finish on contingency time. Will do after perl module.
   - Should be straightforward
4. ANTLR (Perl module)
   - This way was too much work
   - Can we provide an array of Java classes, and you would carry out generation in language you are generating in.
   - Other languages will be on a as needed basis.
5. Benthic Rover Data Management

John

1. M1 Turnaround status
   - M0 puck was reburned
   - Waiting on the new architecture before changing.
2. JSF/Web app flow
3. Documentation Status
4. Will be going to Hawaii

Luis

1. MMI status
   - Luis and I have working on new architecture build running on MySQL.

Mike G

1. ASAP Data Systems Update

Kevin

1. SSDS Hours/Project Plan update
   - Kevin: 126 days
   - John: 40 days
   - Andrew: 20 days

    - Mike: 0 days
2. Need project schedule for 2006
3. Development
    - General
        - New machine has arrived, Pete at Macworld, probably set it up next week
        - Need to draft up deployment and migration scenario for new architecture/new machine
    - Web Application
        - Sputtering to life (http://ssdsdevpc:8080/ssds)
        - Meeting with ops guys this morning
    - Core Metadata
        - DTS Job to copy data from old architecture to new.
        - DTS job was fixed (need perl installed on computer running Enterprise Manager, not SQL servers)
    - Core Data
    - Clients
    - Model Integration
4. MSE
    - Requirements definitions
    - Keith to send out email stating policy will be open if nobody says otherwise.

Action Items

Weekly Notes from January 5, 2006

This page last changed on Jan 19, 2007 by kgomes.

Weekly Meeting Agenda

Updates Around The Table

Mike M

1. New HOOVES/RCP
   - Struggling, but progress.
   - org.mbari.hooves in own CVS project "hooves"
   - Data binding framework that allowed model object to be used in forms. Adding listener support for property changes.
   - So overall, RCP is going well.
   - HOOVES will be on hold for a bit for M0 data.
2. DTS Job to copy data from old architecture to new.
   - Asking Neil to port that to a DTC job on Fog (Kevin needs to follow up with Neil, DTS jobs broken)
     - Look in master script for server polyp.
3. M0 data request/more data processing
   - Will be impacted by meeting tomorrow.
   - Request is to access data from anytime period.
   - These are the post processed data sets.
   - Will meet with Reiko
   - Needs Andrew's work on the auto Perl module and will bite the bullet and go with new architecture.
   - Told John R about a week.
   - Notifications are enabled by listener properties, but does not address remote clients.
   - This could be based on a message bus architecture. dev.java.net project called event bus.

Andrew

1. ACE update
   - Kent and Tom and list of to dos and pretty much done. Waiting on them. Still issue with tracking jars and how to deal with them. Not currently working on it.
2. XML Editor
   - Fallen by the wayside. Needing time to work on. Pick back up after architecture roll out.
3. Caress data
   - Half way finished, will finish on contingency time. Will do after perl module.
   - Front end is done, Google maps with our bathymetry. Will need to sit down and go over to make sure it can extend.
4. ANTLR
   - General parser that gives SSDS code parse tree. Then generate perl modules. Moving into the SSDS code base. Other languages will probably be fairly straightforward.
5. Benthic Rover Data Management
   - In conception right now.
   - Data should be showing up in mid 2006.

John

1. M1 Turnaround status
   - Waiting on the new architecture before changing.
   - M0 puck needs to be reburned.
2. We need to check to see if Paul has already re-burned the PUCK (yes, they have)
   - Tomorrows meeting:
   - Meet with John at 8:00 tomorrow.
3. JSF/Web app flow (This is back to John)
   - Mike SWT components should be able to be exported to web pages.

- Getting up and running on development environment, then focus on JSF.

4. Documentation Status
   - Nothing but with JSF discussion
5. NSF Proposal (likely to pull John off SSDS till February)
   - Not going forward, John back on SSDS.

Luis

1. MMI status
   - Finished final workshop report.
   - SSDS problems with services (no progess).

Mike G

1. ASAP Data Systems Update

Kevin

1. SSDS Hours/Project Plan update
   - 2005
     - 320 allocated for core and we were 43 days over, if you adjust out 60 days MMI time, we were under by 17.
     - As a project we were over by 87 days (billable to SSDS)), if you adjust out 60 days MMI time, we were over by 27.
       - Kevin (144 allocated, spent 156: 12+)
       - John (90 allocated, spent 114: 24+) This is not correct, 60 days went against MMI
       - Andrew (82 allocated, spent 93: 11+)
       - Mike (36 days)
       - Nancy Barr (3 days)
       - Rich (3 days)
       - Dorothy (13 days)
       - Karen (4 days)
     - 2006 Allocations
       - Kevin: 126 days
       - John: 40 days (is this correct?)
       - Andrew: 20 days
2. Need project schedule for 2006(this week)
3. Development
   - General
     - New machine order HP dual athalon with Red Hat
       - Need to draft up deployment and migration scenario for new architecture/new machine
     - Web Application
       - Sputtering to life (http://ssdsdevpc:8080/ssds)
     - Core Metadata
     - Core Data
     - Clients
     - Model Integration
4. MSE
   - Need to document requirements for data access (by tomorrow!)
   - No update on data policy
5. ASAP
   - Need to meet with Mike G to discuss schema changes.

Action Items

1. Getting new architecture rolled out is critical (time wise)
2. Finish up DTS job migration
3. Get project plan done and identify tasks for developers
4. Look at eventbus: as possible solution for remote model object notificaton (and other SSDS stuff).
5. Finish getting John's development environment going
6. Get ASAP update from Mike G.

SSDS Project Documentation

This page last changed on Feb 03, 2012 by kgomes.

Abstracts and Proposals

1. MOOS Project
    a. 2001 MOOS Project Proposal
    b. 2002 MOOS Project Proposal (Phase 2 Feedback)
    c. 2003 MOOS Project Proposal
    d. 2004 MOOS Project Proposal
    e. 2006 MOOS Project Proposal
    f. 2007 MOOS Science Experiment Proposal
    g. 2008 MOOS Upper Canyon Experiment
    h. 2009 MOOS Upper Canyon Experiment
    i. 2010 MOOS Upper Canyon Experiment
    j. 2011 MOOS Upper Canyon Experiment
2. SSDS Specific
    a. 2000 MOOS Data Management Proposal
    b. 2008 SSDS Hardening Project
        i. 2008 Abstract (Word)(PDF)
        ii. 2008 Abstract Presentation (PPT)
        iii. 2008 Abstract Feedback (PDF)
        iv. 2008 Proposal (Word)
        v. 2008 Work Breakdown Structure (Excel)
        vi. OmniPlan Document (HTML)
        vii. OmniPlan Document (ZIP)
    c. 2011 Data Security And Policy Project
        i. 2011 Abstract (Word)
        ii. 2011 Proposal (Notes)

Notes and memos

1. 2002-03-14 SSDS ISI Interface Meeting Notes
2. Weekly Notes from January 5, 2006
3. Weekly Notes from January 12, 2006
4. Weekly Notes from January 26, 2006
5. Weekly Notes from February 2, 2006
6. Weekly Notes from February 16, 2006
7. Weekly Notes from March 2, 2006
8. Weekly Notes from March 9, 2006
9. No meeting on March 16, 2006
10. Weekly Notes from March 23, 2006
11. Weekly Notes from March 30, 2006
12. Weekly Notes from April 6, 2006
13. Weekly Notes from April 13, 2006
14. Weekly Notes from April 21, 2006
15. Weekly Notes from April 27, 2006
16. No Meeting on May 4, 2006
17. No Meeting on May 11, 2006
18. Weekly Notes from May 18, 2006
19. Weekly Notes from May 25, 2006
20. Weekly Notes from June 1, 2006
21. Weekly Notes from June 8, 2006

Other Meetings

1. OSG Meeting Notes from January 12, 2006
2. Mooring Meeting Notes from January 24, 2006
3. Mooring Meeting Notes from January 31, 2006
4. Mooring Meeting Notes from February 14, 2006
5. Mooring Meeting Notes from February 27, 2006
6. Mooring Meeting Notes from April 05, 2006

7. MOOS Test Mooring Meeting (January 17, 2007)
8. SSDS Strategy Meeting on January 22, 2007

Papers and Presentations

1. 2001 Standard Metadata and Data Formats which was presented to the ISI group to frame the discussion of what type of metadata we would use in the ISI system which would then get into the SSDS System.
2. 2006 Oceans Conference Paper
3. 2006 Oceans Conference Presentation

Products

Design

1. Requirements
2. Transmogrify and Ingest
    - Architecture
    - Deployment
    - Testing
3. Services
4. Client
    - Data Simulator
5. User Interfaces

Developer

1. Installation and Development
2. Migration to Google Code Base

Operational

1. new-ssds.mbari.org Setup
2. Installing RabbitMQ (AMQP) on RHEL5
3. OASIS Mooring Data Processing Overview
4. Instrument Swap on Oasis Mooring
5. OASIS Mooring turn
6. Republishing Data From SIAM Node
7. Publishing other non-SIAM data to SSDS
8. Analyzing signals from MARS using SSDS and Matlab
9. How to Configure Graphs
10. An example use of Graphs - FOCE
11. Debugging quick look and contour wind stick plots
12. SQL 2008 Upgrade and Move to Dione

Other installations

1. USC
2. ALOHA
3. NREL
4. SRVI

2008 Abstract

This page last changed on Jul 13, 2007 by mccann.

Abstract

In the years to come, the oceanographic community faces a unique challenge related to ocean observatories. Even with the many legacy observatory systems operating today and with OOI observatories on the horizon, there is a large gap in the community between operational instruments and finished data products. Even if an instrument is part of an ocean observatory, there is no guarantee in today's mix of systems that the data will be available for processing and dissemination. MBARI has developed technology to help fill that gap both within the walls of MBARI and in the external community. Originally a component of the MOOS project, the Shore Side Data System has become an integral component of several MBARI core (and some non-MBARI) data streams and successfully bridges that gap between instrument and finished data products.

As it was envisioned during its development, the SSDS is now successful in managing all the metadata surrounding instruments, their deployments, and the data they produce. Furthermore, the SSDS tracks specific versions of software that produce data sets such that complete provenance of a data set can be provided. This type of system is unique within the oceanographic community and has served MBARI's operational data management needs well by allowing us to remove metadata assignments from data processing software resulting in more reusable and efficient code. Examples of this can be seen in the Mooring netCDF plot pages where a single module of software is used to process data from diverse multiple instruments into a common format. Before SSDS this kind of processing was done by multiple groups and individuals resulting in often incompatible data sets. One approximate measure of efficiency gained is that one programmer can now do the work of what used to take 3 or 4 people. It is currently responsible for managing the metadata and data for the following systems and their post processed data products:

1. MO Mooring (CIMT)
2. M1 Mooring
3. M2 Mooring
4. MSE Mooring (all four nodes)
5. AUVCTD
6. Bruce Howe's Aloha Mooring CTD and flourometer (ADCP will be soon).

In addition to its internal success at MBARI, the SSDS capabilities have been deemed desirable to the ORION OOI program and is, in fact, part of the OOI CyberInfrastructure proposal. Also, if MBARI wins the CGSN proposal, it is envisioned that they will need a system to "bridge" their development with that of the CyberInfrastructure. Because both the CI and the CGSN are being developed in parallel, it is likely that the CGSN will need some data management before the CI is available for that functionality. The SSDS could provide a system that the CGSN team could develop against the help them get started on the highest risk elements of the CGSN-CI interface.

In order to continue to support our internal data needs at MBARI and provide the most value to the external community, we are proposing more work on SSDS to add functionality to existing components as well as developing new pieces to complete the SSDS package. Project resources are being requested in 2008 for tasks that include:

- Improve metadata editing capabilities and client applications
- Provide database integrity checking tools
- Provide more useful and concise query and operational views of data producing systems
- Conduct maintenance on our existing and growing archive of data and metadata
- Allow SSDS to be distributed as an open source project

A detailed list of tasks is available on the project Wiki: http://oceana.shore.mbari.org:8081/display/SSDS/Tasks

Word document submitted 10 July 2007.

Criteria

This will be an infrastructure project. The criteria for infrastructure project evaluation is the following:

1. Importance: Does the project address an important problem in oceanographic research?
    a. There is a gap between ocean instrumentation and data management systems and applications
    b. SSDS (with SIAM) has been filling that gap
    c. Automatic capture of all metadata
    d. Capturing data provenance
2. Uniqueness: How unique is this contribution and well-suited for undertaking at MBARI?
    a. Not really an undertaking, but it operational at MBARI
    b. Due to its uniqueness it is being considered as a component in the ORION CI IO and external interests (Dalhousie, SOPAC)
3. Timeliness: Why should this project move forward now? What are the drivers?
    a. This is the year to break SSDS out of MOOS and have it stand on its own legs.
    b. It is clearly an operational component, while the future of other MOOS technology is not clear
    c. This is the year to include non-MOOS inputs/outputs to make it a easier to use institutional asset
4. Strategic Plan: Does the project demonstrate relevance to MBARI's strategic plan?
    a. Yes, particularly transfer of knowledge to external community.
    b. Particularly well position to help with OOI (both CI and CGSN if we win)
    c. Facilitates the response to opportunities to pass on data and understanding gained in pursuit of MBARI's research plan to organizations overseeing the environmental health of Monterey Bay and other locales
5. The Team: Is the team appropriate for the work, are they available, and are they committed?
    a. Team would consist mainly of Kevin Gomes and Mike McCann (with support from others as more data streams are integrated)
6. Prior Productivity: Has the project leadership been successful with prior support?
    a. SSDS has been successful to date and is the reason we are seeking to push SSDS outside of the MOOS envelope
7. Does the project demonstrate improvements in operation from year to year?
    a. Yes, this past year has seen large improvements in robustness and support for MSE development team. Many processes are moving to depend on SSDS (Mike's processing, Fred's OASIS - M0, M1, M2, NDBC Export, UW/Aloha mooring, WHOI used for MTM3 cable)
8. Does the effort have a significant impact on an important MBARI activity?
    a. Yes, currently supporting M0/CIMT, M1, M2, AUVCTD, UW, MARS/SENSORS Prototype, Could impact CGSN award and serve as bridge between CGSN development and CI development.
9. Does the project team periodically assess the needs or requirements of its beneficiaries?
    a. Definitely, we are constantly fielding requests from Engineering, Operations, Science, and the external community, but we are limited to respond by resources.
10. Will the effort benefit a large number of users?
    a. Operations: Better instrument management and operations status monitoring
    b. Science: More/Better interfaces to find and utilize data and associated processing and resources
    c. Support Engineering: Cut time to manage mooring data streams and data availability to outside community
    d. External community: get SSDS code base out there (this also cuts our time to fields requests from the community).

Salient points from the Strategic Plan:

1. Our capacity for understanding the complexity of the ocean, and for forecasting a realistic view of its future that we will partially create, is limited by the lack of technology for observing the ocean and maintaining a sustained presence in that harsh environment.
2. Goal: Transform and advance understanding of the most significant unsolved problems in oceanography by developing, adapting, and demonstrating innovative technologies.
3. Goal: Utilize those developments to discover and understand how the natural system operates, responds to, and interacts with anthropogenic influences.
4. Goal: Transfer the knowledge gained and the technology developed to communities outside of MBARI, including policy makers, government laboratories, resource managers, and the public.
5. MBARI technology is in demand for adoption by groups external to the institution, and that demand is met through external partnerships, licensing, copying, or other strategies as appropriate.
6. Look at: Natural rhythms of the complex ocean systems (Box 4), such as quantifying and understanding variability in the ocean food web on the seasonal, El Niño, and North Pacific Decadal Oscillation (PDO) time scales (emphasize time-series here).
7. Research Actions: Develop a data archive for Monterey Bay that can be easily accessed by users who are not data providers and which can be integrated seamlessly with related data sets from the larger oceanographic community.

8. Strategy B1: Participate in national initiatives that are aligned closely with MBARI's strategic plan and technology developments (Box 9), such as the National Science Foundation's Ocean Observing Initiative and National Oceanic and Atmospheric Administration's Ocean Exploration Program.
9. Strategy D2: Be alert for opportunities to pass on the data, models, and understanding gained in pursuit of MBARI's research plan to organizations overseeing the environmental health of Monterey Bay and other locales. (Box 11).

2008 Abstract Presentation

This page last changed on Jul 13, 2007 by kgomes.

Taking the text from the 2008 Abstract, an outline for the presentation might look like:

Outline

1. SSDS Requirements Under MOOS
    a. Store data streams from MOOS instruments
    b. Store and manage instrument metadata (catalog) for MOOS
    c. Track instrument lifecycles
    d. Operational Health and status
    e. Query catalog for data discovery
    f. Make data available (generally) in raw, transformed and post processed formats
    g. Provide metadata/data added value
    h. Track data provenance
2. Current SSDS status (meeting the requirements)
    a. Raw data available and keeps legacy programs in tact.
    b. All instrument metadata cataloged in SSDS
    c. MOOS Data/metadata management for:
        i. M0 Mooring (CIMT)
        ii. MTM1 Mooring (Closed)
        iii. MTM2 Mooring (Closed)
        iv. MTM3 Mooring (Closed)
        v. MSE Mooring (all four nodes)
        vi. M0 Mooring processing/provenance tracked
        vii. MSE Mooring processing/provenance tracked
3. SSDS Beyond MOOS
    a. Data/Metadata management for:
        i. M1 Mooring
        ii. M2 Mooring
        iii. AUVCTD
        iv. Bruce Howe's Aloha Mooring CTD and flourometer (ADCP will be soon).
    b. AUVCTD metadata cataloged, raw data transformed, data provenance tracked
    c. OASIS Mooring data and metadata stored and cataloged, data provenance tracked
    d. Bruce Howe's Aloha Mooring (Puget Sound) data and metadata cataloged
4. Uniqueness in the Community and external impact
    a. Large amount of effort goes into
    b. Most "data applications" rely on some structured data and focus more on specific analysis of data
    c. With some form of observatory middleware (like SIAM), SSDS can track and present the operational aspects of the observatory (instrument lifecycles)
    d. Data provenance is something that is lacking severely. Workflow tools exists, but rely on existing services and ties them together (SSDS provides those services and metadata to make workflow go). Most tools work on user's constructing data provenance and saving them, not dynamically tracked by system.
    e. Evidence of this is that SSDS is specified as a component for the ORION CyberInfrastructure.
5. Future for SSDS
    a. SSDS has demonstrated value outside of MOOS
    b. SSDS makes internal data management easier (core data stream management easier 1 person can do work of 3-4 developers).
    c. Manage more of our internal data
    d. Track more data processing.
    e. Move more core data to SSDS.
6. What are we hoping to do?
    a. improving metadata editing capabilities and client applications
    b. redeploying database integrity checking
    c. providing more useful and concise query and operational views of data producing systems
    d. maintaining our existing and growing archive of data and metadata
    e. distributing SSDS as an open source project

My guess for questions (and some thoughts on answers)

1. How does this work support the strategic plan?
    a. I extracted the relevant points from the strategic plan at the bottom of this page. I think you can probably answer them but if we want to hash over them some more, we can.
2. How is this related to the Data Aggregation proposal? I also included some answers below to the Criteria used for project evaluation. Feel free to add/remove.
    a. Data Aggregation is focused on taking curated data products and attaching powerful interfaces on top of them. Data Aggregation will not cover the operational aspects of ocean observatories. They don't care about engineering of operational data. SSDS should provide base data from which Data Aggregation will derive its tailored data sets from.
3. Why haven't we made more progress on science related user interfaces?
    a. Science requirement for MSE were very difficult to extract (some instruments were new to scientists) and a large portion of the SSDS time for 2007 was used by other software components in MOOS.
4. What is SSDS' role in the ORION CI work?
    a. It is largely going to be used for metadata capture and catalog for observatory and instrument information and life cycle. It will capture and store observatory and instrument metadata (and some data) and make available to the CyberInfrastructure and it's users.
5. What is the status of the Asset Tracking work?
    a. Close to being finished and will finish during the last half of 2007 (2007 was extremely front loaded with other projects). Asset tracking helps support this follow on work.
6. How much time will be requested?
    a. Largely this will come out of the proposal writing, but we should have some guess here (30 days me, 30 days you? ... total SWAG at this point).
7. Will this be the last year of the project? (or, when will SSDS be 'done'?)
    a. Not this year, but we see future development being to support direct user requests, not infrastructure. Work on SSDS most likely would be line items in other projects to support domain specific aspects or requests.
8. Has anyone else showed interest in SSDS?
    a. Yes, Dalhousie is interested in it for observatory management and SOPAC was interested in the cataloging component for managing bathymetry.

Criteria

This will be an infrastructure project. The criteria for infrastructure project evaluation is the following:

1. Importance: Does the project address an important problem in oceanographic research?
    a. There is a gap between ocean instrumentation and data management systems and applications
    b. SSDS (with SIAM) has been filling that gap
    c. Automatic capture of all metadata
    d. Capturing data provenance
2. Uniqueness: How unique is this contribution and well-suited for undertaking at MBARI?
    a. Not really an undertaking, but it operational at MBARI
    b. Due to its uniqueness it is being considered as a component in the ORION CI IO and external interests (Dalhousie, SOPAC)
3. Timeliness: Why should this project move forward now? What are the drivers?
    a. This is the year to break SSDS out of MOOS and have it stand on its own legs.
    b. It is clearly an operational component, while the future of other MOOS technology is not clear
    c. This is the year to include non-MOOS inputs/outputs to make it a easier to use institutional asset
4. Strategic Plan: Does the project demonstrate relevance to MBARI's strategic plan?
    a. Yes, particularly transfer of knowledge to external community.
    b. Particularly well position to help with OOI (both CI and CGSN if we win)
    c. Facilitates the response to opportunities to pass on data and understanding gained in pursuit of MBARI's research plan to organizations overseeing the environmental health of Monterey Bay and other locales
5. The Team: Is the team appropriate for the work, are they available, and are they committed?
    a. Team would consist mainly of Kevin Gomes and Mike McCann (with support from others as more data streams are integrated)
6. Prior Productivity: Has the project leadership been successful with prior support?
    a. SSDS has been successful to date and is the reason we are seeking to push SSDS outside of the MOOS envelope

7. Does the project demonstrate improvements in operation from year to year?
    a. Yes, this past year has seen large improvements in robustness and support for MSE development team. Many processes are moving to depend on SSDS (Mike's processing, Fred's OASIS - M0, M1, M2, NDBC Export, UW/Aloha mooring, WHOI used for MTM3 cable)
8. Does the effort have a significant impact on an important MBARI activity?
    a. Yes, currently supporting M0/CIMT, M1, M2, AUVCTD, UW, MARS/SENSORS Prototype, Could impact CGSN award and serve as bridge between CGSN development and CI development.
9. Does the project team periodically assess the needs or requirements of its beneficiaries?
    a. Definitely, we are constantly fielding requests from Engineering, Operations, Science, and the external community, but we are limited to respond by resources.
10. Will the effort benefit a large number of users?
    a. Operations: Better instrument management and operations status monitoring
    b. Science: More/Better interfaces to find and utilize data and associated processing and resources
    c. Support Engineering: Cut time to manage mooring data streams and data availability to outside community
    d. External community: get SSDS code base out there (this also cuts our time to fields requests from the community).

Salient points from the Strategic Plan:

1. Our capacity for understanding the complexity of the ocean, and for forecasting a realistic view of its future that we will partially create, is limited by the lack of technology for observing the ocean and maintaining a sustained presence in that harsh environment.
2. Goal: Transform and advance understanding of the most significant unsolved problems in oceanography by developing, adapting, and demonstrating innovative technologies.
3. Goal: Utilize those developments to discover and understand how the natural system operates, responds to, and interacts with anthropogenic influences.
4. Goal: Transfer the knowledge gained and the technology developed to communities outside of MBARI, including policy makers, government laboratories, resource managers, and the public.
5. MBARI technology is in demand for adoption by groups external to the institution, and that demand is met through external partnerships, licensing, copying, or other strategies as appropriate.
6. Look at: Natural rhythms of the complex ocean systems (Box 4), such as quantifying and understanding variability in the ocean food web on the seasonal, El Niño, and North Pacific Decadal Oscillation (PDO) time scales (emphasize time-series here).
7. Research Actions: Develop a data archive for Monterey Bay that can be easily accessed by users who are not data providers and which can be integrated seamlessly with related data sets from the larger oceanographic community.
8. Strategy B1: Participate in national initiatives that are aligned closely with MBARI's strategic plan and technology developments (Box 9), such as the National Science Foundation's Ocean Observing Initiative and National Oceanic and Atmospheric Administration's Ocean Exploration Program.
9. Strategy D2: Be alert for opportunities to pass on the data, models, and understanding gained in pursuit of MBARI's research plan to organizations overseeing the environmental health of Monterey Bay and other locales. (Box 11).

2011 Proposal Notes

This page last changed on Aug 02, 2010 by kgomes.

This body of work was to try and identify what SSDS changes need to be made to implement some sort of security and policy enforcement for the SSDS. The first thing to do was to try and gather information about what people were looking for in access restrictions and such for their data.

Tasks

Container-based security

1. a. Setup security domain to work with I.S.'s Centrify system
   #

Contextual-based security

Attribution Infrastructure

Interview Notes

Kanna Rajan (CANON)

I talked to Kanna about data access for CANON and his feeling was that it should not be open to the entire world, but within a group of collaborations, everyone should have access to the data that is part of the collaboration. Sort of the once you're in, you're in idea.

Francisco Chavez (CANON, BOG, Mooring)

1. Francisco mentioned that a sort of standard data policy for academics is that raw data is embargoed for 2 years, after which the PI makes it available to the public.
2. Ideally, there would be some way to automatically track all citations of data that people use for publications.
3. He felt that there would probably be some limited number of options that data providers could choose from and apply to their data. For example:
   a. Option 1: Data available to all
   b. Option 2: X Number of days embargo which nobody but the PI has access to the data after which it will be made public
   c. Option 3: X Number of days embargo which the public does not have access to the data, but a select group of collaborators might (defined by the PI). After the X number of days, that data would be available to the public.
   d. Option 4: Different groups of users have different dates of embargo. Group A has immediate access, Group B has 1 year embargo, public has 2 year embargo for example.
4. He mentioned that maybe we should look at the policies used by the Climate Data Center
5. He felt there should be some standard acknowledgement clause that tells people they need to cite the sponsors of the data they are utilizing.
6. He felt there should be some granularity within CANON to control access to various data sources (this goes against what Kanna was saying).
7. We should be able to remove people from the group of collaborations and thus remove access to the data.

Dave Caress (MDUC CTD, MDUC Navigation, Mapping AUV)

1. Some data available to all, some to collaborators
2. Other data should be embargoed

Jim Barry (MUCE, BI AUV)

Charlie Paul (MUCE)

1. Some data sources (instruments) can have their data available to the general public
2. Some data sources (instruments) will only be available to specific people (could be maintained in groups)
   a. This data might be under an embargo of some time frame

3. In all cases, MBARI and the PI should get cited when the data is used.

Bill Ussler (MUCE)

Ken Smith (Benthic Rover, BI AUV)

Chris Scholin (CANON, ESP)

Chris Grech (Ship/ROV Data)

Steve E. (Ship/ROV Data, MARS Engineering)

Nancy Jacobsen (Video)

Craig Dawe (MARS Engineering Data)

Paul McGill (MOBB)

Andy Hamilton (Power Buoy)

Ed Peltzer (FOCE)

Peter Brewer (FOCE)

Mapping AUV (Caress)

Alex Worden

Steve Haddock

John Ryan

Ken Johnson (ISUS)

Mike Godin (AOSN, LRAUV)

Jim Bellingham (AOSN, LRAUV)

ALOHA

This page last changed on Jun 09, 2011 by kgomes.

The ALOHA group contacted MBARI about using SSDS and SIAM for a deployment that they were working towards for January of 2011. This page documents the work to get that installation up and running for them.

Meeting Notes

1. April 27, 2010

Installation Notes

These notes document what was done to configure the SSDS installation in HAWAII on the machine malino.soest.hawaii.edu. There are a couple of accounts that I used on malino during the installation. They are:

1. kgomes
2. jboss
   I also used the 'root' account on the mysql installation for the work I was doing on the database.

> **JBoss and Java were installed by the sysadmin**
>
> For this particular installation, the folks in Hawaii installed JBoss 5.1.0GA and Java 6.0.25 for me.

3. I first ssh'd into the malino and then started up the mysql client using

   ```
   mysql --user=root --password
   ```
4. I then created the two needed databases using:

   ```
   create database ssds_data;
   create database ssds_metadata;
   ```
5. I then created a user that will have all rights to the databases that SSDS will use to connect:

   ```
   create user 'ssdsadmin'@'localhost IDENTIFIED BY 'XXXXXXX'
   ```
6. I then granted all rights to the databases for the newly created user

   ```
   GRANT ALL ON ssds_data.* TO 'ssdsadmin'@'localhost';
   GRANT ALL ON ssds_metadata.* TO 'ssdsadmin'@'localhost';
   ```
7. I then checked out the source code on my malino in the jboss home directory /export/malino/jboss/ssds/build/shore-side-data-system-read-only
8. I downloaded the adobe flex sdk and unpacked it in /export/malino/jboss/flex_sdk
9. I downloaded apache ant and unzipped it in /export/malino/jboss/ant
10. I created a .login file in the jboss home directory and set two environment variables

    ```
    setenv JAVA_HOME /export/malino1/jdk
    setenv ANT_HOME /export/malino/jboss/ant
    ```
11. I removed the following applications by moving them to the jboss/backup directory in the jboss users's home directory
    a. admin-console.war
    b. jmx-console.war
12. I started up the JBoss on malino from the command line just to make sure it worked and all looked OK.
13. I then copied the custom.properties.template file in the ssds source directory to a file called custom.properties and edited it to match all the configurations for the deployment on malino
14. I then ran:

    ```
    ../../../ant/apache-ant-1.8.2/bin/ant deploy
    ```

    from the /export/malino/jboss/ssds/build/shore-side-data-system-read-only directory and it deployed all the files to the JBoss installation.
15. I then ran JBoss from the command line to make sure SSDS started up OK. (started up in 30s)

> **Firewall issue**

I had to use the local firefox and point it to localhost to get to SSDS because of firewall issues (exported XTerm display)

16. I used the ssds/newDeviceType.jsp page to create new device types for camera, CTD, ADP/currents, fluorometer, and inductive modem.
17. I used the ssds/newPerson.jsp page to create a contact for Fernando
18. I used the ssds/newDevice.jsp page to create all new device IDs for the ALOHA instruments and sent those IDs to the ALOHA team.
19. I edited the /export/malino1/jboss/bin/run.conf and added the following parameters to the JAVA_OPTS environment variable:

    ```
    -Djava.awt.headless=true -Duser.timezone=UTC
    ```

    and also changed the Xms and Xmx to 1024 and 2048 respectively

April 27, 2010

This page last changed on Apr 27, 2010 by kgomes.

Initial teleconference with ALOHA Observatory Group:

Attendees:

1. MBARI
   a. Kevin Gomes
   b. Duane Edgington
   c. Bob Herlien
2. ALOHA
   a. Fernando Santiago-Mandujano
   b. Paul Lethaby
   c. Jeffrey Schneider

Other contacts:

1. Pat Thompson (I.T. Support for SSDS machine)
2. MARS Team (ALOHA group was interested in contacting MARS team through Etch)

Notes:

1. Deployment on ALOHA Cabled Observatory
2. Early January 2011
3. Instruments:
   a. SBE37s (two of them)
   b. Array of thermistor SBE39s (10 of them inductively)
   c. ADCP (Sontek 250kHz), probably be getting an RDI deep water monitor
   d. BB2F
   e. Two hydrophones (probably not managed through SIAM and SSDS)
   f. One video camera (not defined), assume video servers (probably not managed through SIAM and SSDS).
4. They connect instruments to serial port terminal server (digi 4 port server).
5. The deployment will be at 5000m depth
6. They would like to use SSDS to do data management
7. Would like to use SIAM to communicate with sensors.
8. DataTurbine is of interest to the ALOHA guys and they would use it.
9. They do not have PUCKs on their instruments, we said no problem.
10. They are interested in Adaptive Sampling (automated).
    a. We felt this might be a good opportunity to get a constrained, real-world use case for adaptive sampling.
11. They were interested in documentation for SIAM and its integration with DataTurbine
12. Sample rates for instruments on average around once a minute (or a little less).
13. Bob (for SIAM) and Kevin (for SSDS) will be the points of contacts for MBARI<->ALOHA interactions.

Timeline:

More details to come from ALOHA group, but some timeline notes:

1. Ship time is January
2. They want test system in lab ASAP (they actually have some of it running currently)

Action Items:

1. Kevin will look back and try to get SSDS install going again on kainani.
2. ALOHA group will send Bob information about instruments (IP addresses, etc.)
3. Bob will look into getting SIAM running against their instruments.
4. ALOHA group will work on generating a timeline with milestones like code freeze, integration and test, etc.

NOTE: As long as the time is not huge, we can charge to TTOS.

An example use of Graphs - FOCE

This page last changed on Sep 30, 2009 by kgomes.

This page describes how you might configure a web application to consume the graphs that are generated by the SSDS. The way this example is done is by using the Eclipse Java EE Edition to create a web application that allows the user to author pages and deploy them to an application server like Tomcat.

Install Servlet Container

The first thing to do is install a servlet container where your web application will be deployed. This is most often Tomcat, but can also be something that utilizes Tomcat internally (like JBoss). For this example, we will download and use JBoss 4.2.2GA.

1. Download JBoss from here http://sourceforge.net/projects/jboss/files/JBoss/JBoss-4.2.2.GA/
2. Unpack the download to where you want to install it, open a command window, and change to the directory where you unpacked the download (where you installed it).
3. Start the JBoss server by running run(.sh for Unix and .bat for Windows). Assuming now errors fly by and you get the 'server started in ...' message at the end, Jboss installed successfully. You can use Cntl-C to stop the server.
4. Now download Eclipse JEE 3.5 from here http://eclipse.org/downloads/ and install it.
5. Startup Eclipse after you install it.
6. Go to the Workbench view so we can configure a server which will connect to your JBoss installation.
7. Click on 'File->New->Other...' and then select 'Server->Server'. Then 'Next>'
8. Then select 'JBoss->JBoss v4.2'. The default server host of 'localhost' and server name of 'JBoss v4.2 on localhost' is fine. Click on 'Next>'.
9. You can use the Default JRE and for the 'Application Server Directory' browse to the location where you unpacked JBoss.

Under Construction

September 30, 2009 this is still under construction

Data Simulator

This page last changed on Jan 05, 2010 by kgomes.

In order to test the various components in SSDS, a simulator was built to allow users to send messages as packets to the SSDS system for testing (and could be used to send messages to the production machine as well).

This page last changed on Feb 03, 2012 by kgomes.

Debugging quick look and contour wind stick plots

The quick look plots on the public Oasis data page (http://www.mbari.org/oasis/qc/index.html) are created by the SSDS-driven NetCDF processing that runs on elvis every 2 hours. To start with understanding the processing look at the crontab for the ssdsadmin account on elvis. There are some notes in the TROUBLESHOOT file in /u/ssdsadmin/dev/DPforSSDS/cimt to help with running the main scripts (DStoNetCDF.pl, combineTS.pl, combineMet.pl, combineAll.pl) for a specific deployment. All of the Perl code that builds the Ferret .jnl files which produce the plots are in the ssds_util.pl library. In addition all of the processing is reported to SSDS with DataProducers with the plots files generated recorded as Resources. One could search and walk through the processing provenance in SSDS to find the script that produced a plot.

A "side effect" of the main processing is the creation of the current_qcPlots.html web page produced for each mooring that is processed. This page has some short-cut links to the Ferret scripts (jnl files) that produce the plots. All of the pages linked in the Plots column ('full Deployment' and 'last 7 days') have a link at the top that points to the .jnl file that produced the plots on the page. The Ferret commands can be copy and pasted from the jnl page into a ferret session. I suggest running ferret on elvis and remoting the X-Display to your computer.

The "Last 30 day Wind Temperature contour" and "Last 30 day Wind Salinity contour" GIF images are created by a jnl file that is in the same directory as the images. Edit the URL to examine the contents or the directory and see the Ferret commands that produce the plot. It's helpful to copy the USE and SET REGION commands from the .jnl page into a ferret session and examine the data to debug what might be wrong in producing the plots.

A. Initial look at the data

Here's an example of doing this (with the SET REGION command edited to list just the last day's data) - the problem being analyzed is gappy subsurface data:

```
 > ferret
NOAA/PMEL TMAP
FERRET v6.62
Linux rh5 (gfortran) 2.6.18-164.11.1.el5 - 06/11/10
30-May-11 21:59

yes? USE "http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
OS_M1_20101027hourly_CMSTV.nc"
yes? SET REGION/T="30-May-2011 04:43":"31-May-2011 04:43"
yes? list sea_water_temperature_hr
VARIABLE : Sea Water Temperature (Celsius)
DATA SET : Hourly Gridded MBARI Mooring M1 Sea Water Temperature and Salinity Observations
FILENAME : OS_M1_20101027hourly_CMSTV.nc
FILEPATH : http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
SUBSET   : 11 by 24 points (DEPTH (m)-TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
1      10     20     40     60     80    100    150    200    250    300
1      2      3      4      5      6      7      8      9     10     11
30-MAY-2011 04:30 / 5145:
10.77   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 05:30 / 5146:
10.69   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 06:30 / 5147:
10.59   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 07:30 / 5148:
10.49   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 08:30 / 5149:
10.42   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 09:30 / 5150:
10.29   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
```

```
30-MAY-2011 10:30 / 5151:
10.23   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 11:30 / 5152:
10.33   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 12:30 / 5153:
10.35   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
30-MAY-2011 13:30 / 5154:  10.34   ....   ....   9.56   9.08   8.85   8.39   8.10   7.89   7.59
7.51
30-MAY-2011 14:30 / 5155:  10.36   ....   ....   9.28   8.92   8.70   8.46   8.12   7.90   7.60
7.51
30-MAY-2011 15:30 / 5156:  10.38   ....   ....   9.26   8.92   8.73   8.39   8.11   7.93   7.67
7.50
30-MAY-2011 16:30 / 5157:  10.45   ....   ....   9.38   9.04   8.81   8.38   8.11   7.93   7.68
7.49
30-MAY-2011 17:30 / 5158:  10.50   ....   ....   9.55   9.13   8.89   8.64   8.15   7.96   7.68
7.49
30-MAY-2011 18:30 / 5159:  10.76   ....   ....  10.16   9.14   8.90   8.71   8.26   7.97   7.67
7.49
30-MAY-2011 19:30 / 5160:  11.06   ....   ....  10.15   9.10   8.91   8.69   8.25   7.98   7.69
7.50
30-MAY-2011 20:30 / 5161:  11.33   ....   ....  10.18   9.07   8.90   8.61   8.21   8.01   7.69
7.49
30-MAY-2011 21:30 / 5162:  11.37   ....   ....   9.75   9.03   8.83   8.50   8.24   7.99   7.69
7.49
30-MAY-2011 22:30 / 5163:  11.10   ....   ....   9.85   9.04   8.83   8.51   8.27   7.99   7.69
7.50
30-MAY-2011 23:30 / 5164:
11.05   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
31-MAY-2011 00:30 / 5165:
10.90   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
31-MAY-2011 01:30 / 5166:
10.84   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
31-MAY-2011 02:30 / 5167:
10.59   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
31-MAY-2011 03:30 / 5168:
10.37   ....   ....   ....   ....   ....   ....   ....   ....   ....   ....
```

The many '....'s indicate missing data in this hourly gridded file. Let's look upstream in the data processing to see what the input data looks like. The .jnl file that created the 201010/OS_M1_20101027hourly_CMSTV.nc file simply USEd the TS data. Listing the data from the TS file shows the same gappy data as above. So let's look at the input data to the TS file. The jnl file link in the Data column in the *[TS:]* row. Listing the data from the individual input files shows that they apparently do not have the gaps that are shown int he gridded data set:

```
yes? USE "http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
m1_ctd0010_20101027_original.nc"
yes? list temperature
VARIABLE : Water Temperature (deg C)
DATA SET : Mooring M1 CTD data from MBARI instrument id 1491 at original sampling intervals
FILENAME : m1_ctd0010_20101027_original.nc
FILEPATH : http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
SUBSET   : 140 points (TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
DEPTH (m): 10
122W
1
30-MAY-2011 04:49:31 / 30019:  10.73
30-MAY-2011 04:59:30 / 30020:  10.72
30-MAY-2011 05:09:31 / 30021:  10.70
30-MAY-2011 05:19:29 / 30022:  10.70
30-MAY-2011 05:29:30 / 30023:  10.69
30-MAY-2011 05:39:31 / 30024:  10.69
(records skipped)
31-MAY-2011 02:59:30 / 30152:  10.12
```

```
31-MAY-2011 03:09:31 / 30153:  10.06
31-MAY-2011 03:19:30 / 30154:  10.08
31-MAY-2011 03:29:30 / 30155:  10.07
31-MAY-2011 03:39:29 / 30156:  10.06
31-MAY-2011 03:49:31 / 30157:  10.08
31-MAY-2011 03:59:30 / 30158:  10.05
```

B. Tips for debugging with Ferret

To figure out what is going wrong we'll need to execute more of the Ferret commands from the jnl file that creates the TS file and examine the data at each step. This is an interactive process that involves editing a temporary .jnl file, executing it in ferret with a "GO <jnl_file>" and analyzing the output. Here are some more tips for diagnosing problems:

1. Examine the production ferret output - this is the 'out' link on the current_qcPlots.html web page
2. Examine CVS for changes in the source code - the change log for combineTS.pl (the script that produces the jnl file link in the Data column in the *[TS:]* row) is http://moonjelly.shore.mbari.org/cgi-bin/viewvc.cgi/DPforSSDS/cimt/combineTS.pl?view=log
3. Examine the data and plots for the individual microcats on the current_qcPlots.html web page
4. Read the excellent online Ferret documentation, starting with the "Thinking like a Ferret" page: http://ferret.pmel.noaa.gov/Ferret/documentation/users-guide/introduction/GETTING-STARTED

To follow through with this example of gappy data I took these steps:

1. Copied the OS_MBARI-M1_20101027_R_TS.jnl file from it's production location (/mbari/ssdsdata/deployments/m1/201010) to /tmp
2. Edited the file to process and save only the 1m and 10m data and save the date to a temporary netcdf file in /tmp
3. Executed the temporary truncated file from a Ferret session

```
yes? go "/tmp/OS_MBARI-M1_20101027_R_TS.jnl"
! Description: Produce netCDF file of all Temperature and Salinity measurements from a mooring
!              Pull data from original instrument netCDF files and grid onto a common grid.
!              Automatically generated by ./combineTS.pl on Tue May 31 09:37:27 2011.
!              For information on Ferret see http://ferret.wrc.noaa.gov/Ferret/.
!

<snip>

SAVE/FILE="/tmp/OS_MBARI-M1_20101027_R_TS.nc"/APPEND/KLIMITS=1:11/LLIMITS=1:`PSAL,return=lend`/Z=1/
T="27-Oct-2010 21:00:00":"31-May-2011 16:00:00" PSAL,
PSAL_QC, TEMP, TEMP_QC, TIME_QC, POSITION_QC, DEPTH_QC
!-> LIST/FORMAT=CDF/FILE="/tmp/OS_MBARI-M1_20101027_R_TS.nc"/APPEND/KLIMITS=1:11/LLIMITS=1:5182/
Z=1/T="27-Oct-2010 21:00:00":"31-May-2011 16:00:00" PSAL,
PSAL_QC, TEMP, TEMP_QC, TIME_QC, POSITION_QC, DEPTH_QC
LISTing to file /tmp/OS_MBARI-M1_20101027_R_TS.nc

<snip>

SAVE/FILE="/tmp/OS_MBARI-M1_20101027_R_TS.nc"/APPEND/Z=10/T="27-Oct-2010 20:00:00":"31-May-2011
15:00:00" PSAL, PSAL_QC, TEMP, TEMP_QC, TIME_QC, POSITION_QC, DEPTH_QC
LISTing to file /tmp/OS_MBARI-M1_20101027_R_TS.nc
**TMAP ERR: error in line definition
disordered output coordinate value:  22215.      Axis: TIME
LIST/FORMAT=CDF/FILE="/tmp/OS_MBARI-M1_20101027_R_TS.nc"/APPEND/Z=10/T="27-Oct-2010 20:00:00":"31-
May-2011 15:00:00" PSAL, PSAL_QC, TEMP, TEMP_QC, TIME_QC, POSITION_QC, DEPTH_QC
Command file, command group, or REPEAT execution aborted
yes?
```

This looks like a good clue. The 10m data are not being written to the output file and we are getting this obscure "disordered output coordinate value" error from Ferret. We need to get to the bottom of this. Here are some more tips on how to proceed:

1. Use the Ferret mail list archive () or Google to search for the meaning of this error message

2. Join the Ferret mail list and post your question. Solutions are typically provided within a day.
3. Examine the input data and variables using ncdump(1) or Ferret LIST and SHOW commands

Here are some Ferret commands to examine the first 3 temperature values from the 1m and 10m input CTD data and the first 3 times of the output axis:

```
yes? list/l=1:3/d=1 temperature
VARIABLE : Water Temperature (deg C)
DATA SET : Mooring M1 CTD data from MBARI instrument id 1338 at original sampling intervals
FILENAME : m1_ctd0001_20101027_original.nc
FILEPATH : http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
SUBSET   : 3 points (TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
DEPTH (m): 1
122W
1
27-OCT-2010 21:08:43 / 1:  14.01
27-OCT-2010 21:22:27 / 2:  14.10
27-OCT-2010 21:28:23 / 3:  14.10
yes? list/l=1:3/d=2 temperature
VARIABLE : Water Temperature (deg C)
DATA SET : Mooring M1 CTD data from MBARI instrument id 1491 at original sampling intervals
FILENAME : m1_ctd0010_20101027_original.nc
FILEPATH : http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
SUBSET   : 3 points (TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
DEPTH (m): 10
122W
1
27-OCT-2010 20:11:17 / 1:  13.38
27-OCT-2010 20:21:16 / 2:  13.59
27-OCT-2010 20:31:16 / 3:  13.45
yes? show/l=1:3 axis time
name       axis              # pts   start                end
TIME      TIME              5181 r   27-OCT-2010 20:30    31-MAY-2011 16:30
T0 = 01-JAN-1950 00:00:00
Axis span (to cell edges) = 215.875

L     T                    TBOX      TBOXLO                TSTEP (DAYS)
1>  27-OCT-2010 20:30:00  0.0416667  27-OCT-2010 20:00:00    22214.85
2>  27-OCT-2010 21:30:00  0.0416667  27-OCT-2010 21:00:00    22214.9
3>  27-OCT-2010 22:30:00  0.0416667  27-OCT-2010 22:00:00    22214.94
```

C. Fixing the start times for all the child instrument deployments

The problem seems to be that the 10m data begin at 20:11:17 before the 1m data at 21:08:43. The first write of the 1m data to the output netCDF file gives the file its shape with the KLIMITS= and LLIMITS= options. Writing the 10m data, which begins in before the bounds that were defined violates Ferret's axis ordering logic. To test this hypothesis edit the SAVE statement in the temporary .jnl file to make the start time for the 10m data 21:00, execute it and then examine the output file:

```
yes? go "/tmp/OS_MBARI-M1_20101027_R_TS.jnl"

<snip>

SAVE/FILE="/tmp/OS_MBARI-M1_20101027_R_TS.nc"/APPEND/Z=10/T="27-Oct-2010 21:00:00":"31-May-2011
15:00:00" PSAL, PSAL_QC, TEMP, TEMP_QC, TIME_QC, POSITION_QC, DEPTH_QC
LISTing to file /tmp/OS_MBARI-M1_20101027_R_TS.nc

(Another ferret session)
> ferret
NOAA/PMEL TMAP
FERRET v6.62
```

```
Linux rh5 (gfortran) 2.6.18-164.11.1.el5 - 06/11/10
31-May-11 12:19

yes? use "/tmp/OS_MBARI-M1_20101027_R_TS.nc"
yes? list/l=1:10 temp
VARIABLE : Hourly sea_water_temperature (celsius)
FILENAME : OS_MBARI-M1_20101027_R_TS.nc
FILEPATH : /tmp/
SUBSET   : 11 by 10 points (DEPTH (m)-TIME)
LONGITUDE: 122W(-122)
LATITUDE : 36.8N
1      10     20     40     60     80    100    150    200    250    300
1      2      3      4      5      6      7      8      9      10     11
27-OCT-2010 21:30 /  1:  14.09
13.60   ....   ....   ....   ....   ....   ....   ....   ....   ....
27-OCT-2010 22:30 /  2:  14.17
13.62   ....   ....   ....   ....   ....   ....   ....   ....   ....
27-OCT-2010 23:30 /  3:  14.34
13.63   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 00:30 /  4:  14.28
13.69   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 01:30 /  5:  14.22
13.72   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 02:30 /  6:  14.21
13.78   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 03:30 /  7:  14.21
13.76   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 04:30 /  8:  14.09
13.79   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 05:30 /  9:  14.01
13.77   ....   ....   ....   ....   ....   ....   ....   ....   ....
28-OCT-2010 06:30 / 10:  13.89
13.76   ....   ....   ....   ....   ....   ....   ....   ....   ....
yes?
```

O.K.! That looks good. The 10m data got written. Now to fix it for production. This deployment was a special case where the 1m data start after the 10m data. Typically, it's most convenient to have all of the microcats have the same deployment start time. This makes the follow on data processing much simpler.

All of the deployment start times in SSDS_METADTA for the M1 201010 deployment were adjusted to be the same as the start time for the Mooring deployment. Performing this step is now part of the standard operating procedure for performing mooring turns. Once SSDS_METADATA has been updated the individual instrument netCDF files must be created by DStoNetCDF.pl before combineTS.pl in run.

D. Examining the gridding method

There are still some problems with gappy data at 40m and below in the _TS file. To debug these we'll make another copy of the OS_MBARI-M1_20101027_R_TS.jnl to /tmp and edit it to go through the processing of the 40m data saving the data to a netCDF file in /tmp so as not to disturb the production processing. We also comment out all of the "CANCEL DATA 1" statements so that we can examine data from the files. This file is executed and we see no errors. We need to examine the input data and the Ferret variables that are constructed for the gridding. Here are the commands from the .jnl that perform the the gridding for the 40m data:

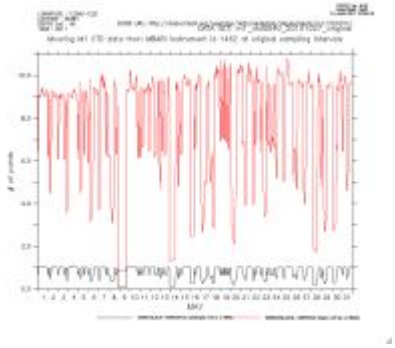
```
!
! Remove temperature outliers before gridding
!
LET Temperature_QFLAG = IF Temperature GT 2 AND Temperature LT 20 THEN 1 ELSE (-99999)
SET VAR/BAD=-99999 Temperature_QFLAG
LET Temperature_QC = Temperature_QFLAG * Temperature
!
! Compute my own mean of the data, making sure to assign missing values in the gaps in the Gap FLAG
! Allow at least 1 data point in each destination cell - there are usually 6 for 10 minute data
!
LET Temperature_GOOD = Temperature_QC[gt=TIME@SUM] / Temperature_QC[gt=TIME@NGD]
```

```
LET Temperature_GFLAG = IF Temperature[gt=TIME@NGD,gz=DEPTH@XACT,gy=LATITUDE,gx=LONGITUDE] LT 1
THEN (-99999) ELSE 1
SET VAR/BAD=-99999 Temperature_GFLAG
LET TEMP = Temperature_GFLAG * Temperature_GOOD[gz=DEPTH@XACT,gy=LATITUDE,gx=LONGITUDE]
```

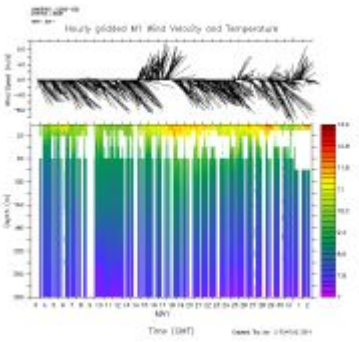
Here are some sample Ferret commands for plotting these data from the session that just executed the OS_MBARI-M1_20101027_R_TS.jnl script:

```
yes? go "OS_MBARI-M1_20101027_R_TS.jnl"
<snip>
yes? set region/t=1-may-2011:1-jun-2011
yes? plot Temperature_QC
yes? plot/ov/symbol=1 temperature
yes? set win 2
yes? plot Temperature_QC[gt=TIME@NGD]
yes? plot TEMPERATURE_QC[GT=TIME@SUM]
```

These last two plot commands indicate the source of the problem of the gappy data from the inductive modem microcats. Another big clue is the comment in the .jnl file referring to the 10-minute input data and having at least 6 good values within each gridding cell. As can be seen in the plot we often have less than 1 for the @NGD value for the 40m data. These data are hourly and we should be using a different gridding algorithm for them.



The October 2010 M1 deployment was the first one where all the inductive modem connected CTDs were logged as separate instruments, and the first deployment where the serial microcat processing software was used to process the inductive modem microcat data. The fix for this problem appears to be to do a different gridding for the hourly data from 40m and below. This change was implemented in the code on 2 June 2011 using the Ferret same gridding transform (@MAX) that was previously used for the inductive modem data processing. Here is the after figure:



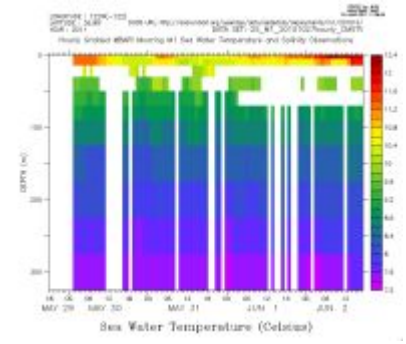
The subsurface data looks a lot better, but there are still some gaps. If the gaps extend across the whole water column encompassing both the inductive modem and serial microcats then perhaps the gap is due to an interruption in data processing through oasisToSSDS. This can be remedied by reprocessing the telemetered data by following the README instructions in /u/ssdsadmin/dev/DPforSSDS/oasis/scripts on elvis.

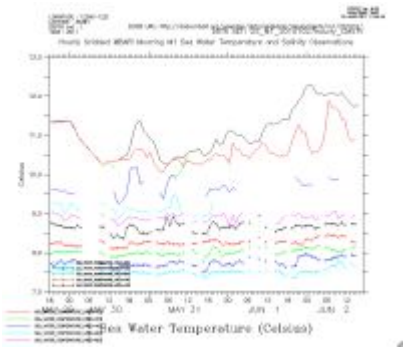
E. Testing the results of the gridding

It is worthwhile to do another check on the quality of the gridding that is now being performed. The following Ferret exercice will show the input data compared to the gridded data.

```
USE "http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
OS_M1_20101027hourly_CMSTV.nc"
SET REGION/T="29-May-2011 18:13":"02-Jun-2011 18:13"
shade SEA_WATER_TEMPERATURE_HR
set win 2
plot/k=1/vlimits=7:13 SEA_WATER_TEMPERATURE_HR
repeat/k=2:11 plot/ov SEA_WATER_TEMPERATURE_HR
```

Produces these images showing missing data, especially from the 20m and 40m microcats:



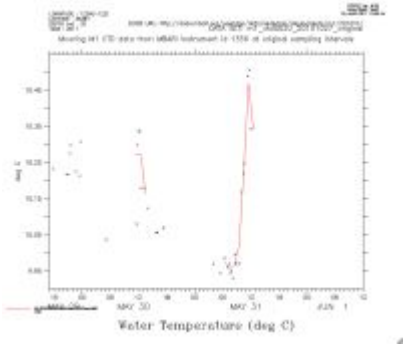


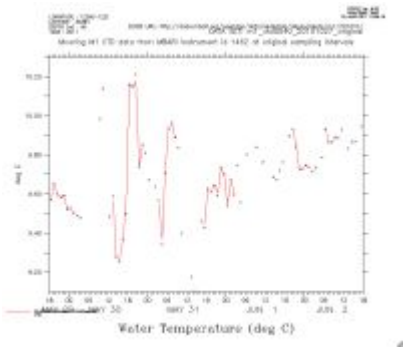
Let's compare the original instruments date from 20m and 40m to the gridded product, using the same Ferret session from above:"

```
use "http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
m1_ctd0020_20101027_original.nc"
plot/symbol=1 temperature[d=2]
plot/ov SEA_WATER_TEMPERATURE_HR[k=3,d=1]

use "http://dods.mbari.org/opendap/data/ssdsdata/deployments/m1/201010/
m1_ctd0040_20101027_original.nc"
plot/symbol=1 temperature[d=3]
plot/ov SEA_WATER_TEMPERATURE_HR[k=4,d=1]
```

Gives:





We obviously still have some gridding issues...

F. Rinse and repeat

Now is the time to iterate using our method of starting with the .jnl file that aggregates and grids the data to the OceanSITES _TS.nc file going back to step D.

How to Configure Graphs

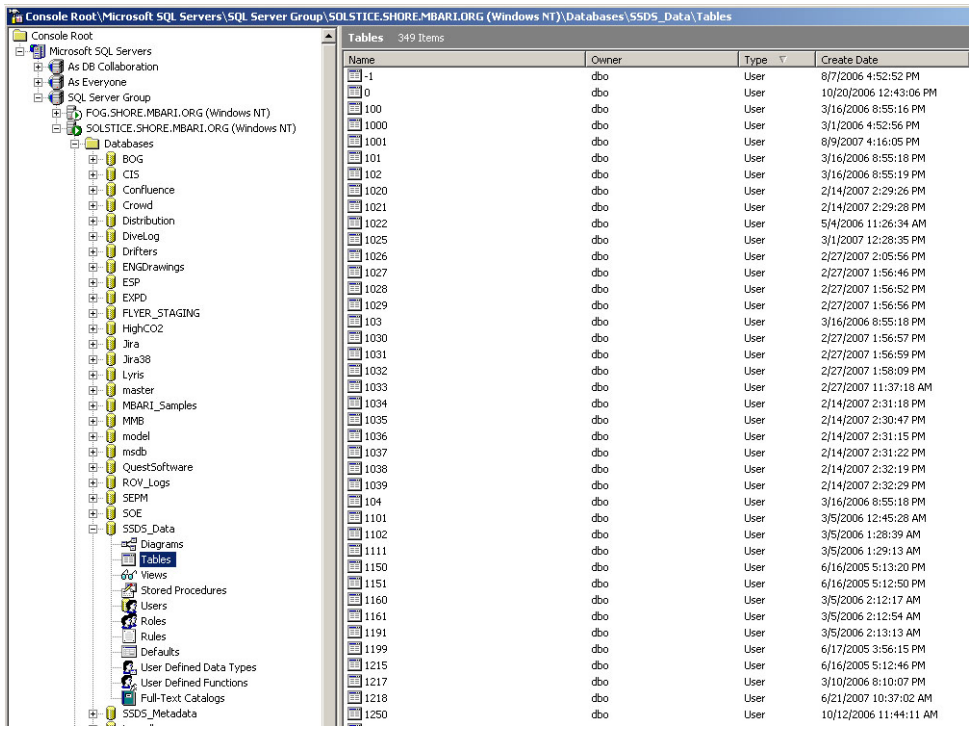
This page last changed on Aug 11, 2009 by kgomes.

This page documents how to configure SSDS to generate batch plots of various data that is tracked in the Shore Side Data System. The batch process runs every hour so keep that in mind as you configure your plots. It is on our plate to develop a nice web front end to this configuration, but for now, you will have to brute force it through the database itself. The information you will need to know ahead of time is:

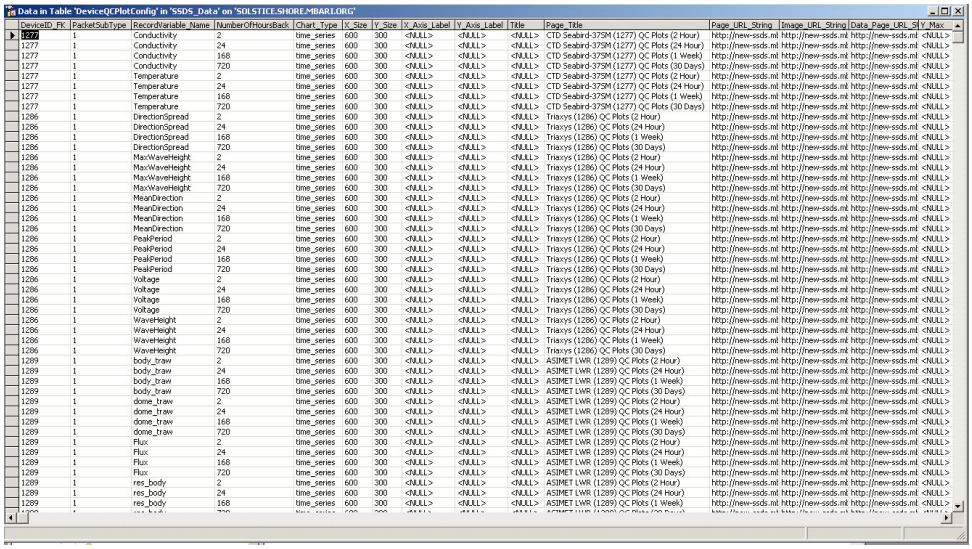
1. The SSDS ID of the instrument you want to create plots for.
2. The variable names (exactly!) that you want plotted
3. The length of times you want the plots to go back (this software gives you the option to choose the number of hours back and uses that to plot the data).
4. The titles you want listed on your plot

Now, in order to get to the table in the database, you will need to be on the MBARI network and have access to the Enterprise Manager software from Microsoft. That is what is used to connect to the database where this configuration is kept. You will also need an account that you can use to connect to the database with that has permissions to edit the table (check with the SSDS guys for this). So, first start up Enterprise Manager in windows. Once you have started it, there should be a group called 'SQL Server Group'. If you right click on that, you can register a connection to a database server by selecting "New SQL Server Registration". A wizard will walk you through the registration process and you want to connect to the server solstice.shore.mbari.org using the name and password from the SSDS guys.

Once you have registered the server, you can open the 'plus' sign next to it to dive down to the database of interest: SOLSTICE.SHORE.MBARI.ORG->Databases->SSDS_Data->Tables and you should see something like this:



You are looking for the **DeviceQCPlotConfig** table and if you scroll down and right-click on it, then choose 'Open Table->Return all Rows' you will see the listing of all the configurations for creating plots (Figure 2).



Here is a list of the columns and what they mean:

- **DeviceID_FK**: This is the SSDS ID of the instrument whose data you want to plot.
- **PacketSubType**: This is the RecordType that contains the variable you want to plot. Some instruments have more than one type of data they produce. In order to differentiate those, all device data descriptions must specify a RecordType (usually it is a 1, but not always). You can consult the SSDS web interface (http://new-ssds.mbari.org) or the device XML to find what the RecordTypes are.
- **RecordVariable_Name**: This is the short name of the variable you want to plot.
- **NumberOfHoursBack**: This tells the software how far back (in hours) you want to go to grab the data to plot. It will grab data back that many hours until the current time and create a plot of that data.
- **Chart_Type**:
- **X_Size**:
- **Y_Size**:
- **X_Axis_Label**:
- **Y_Axis_Label**:
- **Title**:
- **Page_Title**:
- **Page_URL_String**:
- **Image_URL_String**:
- **Data_Page_URL_String**:
- **Y_Max**:
- **Y_Min**:
- **Y_Axis_One_Or_Two**:
- **Use_Long_Variable_Name**:
- **Gps_Show_Anchor**:
- **Gps_Anchor_Latitude**:
- **Gps_Show_Watch_Circle**:
- **Gps_Watch_Circle_Diameter_Km**:
- **Gps_Scale_Chart_To_Fit_Data**:
- **GraphOn**:

Ingest Architecture

This page last changed on Mar 18, 2011 by kgomes.

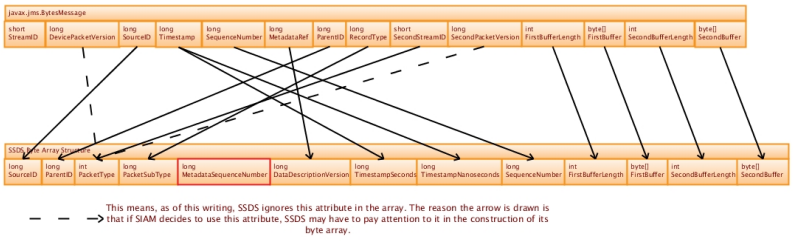
This doc has been moved

Most of the documentation for SSDS should be on the Google code site now and this page is here:
http://code.google.com/p/shore-side-data-system/wiki/TransmogrifyAndIngest

Name	Type	Description
StreamID	java.lang.short	This basically states that the bytes are coming from a SIAM ExportablePacket class. SIAM uses constants defined in the org.mbari.siam.distributed.Exportable.java class to enumerate things like this and the short value for this is always 0x0100. SSDS Doesn't really care so we essentially ignore it.
DevicePacketVersion	java.lang.long	This is the "serialVersionUID" on the class that was used to export the bytes. It is the version of SIAM class that generated the byte array. SSDS Does not really care and as of this writing, it is always 0.
SourceID	java.lang.long	The ID of the device that the message was generated by.
Timestamp	java.lang.long	Epoch milliseconds (number of elapsed milliseconds since 1/1/1970 00:00:00) that the packet was generated by the device
SequenceNumber	java.lang.long	A number that is supposed to show the order of generation of packets from the device
MetadataRef	java.lang.long	This is the sequence number of the packet that contains the metadata that describes the information in this packet
ParentID	java.lang.long	The ID of the device that the generated device was attached to when it generated the packet
RecordType	java.lang.long	The type of record that this packet contains: • 0 = MetadataPacket • 1 = Non-MetadataPacket (Data and other)
SecondStreamID	java.lang.short	This defines the type of DevicePacket that was used to

		construct the byte array. The values are as follows: 1. MetadataPacket = 0x101 2. SensorDataPacket = 0x102 3. DeviceMessagePacket = 0x103 4. SummaryPacket = 0x102 (same as SensorDataPacket)
SecondPacketVersion	java.lang.long	This is the "serialVersionUID" on the class that was used to export the bytes. It is the version of SIAM class that generated the byte array. As of this writing, it is the same as the DevicePacketVersion. Since currently it is always 0, SSDS ignores it.
FirstBufferLength	java.lang.int	This is the length of the array that holds the bytes of the first buffer
FirstBuffer	java.lang.byte []	This is the bytes array that represents the first buffer
SecondBufferLength	java.lang.int	This is the length of the array that holds the bytes of the second buffer.
SecondBuffer	java.lang.byte []	This is the array that holds the bytes of the second buffer.

Now, in order to handle both types of inputs in Transmogrify (DevicePackets and BytesMessage structure), Transmogrify would take both and convert to a common format that would contain the information to cover both types of messages. Since the BytesMessage structure encompasses all the information in the DevicePacket, we simply used that byte structure and in Transmogrify, a DevicePacket is converted to a SSDSDevicePacket which is then converted to the same BytesMessage structure using the SSDSDevicePacket.convertToPublishableByteArray method. So at the end of the Transmogrify process, we have on byte array that is in the form of the diagram above that will then be used to publish a message to the next component which is Ingest. Transmogrify takes the SIAM byte array structure and converts it to the SSDS native byte array structure:



Notes on the conversion:

1. The DevicePacketVersion, SecondStreamID, and SecondPacketVersion are used to determine the correct packetType (although, right now, DevicePacketVersion and SecondPacketVersion are ignored).
    - For MetadataPackets, the packetType is 1.

    - For SensorDataPackets, the packetType is 0 (SummaryPackets come across as SensorDataPacket and are differentiated by their recordType).
    - For DeviceMessagePackets, the packetType is 4.
2. If the incoming packet is a MetadataPacket, the packetSubType is set to 0. Otherwise, it is set to the RecordType field.
3. The RecordType is set to zero if the packet is a MetadataPacket and set equal to the RecordType from SIAM if not a MetadataPacket.
4. The MetadataSequenceNumber is calculated depending on the device, it's parent, and the XML that is in it's payload. There is a component called the SIAMMetadataTracker that keeps track of this information and looks for real XML changes which is what should fire a change in metadata.
5. The buffers are swapped if it is a MetadataPacket. It always seemed to logical to do it that way.
6. This timestamp (epoch milliseconds) is split into seconds and nanoseconds.

Message Size Limitation!

Please note that because byte arrays are limited to 32 bit sizes, the largest payload of a message that can be converted by SSDS is 2GB. While this does not seem like a major restriction, it can be hit if somebody is using straight JMS messaging (or other) and makes a payload bigger than 2GB. SSDS will just ignore such a message.

Ingest Packet Structure

So now we have all messages coming into Ingest in a format that SSDS is expecting (i.e. that matches the SSDS view of the world). For the diagram in the previous section, the attributes in the SSDS Bytes Array are:

Attribute	Type	Description
sourceID	java.lang.long	This is what is known as the SSDS ID for the device (i.e. DeviceID) that actually generated the packet of information.
parentID	java.lang.long	This is the SSDS ID for the parent that the device was connected to when it sent the packet. If the ID is zero (0), then the generating device was not connected to a parent.
packetType	java.lang.int	This is the "Type" of packet that is being sent. It is basically an enumerated list the with following context: 0 = Data Packet 1 = Metadata Packet 2 = 3 = 4 = Device Message Packet
packetSubType	java.lang.long	This is the equivalent of the "recordType" listed in the Transmogrify component. It is used to provide the hook to tell the client applications what type of record this is that was sent. It really only has meaning in the context of data packets as a device can often send data packets of different formats. This basically tells the

		application which record form is being sent in this packet.
metadataSequenceNumber	java.lang.long	Also referred to as dataDescriptionID
dataDescriptionVersion	java.lang.long	
timestampSeconds	java.lang.long)	
timestampNanoseconds	java.lang.long	
sequenceNumber	java.lang.long)	
bufferLen	java.lang.int	
bufferBytes	java.lang.byte[bufferLen]	
bufferTwoLen	java.lang.int	
bufferTwoBytes	java.lang.byte[bufferTwoLen]	

The Ingest Message Driven Bean (MDB) then takes that byte array and using a PacketOutput class that corresponds to the correct source ID, metadataSequenceNumber, packetSubType, and parentID, it writes the packet to disk. It then uses a PacketSQLOutput to write that same packet to a table in the database.

Packet Translations

So, through all this, there are basically four representations of data packets in the SSDS ecosystem:

1. SIAM Device Packet (and its sub classes MetadataPacket, SensorDataPacket, DeviceMessagePacket)
2. SSDSDevicePacket (and its sub class SSDSGeoLocatedDevicePacket)
3. SIAM Byte array (from Exportable class)
4. SSDS Byte array

Here is a diagram of these various forms of data

And the translation rules (some of these may seem very strange for legacy reasons).

DevicePacket to SSDSDevicePacket

This translation is done in the constructor of SSDSDevicePacket which takes in a DevicePacket

DevicePacket	Translation Rule	SSDSDevicePacket
sourceID	direct copy	sourceID
systemTime	direct copy	systemTime
sequenceNo	direct copy	sequenceNo
metadataRef	direct copy: 1. metadataRef->metadataRef 2. metadataRef->metadataSequenceNumber 3. metadataRef->dataDescriptionID	1. metadataRef 2. metadataSequenceNumber 3. dataDescriptionID
parentId	direct copy: 1. parentId->parentId	1. parentId 2. platformID

	2. parentId->platformID	
recordType	1. MetadataPacket: recordType to 0 2. Other: direct copy	recordType
	1. If MetadataPacket, packetType = 0 2. If SensorDataPacket, packetType = 1 3. If DeviceMessagePacket, packetType = 2	packetType
firstBufferLength	ignored	
firstBuffer	First buffer depends on which type of packet 1. If MetadataPacket, copy "bytes" buffer 2. If SensorDataPacket, copy "dataBuffer" 3. If DeviceMessagePacket, copy "message"	firstBuffer
secondBufferLength	ignored	
secondBuffer	Only exists if MetadataPacket and will copy over "cause" buffer	secondBuffer

DevicePacket to SIAM Byte Array

This is done by the SIAM Exportable Packet class

DevicePacket	Translation Rule	SIAM Byte Array
	This is a static value that is set to indicate the byte array is a DevicePacket and is set to 0x0100	EX_DEVICEPACKET
	This is just the serial version UID of the class which is always 0	serialVersionUID
sourceID	direct copy	sourceID
systemTime	direct copy	systemTime
sequenceNo	direct copy	sequenceNo
metadataRef	direct copy	metadataRef
parentId	direct copy	parentId
recordType	direct copy	recordType
	This is set based on what type of packet: 1. If MetadataPacket, set to 0x0101	EX_XXXXXXPACKET

	2. If SensorDataPacket, set to 0x0102 3. If DeviceMessagePacket, set to 0x0103	
	This is just the serial version UID of the class which is always 0	serialVersionUID
firstBufferLength	direct copy (see note for first buffer)	firstBufferLength
firstBuffer	This depends on the type of packet: 1. If MetadataPacket, "cause" bytes are copied over 2. If SensorDataPacket, "dataMessage" bytes are copied over 3. If DeviceMessagePacket, "message" bytes are copied over	firstBuffer
secondBufferLength	only valid with MetadataPacket, but is copied directly over	secondBufferLength
secondBuffer	only valid with MetadataPacket, and the "buffer" bytes are copied over	secondBuffer

SSDSDevicePacket to SIAM Byte Array

Originally done in TransmogrifyMDB by calling SSDSDevicePacket.convertToPublishableVersion3ByteArray before passing byte array to method to translate to SSDS format and then send to ingest

SSDSDevicePacket	Translation Rule	SIAM Byte Array
	This is a static value that is set to indicate the byte array is a DevicePacket and is set to 0x0100	EX_DEVICEPACKET
	This is just the serial version UID of the class which is always 0	serialVersionUID
sourceID	direct copy	sourceID
systemTime	direct copy	systemTime
sequenceNo	direct copy	sequenceNo
metadataSequenceNumber	direct copy	metadataRef
parentID	direct copy	parentID
recordType	1. If packetType = 0, set recordType = 0 2. If packetType = 1, set recordType = recordType	recordType

	3. If packetType = 2, set recordType = recordType	
	This is set based on what type of packet: 1. If packetType = 0, set to 0x0101 2. If packetType = 1, set to 0x0102 3. If packetType = 2, set to 0x0103	EX_XXXXXXPACKET
	This is just the serial version UID of the class which is always 0	serialVersionUID
	This depends on the type of packet: 1. If packetType = 0, set to length of "otherBuffer" 2. If packetType = 1, set to length of "dataBuffer" 3. If packetType = 2, set to length of "dataBuffer"	firstBufferLength
other/dataBuffer	This depends on the type of packet: 1. If packetType = 0, set to bytes from "otherBuffer" 2. If packetType = 1, set to bytes from "dataBuffer" 3. If packetType = 2, set to bytes from "dataBuffer"	firstBuffer
	This only exists if it is packetType = 0 then it is set to the length of the "dataBuffer"	secondBufferLength
dataBuffer	This only exists if it is packetType = 0 then it is set to the byte from the "dataBuffer"	secondBuffer

SSDSDevicePacket to SSDS Byte Array

Originally done in SSDSDevicePacket.convertToVersion3ByteArray

SSDSDevicePacket	Translation Rule	SSDS Byte Array
sourceID	direct copy	sourceID
systemTime	ignored during the translation directly, but used through getter methods for seconds and nanoseconds	
timestampSeconds	direct copy (note that this is *sort of* a direct copy, there are getter methods on SSDSDevicePacket that convert the systemTime to	timestampSeconds

	seconds and nanoseconds when called).	
timestampNanoseconds	direct copy (note that this is *sort of* a direct copy, there are getter methods on SSDSDevicePacket that convert the systemTime to seconds and nanoseconds when called).	timestampNanoseconds
sequenceNo	direct copy	sequenceNumber
metadataRef	ignored	
parentID	ignored	
recordType	If packetType = 0, set packetSubType to 0, otherwise set to recordType	packetSubType
packetType	Depends on packetType: 1. If packetType = 0, set to 1 2. If packetType = 1, set to 0 3. If packetType = 2, set to 4	packetType
metadataSequenceNumber	direct copy	metadataSequenceNumber
dataDescriptionVersion	direct copy	dataDescriptionVersion
platformID	direct copy	parentID
	copy length of dataBuffer	firstBufferLength
dataBuffer	direct copy	firstBuffer
	copy length of otherBuffer	secondBufferLength
otherBuffer	direct copy	secondBuffer

SIAM Byte Array to SSDS Byte Array

Originally done in TransmogrifyMDB in checkAndPublishBytes method

SIAM Byte Array	Translation Rules	SSDS Byte Array
EX_DEVICEPACKET	ignored	
serialVersionUID	ignored	
sourceID	direct copy	sourceID
	Depending on EX_XXXXXXXPACKET: 1. If MetadataPacket, set packetType to 1 2. If SensorDataPacket, set packetType to 0 3. If DeviceMessagePacket, set packetType to 4	packetType

	This was set using the SIAMMetadataTracker that tried to keep track of real version numbers based on XML in payload	metadataSequencNumber
systemTime	Split into timestampSeconds and timestampNanoseconds	1. timestampSeconds 2. timestampNanoSeconds
sequenceNo	direct copy	sequenceNumber
metadataRef	direct copy	dataDescriptionVersion
parentId	direct copy	parentID
recordType	If MetadataPacket (determined from EX_XXXXXXXPACKET), recordType set to 0, otherwise set to recordType	packetSubType
EX_XXXXXXXPACKET	ignored in storage, but used in logic	
serialVersionUID	ignored	
first/secondBufferLength	If MetadataPacket (determined from EX_XXXXXXXPACKET), firstBufferLength is set to secondBufferLength so we can flip the "cause" and "buffer" bytes because it just made more sense since the cause was rarely populated. Otherwise set to firstBufferLength	firstBufferLength
first/secondBuffer	If MetadataPacket (determined from EX_XXXXXXXPACKET), firstBuffer is set to secondBuffer so we can flip the "cause" and "buffer" bytes because it just made more sense since the cause was rarely populated. Otherwise set to firstBuffer	firstBuffer
firstBufferLength	If MetadataPacket (determined from EX_XXXXXXXPACKET), secondBufferLength is set to firstBufferLength to flip "cause" and "buffer" bytes	secondBufferLength
firstBuffer	If MetadataPacket (determined from EX_XXXXXXXPACKET), secondBuffer is set to firstBuffer to flip "cause" and "buffer" bytes	secondBuffer

Exploration of Upgrade of Ingest/Transmogrify to AMQP

In an effort to allow non-Java clients to send data to SSDS in the form of messages and to upgrade the messaging system to a technology that is more scalable and higher performance, an investigation of AMQP implementations was done.

1. Qpid Exploration

Qpid Exploration

This page last changed on Oct 07, 2009 by kgomes.

For the Qpid exploration, I setup a Red Hat Enterprise Linux machine. I did all the following as root:

1. Downloaded the Qpid tar.gz from http://qpid.apache.org/download.html (http://www.apache.org/dist/qpid/0.5/qpid-0.5.tar.gz)
2. Unzipped the file.

   ```
   gunzip qpid-0.5.tar.gz
   ```
3. Untarred the file.

   ```
   tar -xvf qpid-0.5.tar
   ```
4. I decided to run the C++ broker and so I needed to build it.
5. Before I could do that though, I had to have all the right packages to build, so I ran:

   ```
   yum install boost-devel e2fsprogs-devel pkgconfig gcc-c++ make autoconf automake ruby libtool
   help2man doxygen graphviz
   ```

   of which the help2man and graphviz were not available.
6. To get help2man, I downloaded the tar.gz from here: http://ftp.gnu.org/gnu/help2man/help2man-1.36.4.tar.gz
7. Unzipped the file

   ```
   gunzip help2man-1.36.4.tar.gz
   ```
8. Untarred it

   ```
   tar -xvf help2man-1.36.4.tar
   ```
9. I then cd'd into that directory and ran

   ```
   ./configure --prefix=/root/Qpid/qpid-tools
   ```

   which then died with:

   ```
   configure: error: perl module Locale::gettext required
   ```
10. So, I installed the gettext-1.05.tar.gz perl module by downloading it, unzipping, unpacking, cd'ing into gettext-1.05 and running:

    ```
    perl Makefile.PL
    make
    make test
    make install
    ```
11. Went back to the help2man directory and tried it again (it succeeded this time)
12. Now run make

    ```
    make install
    ```

    which seemed to work OK.
13. Now for graphviz, I went to http://www.graphviz.org and downloaded the .rmp file and let RHEL open it with the package installer automatically.
14. I clicked on 'Apply' and installed it OK.
15. Because I wanted to explore using the clustering, I installed openais using yum by executing

    ```
    yum install openais
    ```
16. In order to get the XML exchange working, I need to install xerces by going to http://xerces.apache.org/xerces-c/ and downloading the latest version for 64 bit linux (tar.gz).
17. I unzipped it, untarred it
18. I then wanted to install xqilla and went to http://sourceforge.net/projects/xqilla/files/ to get the source.

Installation and Development

This page last changed on Mar 10, 2010 by kgomes.

Installation Instructions and Development Setup for the Shore Side Data System

Although these instructions may seem VERY long, they cover a lot of ground and with much detail. The idea was to make this as detailed as possible to make it exceptionally clear every step of the way. Some topics are somewhat lengthy to setup (like SSL), but are, in fact, very necessary for various reasons (security for the SSL case). These instructions were performed on a Apple OS X installation, but should apply to most Unix variants including Linux and OS X. See Windows Installation for an example done for USC on a Windows box.

So, without further ado, let's get to it!

1. If your platform does not already have Java installed, install it. You will need the SDK for Java 1.5+ (not just the JRE) in order to build and run the Shore Side Data System. For the OS X installation, it was already part of the OS, but if not, you will most likely retrieve it from http://java.sun.com. (The details of a Java installation are not listed here).
2. Install Ant which can be retrieved from http://ant.apache.org/. Ant 1.7.0 was used during the development of these instructions. You will want to make sure that the bin directory of the Ant installation is in your path so that you can run 'ant' from any command line location.
3. Install an instance of an Apache 2 web server. The scope of this installation is outside these instructions, but once you have the apache web server installed, make sure there is a directory where users can get http access to file and directory listings. All SSDS data files and generated products will be stored in some local directory. The idea is that you make that directory available through an HTTP server and all SSDS managed assets become available over HTTP. So once you have installed Apache and have a directory that can be browsed via http, remember the local directory location for later. For example, on the Mac, an Apache server is already installed and if you turn on Web Sharing, you can make your /Users/kgomes/Sites directory available via HTTP. So, I chose to create a directory in my Sites folder called ssdsdata that is then browseable via http://localhost/~kgomes/ssdsdata. So the directory to remember is:

   ```
   content.directory.location=/Users/kgomes/Sites/ssdsdata
   ```

   TODO detail an installation of OPeNDAP on Apache

   It is helpful to add an OPeNDAP server to the apache server so some of the products can be served via OPeNDAP. I still need to document how to do that.

4. The next thing to do is install version 4.2.2GA of JBoss. You can retrieve that from http://jboss.org. After you download the .ZIP file, you can simply unzip it to create a new jboss-4.2.2GA folder. For this example, I unzipped the file to /Applications on the Mac so my Jboss home directory is. You will want to make sure you have full write and execution permissions on this directory.

   ```
   jboss.home=/Applications/jboss-4.2.2.GA
   ```

   Feel free to Run JBoss

   Just to make sure that JBoss will run OK, I usually like to run it once before building SSDS just to make sure it runs OK. Open a command prompt (terminal) and cd to the jboss home directory. Then run (at least on the Mac):

   ```
   sudo ./bin/run.sh
   ```

   A whole bunch of stuff should go by and eventually you should see something like:

   ```
   16:37:08,072 INFO  [Server] JBoss (MX MicroKernel) [4.2.2.GA (build:
   SVNTag=JBoss_4_2_2_GA date=200710221139)] Started in 9s:33ms
   ```

   This means JBoss is up and running. You can also browse to http://localhost:8080 and make sure you can view it through a browser. You can then shutdown JBoss by going back to the terminal window and typing Cntl-C.

5. For Flex development, download and install the Adobe Flex SDK. You can download the SDK from http://opensource.adobe.com/wiki/display/flexsdk/Downloads. For this example, I installed it to the /Applications folder so I ended up with:

   ```
   FLEX_HOME=/Applications/flex_sdk_3.3.0.4852
   ```

6. Now you will need to setup the database server where you want the SSDS to store the metadata and data that it manages. These instructions show how to do it on a remote Microsoft SQL Server, but at some point, I will try to write up parallel instructions for MySQL. I will not go through the setup of the SQL Server, but once it is up and running, you will need to create two databases. For these instructions, I created 'SSDS_Data' and 'SSDS_Metadata'. Also, create an account that has database ownership on both and remember the following information for later:

   ```
   database.server.name=database.host.name
   database.server.login.username=dbo_username
   database.server.login.password=dbo_password
   ```

7. Check out the SSDS code base from Google Code.
   a. You will have to have a Google account to check out the code
   b. This step will depend on the subversion client you use, but as an example, here is how you would do it with command line as a project member (can make changes)

      ```
      svn checkout https://shore-side-data-system.googlecode.com/svn/trunk/ shore-side-data-system --username jdoe
      ```

      A read-only checkout can happen anonymously like:

      ```
      svn checkout http://shore-side-data-system.googlecode.com/svn/trunk/ shore-side-data-system-read-only
      ```

      Ignore directories

      If you are using a GUI client for subversion that can ignore directories, you will want to ignore the following directories (they won't appear until you run ant):

      - build
      - dist
      - src/gen

8. You should now have everything you need installed to get the SSDS up and running. The next step is to configure properties so that you can build and deploy the SSDS components to their various locations and then startup JBoss to start the SSDS. In order to configure the build to be appropriate for you configuration, you will need to create a custom.properties file in the root of the SSDS code base and set all the properties to match you installation. Fortunately, we have created a template that you can work from. Copy custom.properties.template to custom.properties and then edit the properties to match your configuration. Use the instructions in the template to help you define the properties correctly and you will need the information gathered above to complete the property configuration.
9. During the custom.properties configuration, you defined various aspects for the LDAP security that will back the SSDS. You need to then define security roles (i.e. LDAP groups) in src/web/src/WEB-INF/web.xml

   ```
   <security-constraint>
   <web-resource-collection>
   <web-resource-name>login result page</web-resource-name>
   <url-pattern>/loginResult.jsp</url-pattern>
   </web-resource-collection>
   <auth-constraint>
   <!-- Place your LDAP groups that are authorized to login here -->
   <role-name>Engineering Distribution List</role-name>
   <role-name>Research Distribution List</role-name>
   <role-name>ITD Distribution List</role-name>
   <role-name>DMO Distribution List</role-name>
   <role-name>OED Distribution List</role-name>
   </auth-constraint>
   </security-constraint>
   ```

10. Whew, once all that is done, you are ready to build and deploy the SSDS. Open command prompt, cd to directory where SSDS was checked out and type

```
ant -Dtarget=deploy
```

11. Once the build is complete, open a terminal window, change directory to where the JBoss home is, and start JBoss

```
sudo ./bin/run.sh
```

12. Once JBoss finishes startup, you should see the line:

```
17:12:41,607 INFO  [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA
date=200710221139)] Started in 19s:45ms
```

and hopefully you did not see any stack traces fly by during startup. If all looks OK, you should be able to go the SSDS_Metadata database and see newly created tables that will be where SSDS will store is metadata. You can also open a browser and go to http://localhost:8080/ssds-docs/ to see a simple page with some links to SSDS documentation and various utilities and libraries.

> ⚠ **Still to document**
>
> a. Configure mod_jk in Apache/JBoss
> b. Configure SSL for login.jsp page

Setup an Eclipse project:

In order to actually do some development on the SSDS code base, you can use an IDE like Eclipse to edit the code and use ant to deploy your changes to JBoss. To setup an Eclipse project.

1. Install eclipse that you download from http://www.eclipse.org
2. Startup Eclipse and select File->New->Java Project.
3. In the New Project Window, select 'Create project from existing source' and browse to where you checked out the SSDS source code. Once you configure the project correctly, you should have the following project settings:
   a. There should be two source directories:

      ```
      src/java
      ```

      and

      ```
      src/gen
      ```

      (src/gen is created by ant during build)
4. Add all jars in the lib directory
5. Add src/resources/build/antlr/antlr-2.7.5.jar
6. Add JBoss jars
   a. JBOSS_HOME/client/activation.jar
   b. JBOSS_HOME/client/servlet-api.jar
   c. JBOSS_HOME/client/jboss-j2ee.jar
   d. JBOSS_HOME/client/log4j.jar
   e. JBOSS_HOME/server/default/lib/commons-codec.jar
   f. JBOSS_HOME/server/default/lib/commons-collections.jar
   g. JBOSS_HOME/server/default/lib/commons-httpclient.jar
   h. JBOSS_HOME/server/default/lib/hibernate3.jar
   i. JBOSS_HOME/server/default/lib/mail.jar
7. output set to build/classes (to align with ant's build files)

Create a FlexBuilder (plug-in) project:

1. Start Eclipse with Flex-Builder plug-in installed
2. File->New->Other..
3. Select Flex Builder->Flex Project
4. Type in 'ssds-flex' for name
5. Uncheck 'Use default location'
6. Browse to SSDS_HOME/src/web and select choose
7. Select 'Web Application'
8. Select 'J2EE' as Application Server Type
9. Check Use remote object access service

10. Click Next>
11. Uncheck 'Use default location for Local LifeCycle Data Service server'
12. Browse to SSDS_HOME/src/resources/build/flex and choose it for the 'Root folder'
13. Change 'Root URL' to the URL of the servlet context (for example, on localhost it would be 'http://localhost:8080/servlet/')
14. Change the 'Context Root'to '/servlet/'
15. Select 'Compile Application Locally in Flex Builder'
16. For 'Output folder location' put your deployment directory for the web application (for example /Users/kgomes/Applications/jboss-4.2.2.GA/server/default/deploy/ssds.war)
17. Click Validate Configuration (it will warn that the output folder is not a subfolder of the server root (that is OK).

> **Will create a Main.mxml**
>
> Note that the creation of the Flex Builder project will create a Main.mxml file. To fix this, right click on explorer.mxml and choose 'Set As Default Application', then you can delete the Main.mxml file.

18. Click on Finish

NOTE: To run the tests fully, you must have perl installed with the following module:

1. Class-ObjectTemplate-0.7 (http://search.cpan.org/~jasons/Class-ObjectTemplate-0.7/)

Windows Installation

This page last changed on Jun 03, 2010 by kgomes.

For the CANON project, Jnaneshwar Das at USC wanted to get SSDS installed to manage the data from their gliders. He setup a WindowsXP machine with the following information:

1. Hostname: amphisbaena.usc.edu
2. Static IP: 128.125.125.51
3. Username: kevin

Here are the steps I took to install and configure SSDS (all the downloaded files went into the 'SSDS Installation' directory on the Desktop):

1. I logged in to the Windows machine using Remote Desktop
2. I opened up Internet Explorer and browsed to http://java.sun.com
3. I downloaded Java SE 6 **JDK** and ran the .exe to install it
   a. I am weird, but I changed the default installation location to C:\bin\Java\jdk1.6.0_18 (I hate spaces in directories and paths)
   b. Same with the JRE, I installed it to C:\bin\Java\jre6
   c. Also, create an environment variable for JAVA_HOME and point it to the Java **JDK** location
4. After installing Java, I browsed to http://ant.apache.org and downloaded Ant 1.8.0 (I won't go through the steps here, but don't forget to add the ANT_HOME environment variable and the ANT_HOME\bin directory to the PATH environment variable
5. I installed the Apache HTTP server by browsing to http://httpd.apache.org and I downloaded the Win32 Binary including OpenSSL for 2.2.15
   a. Again, 'cause I am weird and old, I installed it to C:\bin\Apache2.2 and accepted the default values

> **Did not use Apache**
>
> Turns out IIS was installed already, so I just used that

6. In order to use IIS to get HTTP access to SSDS files, I had to configure it for directory browsing.
   a. Under Control Panel->Administrative Tools->Internet Information Services, right-click on Default Web Site and select Properties.
   b. Then on the Home Directory tab, check the box next to Directory browsing.
7. I created the directory C:\Inetpub\wwwroot\ssdsdata where I will have SSDS store everything. With the data there, it will then be HTTP accessible.
8. I then browsed to http://jboss.org and downloaded JBoss 4.2.2GA and unzipped it to the C:\bin \jboss-4.2.2GA directory
   a. I started up JBoss from the command line and noticed that port 8080 had been taken by the SQL services. I shutdown all SQL services, started JBoss so it could claim those ports, then restarted the SQL components.
9. I then installed SmartSVN so that I could get the code from the Google SVN server.
10. I started up SmartSVN and added a repository for Google code:
    a. SVN Location: http://shore-side-data-system.googlecode.com/svn/trunk
    b. Read only username: shore-side-data-system-read-only
11. I checked the code out read-only to a directory with the name of the server (amphisbaena.usc.edu).
12. I installed the Flex 4 SDK.
    a. I downloaded the SDK as a zip file and then extracted it in the c:\bin directory

Instrument Swap on Oasis Mooring

This page last changed on Jun 16, 2009 by mccann.

This is the procedure to take when OSG swaps an instrument on an OASIS mooring in order to keep the metadata and data all lined up in SSDS. The easiest way is to try to do these steps exactly when they actually do the instrument swap. The reason is that due to the fact that the data from the instrument is downloaded to the same file in the OASIS directory so there is no way (currently) to automate some sort of notice that the instrument has been swapped. An external process reads that raw data file from the instrument, looks up the device ID from a shore-side configuration file and then publishes that data to SSDS under that device ID. If the timing is not right, some extra steps need to be taken. These steps will assume that the timing is correct and I will add steps at the end in case this is being done after the swap happened (usually the case).

1. Get new device ID of the new instrument to be installed.
2. Check out the XML for that instrument from the 'puckxml' project in CVS.
3. Use a validating XML editor like XML Spy, oXygen, or jEdit to open the XML file.
4. Make sure the schema location at the top of the XML file points to:
    a. http://new-ssds.mbari.org/ssds-docs/xml/schema/SSDS_Metadata.xsd
5. Run the editor's validation on the XML.
6. If it does not validate, fix errors
7. Remove any deployment attributes from the <Deployment> tag. For instance any nominalLat/Lon/Depth. The one exception is the nominalDepth, if it is known please set it.
8. If the <Deployment> tag has a 'name' attribute, make sure it does not have any deployment specific information in it. For example, 'ISUS Deployment' is better than 'M2 ISUS Deployment'. The reason for removing any deployment information from the XML is so that when the device moves to a different mooring, the user should not have to edit the XML. The goal is to get all the XML to a point where it never needs to be edited when an instrument is deployed (unless something in the way the data stream is generated from the instrument changes).
9. Go to the SSDS Device pages and verify that the all the device information (mfg, model, serial number, name, type, etc.) matches what is currently in SSDS. If any of those are different it will update the device information in SSDS when the XML comes in the data stream.
10. Verify RecordDescription and RecordVariables look correct. I usually go to the raw data pages in SSDS and bring up the last few packets from the device just to verify that the number of columns and bufferSeparator look about right.
11. Check any changes to the XML back into CVS.
12. Copy the XML to the \\Tornado\ssdsdata\mooring(m1|m2)\YYYY\xml directory
13. Go to the \\Tornado\ssdsdata\mooring(m1|m2)\YYYY\cfg directory.
14. This next step is the one that needs to be timed with the mooring turn. When the old instrument is shutdown:
    a. Open the ssds.cfg file in a text editor
    b. Find the line that shows the currently deployed instrument and copy it to a line just below it. For example, if we are replacing the GPS, it might look like this before:

Before Copy

```
instrument = PCO2,1471,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1471.xml
instrument = Metsys,1480,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1480.xml
instrument = GPS_TYPE3,1416,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1416.xml,,,TransformGPS
instrument = Spec_PRR,1420,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1420.xml
instrument = ADCP,1417,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1417.xml
```

and this after:

After Copy

```
instrument = PCO2,1471,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1471.xml
instrument = Metsys,1480,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1480.xml
```

```
instrument = GPS_TYPE3,1416,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1416.xml,,,TransformGPS
instrument = GPS_TYPE3,1416,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1416.xml,,,TransformGPS
instrument = Spec_PRR,1420,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1420.xml
instrument = ADCP,1417,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1417.xml
```

15. Now change the new line to have the correct device ID **and** the correct XML file URL

After Device ID update

```
instrument = PCO2,1471,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1471.xml
instrument = Metsys,1480,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1480.xml
instrument = GPS_TYPE3,1416,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1416.xml,,,TransformGPS
instrument = GPS_TYPE3,1511,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1511.xml,,,TransformGPS
instrument = Spec_PRR,1420,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1420.xml
instrument = ADCP,1417,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1417.xml
```

16. To clean up the previous deployment information, put start and end dates after the XML URL

After Adding Start/End dates

```
instrument = PCO2,1471,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1471.xml
instrument = Metsys,1480,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1480.xml
instrument = GPS_TYPE3,1416,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1416.xml,2007/04/25 16:21:58,2007/08/01 10:00:00,TransformGPS
instrument = GPS_TYPE3,1511,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/
xml/1511.xml,,,TransformGPS
instrument = Spec_PRR,1420,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1420.xml
instrument = ADCP,1417,http://dods.mbari.org/data/ssdsdata/mooring/m2/2007/xml/1417.xml
```

(Be careful on the format for the start and end date strings. Leading 0s are required.)

17. Save the cfg file.

> ⚠ **Saving the file will make the change take hold**
> When the ssds.cfg file changes (saved) is when the OASIS2SSDS processing will pick up the instrument change. Now, the next time it runs it will pick up the instrument change, grab the XML file from the 'xml' directory and publish it to SSDS. It will then publish all data under the new device ID.

18. Edit the metadata in SSDS to put a close date on the old instrument deployment in SSDS. Currently I do that using Enterprise Manager.

If this is being done after the fact the next steps will also need to be taken.

1. After the new deployment shows up in SSDS (which can take up to 5 minutes after the OASIS2SSDS has completed), the start time for the new deployment will need to be edited to match the actual time the instrument was swapped.
2. Also because the data was being published under the incorrect device ID, it will need to be moved from one database table in SSDS_Data on Solstice to another table.
   a. The first thing that I do is grab the timestamp from the last packet sent from the old device in the raw data page on SSDS.

b. For example, I go to: http://new-ssds.mbari.org:8080/ssds/siamRawDataStep1.jsp and enter the old device ID and set the number of packets back to make sure it goes far enough back to cover the actual time of the instrument swap. Then click on 'Next->'.
c. Once the raw data shows up, find the last packet from the old device and grab the 'SIAM Timestamp' value (not the date/time) as that will be used in the Enterprise Manager query.
d. Open Enterprise Manager and navigate to the 'SSDS_Data' database on Solstice.
e. Browse the tables and find the table with the device ID of the old device and right click on it and select 'Open Table->Return all rows'.
f. Click on the 'SQL' button in Enterprise Manager to bring up the SQL pane. It should show the basic query which should look something like this:

```
SELECT     *
FROM         [1416]
```

g. Now add the where clause to pick only the data that is after the timestamp you grabbed from the last packet on the web page.

> **Timestamps in SQL are in Seconds**
> A quick note here, the 'SIAM Timestamp' on the raw data page is actually in milliseconds and the database column is in seconds so you will have to remove the last three digits of the 'SIAM Timestamp' before putting it in this query.

```
SELECT     *
FROM         [1416]
WHERE timestampSeconds > 1185963023
```

h. Run this query by clicking the run button '!' in Enterprise Manager.
i. Look over the results to make sure they look about right (usually you are looking for the length of the return which should be much shorter). You can actually use a count query to see how many rows this query will return. A count query would look like:

```
SELECT    count(*)
FROM         [1416]
WHERE timestampSeconds > 1185963023
```

j. Once you know the query is correct (also compare sequence number in query return and the raw data page), copy it to the clipboard and close the query window in Enterprise Manager.
k. Navigate to the table of the device you want to copy the data into and right click and select 'All Tasks->Import Data...' which will fire up the DTS wizard.
    i. Click on 'Next>'
    ii. For the Data Source database choose Solstice
    iii. Select the 'SSDS_Data' database (note you should have permissions to do all this and use your windows authentication)
    iv. Click on 'Next>'
    v. The destination configuration should be already to go (Solistice and SSDS_Data database).
    vi. Click on 'Next>'
    vii. Select 'Use a query to specify the data to transfer'
    viii. Click on 'Next>'
    ix. Paste the query from your clipboard into the 'Query Statement' window (You can click on 'Parse' if you want a quick sanity check)
    x. Click on 'Next>'
    xi. Click on the 'Results' entry under the 'Destination' column which will enable a drop down box.
    xii. Choose the table of the newly installed device where you will be copying the data to.
    xiii. Click on 'Next>'
    xiv. Click on 'Next>'
    xv. Click on 'Finish' which will copy the data.
    xvi. Once that is done, open the table of the old instrument and the SQL pane so that we can construct the delete query on the old data.
    xvii. Paste in the select query and verify it is the same data you copied over:

```
SELECT     *
FROM         [1416]
WHERE timestampSeconds > 1185963023
```

    xviii. If it looks good, click on the 'Change Query type ...' button in Enterprise Manager and select 'Delete'. This will change the query to a delete query.

xix. Run the query by click on the run '!' button. That will remove all the data from the old instrument.

That's it ... whew!

Kevin Gomes (August 3, 2007)

M0 Text File Generation Scripts

This page last changed on May 30, 2008 by mccann.

M0 Text File Generation Scripts - Documentation

Created by Seth Bushinsky - May 30, 2008

Purpose:

These programs produce text files from available M0 data. Two types of text files are produced: one set of files for each M0 deployment which consists of one text file per instrument and two text files (one for surface data, one for profile data) that include selected data for all M0 deployments that can be read into Loboviz (http://www.mbari.org/lobo/loboviz.htm) All files can be read using ODV or any text reader.

Deployment Instrument Files:

Location: \\Tornado\ssdsdata\deployments\m0\ [YYYYMM] \ascii_output

Example (\\Tornado\ssdsdata\\deployments\m0\200706\ascii_output) Files created by 'nc_loaddap_text.m' called with the appropriate deployment date by a crontab script on Elvis (see table at the end of this document).

For mooring turnaround,

go to new deployment folder (\\Tornado\ssdsdata\deployments\m0\ [YYYYMM]) and create sub-directory titled "ascii_output". Then change deployment call in crontab script - "cron_m0_txt_process". This should find all '.nc' files being created by ssds and make text files from these.

Loboviz Text Files:

This program produces two text files: M0PROF.txt and M0SURF.txt (along with M0PROF.cfg and M0SURF.cfg). Once a day, these files are copied by Loboviz and ingested so that the data can be plotted online. File location: \\Tornado\ssdsdata\deployments\m0\ascii_all_dep. Generated by "nc_loaddap_odv_surf.m" and "nc_loaddap_odv_prof.m". These files read in the netcdf files generated by ssds, as well as nitrate data from \\Tornado\ssdsdata\isuscimt\data\M0.txt . The netcdf file names are hard-coded into the processing. This script is run once a day. It deletes the last 3 hours of data in the file (which are usually NaN's) and appends new data to the end of the files. It also generates the '.cfg' files that contain the number of lines of data in each file.

For mooring turnaround,

open *both*"nc_loaddap_odv_surf.m" and "nc_loaddap_odv_prof.m", go down to where the long list of netcdf files is under the heading "%% Load Variables" and add the new deployments' netcdf files to the list.

*If there is a change to the mooring data or some other error, keep in mind that because these files are appended instead of re-written, the mistake will stay in the text files. If this happens, simply delete the files and new ones will be created. However, when this happens, open the text files the next day and look in the header for weird characters that sometimes get added next to the "micro" or "degree" symbols. These can trip up Loboviz and prevent data ingestion.

| *Program | Machine | Crontab Script | Scripts called (next table has script location) | '.m' file location |

Text file generation	Elvis (user: ssdsadmin, pw: water4u)	/bin/sh / ssdsdata/ deployments/ m0/ ascii_all_dep/ cron/ cron_m0_txt_process	nc_loaddap_text.m	\\Tornado \ssdsdata \deployments \m0\ascii_all_dep \mfiles	
	Elvis				

Loboviz file generation		/bin/sh / ssdsdata/ deployments/ m0/ ascii_all_dep/ cron/ cron_m0_loboviz	nc_loaddap_odv_swfom, nc_loaddap_odv_psofsd	\\Tornado \ssdsdata \deployments \m0\ascii_all_dep \mfiles	

Migration to Google Code Base

This page last changed on Apr 26, 2010 by kgomes.

In order to make sure I can move the production application to the Google code base, I need to go through each .java file and make sure I have tests and documentation written for each one. Here is the listing of all the .java files currently in the SSDS code base:

Classes

Package	File	Test Class	Complete
moos.ssds.clients.graphing	DeviceQCPlotCreator.java		
moos.ssds.clients.ssdsLoader	FileParser.java		
moos.ssds.clients.ssdsLoader	SubmitFiles.java		
moos.ssds.clients.updateBot	UpdateBot.java		
moos.ssds.clients.updateBot	UpdateBotListener.java		
moos.ssds.clients.updateBot	UpdateBotRunner.java		
moos.ssds.dao	DataContainerDAO.java		
moos.ssds.dao	DataContainerGroupDAO.java		
moos.ssds.dao	DataProducerDAO.java		
moos.ssds.dao	DataProducerGroupDAO.java		
moos.ssds.dao	DeviceDAO.java		
moos.ssds.dao	DeviceTypeDAO.java		
moos.ssds.dao	EventDAO.java		
moos.ssds.dao	HeaderDescriptionDAO.java		
moos.ssds.dao	IMetadataDAO.java		
moos.ssds.dao	KeywordDAO.java		
moos.ssds.dao	MetadataDAO.java		
moos.ssds.dao	PersonDAO.java		
moos.ssds.dao	RecordDescriptionDAO.java		
moos.ssds.dao	RecordVariableDAO.java		
moos.ssds.dao	ResourceDAO.java		
moos.ssds.dao	ResourceTypeDAO.java		
moos.ssds.dao	SoftwareDAO.java		
moos.ssds.dao	StandardDomainDAO.java		
moos.ssds.dao	StandardKeywordDAO.java		
moos.ssds.dao	StandardReferenceScaleDAO.java		

moos.ssds.dao	StandardUnitDAO.java		
moos.ssds.dao	StandardVariableDAO.java		
moos.ssds.dao	UserGroupDAO.java		
moos.ssds.dao.util	MetadataAccessException.java		
moos.ssds.data.converters	NetcdfConverter.java		
moos.ssds.data.graphing	AnchorDrawer.java		
moos.ssds.data.graphing	ColorBarBugFix.java		
moos.ssds.data.graphing	DegreesMinutesNumberFormat.java		
moos.ssds.data.graphing	DistanceScaleDrawer.java		
moos.ssds.data.graphing	GpsCharter.java		
moos.ssds.data.graphing	LatLonConverter.java		
moos.ssds.data.graphing	LocationAndTimeDataset.java		
moos.ssds.data.graphing	WatchCircleDrawer.java		
moos.ssds.data.graphing	XYCircleRenderer.java		
moos.ssds.data.graphing	XYPlotWithColorBar.java		
moos.ssds.data	IDataAccess.java		
moos.ssds.data	ITimeIndexedDataAccess.java		
moos.ssds.data.parsers	AsciiFileContext.java		
moos.ssds.data.parsers	AsciiFixedWidthParser.java		
moos.ssds.data.parsers	AsciiPacketParser.java		
moos.ssds.data.parsers	AsciiRecordParser.java		
moos.ssds.data.parsers	BinaryFileContext.java		
moos.ssds.data.parsers	BinaryPacketParser.java		
moos.ssds.data.parsers	BinaryRecordParser.java		
moos.ssds.data.parsers	IParser.java		
moos.ssds.data.parsers	MultipartMimeFileContext.java		
moos.ssds.data.parsers	Nmea21PacketParser.java		
moos.ssds.data.parsers	Nmea21RecordParser.java		
moos.ssds.data.parsers	PacketParser.java		
moos.ssds.data.parsers	PacketParserContext.java		
moos.ssds.data.parsers	Parser.java		
moos.ssds.data.parsers	ParserContext.java		

moos.ssds.data.parsers	ParsingException.java		
moos.ssds.data.parsers	RecordParser.java		
moos.ssds.data.parsers	VariableFormatMap.java		
moos.ssds.data	TimeIndexedDataAccessFactory.java		
moos.ssds.data	TimeIndexedFreeFormAccess.java		
moos.ssds.data	TimeIndexedNetcdfAccess.java		
moos.ssds.data	TimeIndexedPacketAccess.java		
moos.ssds.data.util	DataException.java		
moos.ssds.data.util	LocationAndTime.java		
moos.ssds.ingest	IngestMDB.java		
moos.ssds.ingest	SQLIngestMDB.java		
moos.ssds.io	PacketInput.java		
moos.ssds.io	PacketOutput.java		
moos.ssds.io	PacketOutputManager.java		
moos.ssds.io	PacketSQLInput.java		
moos.ssds.io	PacketSQLOutput.java		
moos.ssds.io.util	Base64.java		
moos.ssds.io.util	CountInputStream.java		
moos.ssds.io.util	PacketMerger.java		
moos.ssds.jms	PublisherComponent.java		
moos.ssds.jms	SubscriberComponent.java		
moos.ssds.metadata	CommentTag.java		
moos.ssds.metadata	DataContainer.java		
moos.ssds.metadata	DataContainerGroup.java		
moos.ssds.metadata	DataProducer.java		
moos.ssds.metadata	DataProducerGroup.java		
moos.ssds.metadata	DateRange.java		
moos.ssds.metadata	Device.java		
moos.ssds.metadata	DeviceType.java		
moos.ssds.metadata	Event.java		
moos.ssds.metadata	HeaderDescription.java		
moos.ssds.metadata	IDateRange.java		

moos.ssds.metadata	IDescription.java		
moos.ssds.metadata	IMetadataObject.java		
moos.ssds.metadata	IResourceOwner.java		
moos.ssds.metadata	Keyword.java		
moos.ssds.metadata	Person.java		
moos.ssds.metadata	RecordDescription.java		
moos.ssds.metadata	RecordVariable.java		
moos.ssds.metadata	Resource.java		
moos.ssds.metadata	ResourceBLOB.java		
moos.ssds.metadata	ResourceType.java		
moos.ssds.metadata	Software.java		
moos.ssds.metadata	StandardDomain.java		
moos.ssds.metadata	StandardKeyword.java		
moos.ssds.metadata	StandardReferenceScale.java		
moos.ssds.metadata	StandardUnit.java		
moos.ssds.metadata	StandardVariable.java		
moos.ssds.metadata	UserGroup.java		
moos.ssds.metadata.util	MetadataException.java		
moos.ssds.metadata.util	MetadataFactory.java		
moos.ssds.metadata.util	MetadataValidator.java		
moos.ssds.metadata.util	ObjectBuilder.java		
moos.ssds.metadata.util	XmlBuilder.java		
moos.ssds.ruminate	Handler.java		
moos.ssds.ruminate	MetadataHandler.java		
moos.ssds.ruminate	RuminateMDB.java		
moos.ssds.ruminate	XMLMetadataTracker.java		
moos.ssds.services.blazeds	EJBFactory.java		
moos.ssds.services.blazeds	IResourceLocator.java		
moos.ssds.services.blazeds	LocalCachingJNDIResourceLocator.java		
moos.ssds.services.blazeds	LocalJNDIResourceLocator.java		
moos.ssds.services.blazeds	ResourceException.java		
moos.ssds.services.blazeds	SessionRO.java		

moos.ssds.services.data	DeviceDataAccessEJB.java		
moos.ssds.services.data.graphing	GeospatialGraphingAccessEJB.java		
moos.ssds.services.data	PacketFileToSQLServicePublisher.java		
moos.ssds.services.data	PacketSubmissionAccessEJB.java		
moos.ssds.services.data	RecordVariableDataAccessEJB.java		
moos.ssds.services.data	SQLDataStreamRawDataAccessEJB.java		
moos.ssds.services.data.util	DataException.java		
moos.ssds.services.metadata	AccessBean.java		
moos.ssds.services.metadata	DataContainerAccessEJB.java		
moos.ssds.services.metadata	DataContainerGroupAccessEJB.java		
moos.ssds.services.metadata	DataProducerAccessEJB.java		
moos.ssds.services.metadata	DataProducerGroupAccessEJB.java		
moos.ssds.services.metadata	DeviceAccessEJB.java		
moos.ssds.services.metadata	DeviceTypeAccessEJB.java		
moos.ssds.services.metadata	EventAccessEJB.java		
moos.ssds.services.metadata	IMetadataAccess.java		
moos.ssds.services.metadata	IMetadataAccessRemote.java		
moos.ssds.services.metadata	KeywordAccessEJB.java		
moos.ssds.services.metadata	PersonAccessEJB.java		
moos.ssds.services.metadata	RecordVariableAccessEJB.java		
moos.ssds.services.metadata	ResourceAccessEJB.java		
moos.ssds.services.metadata	ResourceTypeAccessEJB.java		
moos.ssds.services.metadata	SoftwareAccessEJB.java		
moos.ssds.services.metadata	StandardDomainAccessEJB.java		
moos.ssds.services.metadata	StandardKeywordAccessEJB.java		
moos.ssds.services.metadata	StandardReferenceScaleAccessEJB.java		
moos.ssds.services.metadata	StandardUnitAccessEJB.java		
moos.ssds.services.metadata	StandardVariableAccessEJB.java		
moos.ssds.services.metadata	UserGroupAccessEJB.java		
moos.ssds.services.servlet.gps	GoogleMapsGPSDataServlet.java		
moos.ssds.services.servlet.data	DataAccessServlet.java		
moos.ssds.services.servlet.data	GetOriginalDataServlet.java		

moos.ssds.services.servlet	MetadataAccessServlet.java		
moos.ssds.services.servlet	SensorMLConverterServlet.java		
moos.ssds.services.servlet	MethodProxy.java		
moos.ssds.services.servlet	ServletFaultHandler.java		
moos.ssds.services.servlet	ServletUtils.java		
moos.ssds.simulator	DataPacket.java		
moos.ssds.simulator	FileInfo.java		
moos.ssds.simulator	PacketGenerator.java		
moos.ssds.transmogrify	SIAMMetadataTracker.java		
moos.ssds.transmogrify	SSDSDevicePacket.java		
moos.ssds.transmogrify	SSDSGeoLocatedDevicePacket.java		
moos.ssds.transmogrify	TransmogrifyMDB.java		
moos.ssds.util	DateUtils.java		
moos.ssds.util	XmlDateFormat.java		
moos.ssds.wrapper.ogc.sensorml	SensorMLFactory.java		
moos.ssds.wrappergenerator	Method.java		
moos.ssds.wrappergenerator	MyClass.java		
moos.ssds.wrappergenerator	Parameter.java		
moos.ssds.wrappergenerator.antlr	Main.java		
moos.ssds.wrappergenerator.antlr	JavaRecognizer.java		
moos.ssds.wrappergenerator.antlr	JavaTokenTypes.java		
moos.ssds.wrappergenerator.antlr	JavaTreeParser.java		
moos.ssds.wrappergenerator.antlr	JavaTreeParserTokenTypes.java		
moos.ssds.wrappergenerator	PerlGenerator.java		
moos.ssds.wrappergenerator	TranslationController.java		
moos.ssds.wrappergenerator	TranslationGenerator.java		
org.mbari.io	FilterCommentInputStream.java		
org.mbari.io	URLDataBuffer.java		
org.mbari.util	Email.java		
org.mbari.util	IsBinary.java		
org.mbari.util	MathUtil.java		
org.mbari.util	SwingDecorators.java		

org.mbari.util	SwingWorker.java		
org.mbari.util	SystemUtil.java		
org.mbari.util	URLUTF8Encoder.java		

Test Classes

Package	File
test.moos.ssds.data.parsers	TestAsciiPacketParser.java
test.moos.ssds.data.parsers	TestAsciiRecordParser.java
test.moos.ssds.data.parsers	TestBinaryPacketParser.java
test.moos.ssds.data.parsers	TestBinaryRecordParser.java
test.moos.ssds.data.parsers	TestNmea21PacketParser.java
test.moos.ssds.data.parsers	TestNmea21RecordParser.java
test.moos.ssds.jms	IngestPubThread.java
test.moos.ssds.jms	TestIngest.java
test.moos.ssds.metadata	TestDataContainer.java
test.moos.ssds.metadata	TestDevice.java
test.moos.ssds.metadata	TestDeviceType.java
test.moos.ssds.metadata	TestPerson.java
test.moos.ssds.metadata	TestStandardUnit.java
test.moos.ssds.metadata	TestStandardVariable.java
test.moos.ssds.metadata.util	LoggingDifferenceListener.java
test.moos.ssds.metadata.util	SansSuperficialDifferenceListener.java
test.moos.ssds.metadata.util	TestMetadataFactory.java
test.moos.ssds.metadata.util	TestObjectBuilder.java
test.moos.ssds.metadata.util	TestXmlDateFormat.java
test.moos.ssds.ruminate	TestNetCDFWatchdog.java
test.moos.ssds.ruminate	TestRuminate.java
test.moos.ssds.services.data	TestDeviceDataAccess.java
test.moos.ssds.services.metadata	TestAccessCase.java
test.moos.ssds.services.metadata	TestDataContainerAccess.java
test.moos.ssds.services.metadata	TestDataProducerAccess.java
test.moos.ssds.services.metadata	TestDeviceAccess.java

test.moos.ssds.services.metadata	TestDeviceTypeAccess.java
test.moos.ssds.services.metadata	TestEventAccess.java
test.moos.ssds.services.metadata	TestPersonAccess.java
test.moos.ssds.services.metadata	TestRecordVariableAccess.java
test.moos.ssds.services.metadata	TestStandardUnitAccess.java
test.moos.ssds.services.metadata	TestStandardVariableAccess.java
test.moos.ssds.services.metadata	TestUserGroupAccess.java
test.moos.ssds.transmogrify	TransmogrifyMDBTest.java
test.moos.ssds.util	TestDateUtil.java
test.moos.ssds.wrapper.ogc.sensorml	TestSensorMLFactory.java

new-ssds.mbari.org Setup

This page last changed on Jul 28, 2010 by kgomes.

1. Pat installed RHEL 5
2. He created a local lroot account for me.
3. After talking to IS, in order to mount the Tornado shares properly (AUVCTD, AUVBI, and ssdsdata), we created a domain account named ApacheSSDSRO and I changed the password to something hard to crack.
4. I went on to new-ssds and created a new user ApacheSSDSRO with the same UID as the domain account (1113) and added the group apache to its membership.

   ```
   # adduser -u 1113 -G apache -b /home -s /bin/bash -p ********** -g apache ApacheSSDSRO
   ```
5. I edited the /etc/httpd/conf/httpd.conf file and changed the "User" line from "apache" to "ApacheSSDSRO" which should run the httpd service as ApacheSSDSRO. This was important so that it's UID will get passed to the network share when serving http requests.
6. I ran the chkconfig command to make sure httpd started on reboot

   ```
   # chkconfig --level 35 httpd on
   ```
7. I then edited the /etc/fstab file to mount the tornado shares that SSDS needs:

   ```
   /dev/VolGroup00/LogVol00 /                       ext3    defaults        1 1
   LABEL=/boot             /boot                   ext3    defaults        1 2
   tmpfs                   /dev/shm                tmpfs   defaults        0 0
   devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
   sysfs                   /sys                    sysfs   defaults        0 0
   proc                    /proc                   proc    defaults        0 0
   /dev/VolGroup00/LogVol01 swap                    swap    defaults        0 0
   # MBARI mounts
   tornado.shore.mbari.org:/vol/vol0/ssdsdata /ssdsdata nfs ro 0 0
   tornado.shore.mbari.org:/vol/vol0/AUVCTD /data/auvctd nfs ro 0 0
   tornado.shore.mbari.org:/vol/AUVBI /data/auvbi nfs ro 0 0
   ```
8. I created the directories /data/auvctd, /data/auvbi, /data/ssds/generated, /data/ssds/ruminate/xml, /ssdsdata and made ApacheSSDSRO as the owner and apache as the group for these. (including the parent /data directory).
9. I put in a request to IS to have them restore the /data/ssds/ruminate/xml directory
10. I downloaded jdk1.6.0_20 from Sun (Oracle's) web site to the Desktop on /root and then ran the .bin executable. It created a directory jdk1.6.0_20 which I then moved to /opt
11. I created a symbolic link in /opt to /opt/java which pointed to that folder.
12. I then created symbolic links to all the stuff in /opt/java/bin to links in the /usr/bin directory to put them all on the path

    ```
    ln -sf /opt/java/bin/* /usr/bin
    ```
13. I rebooted here just to make sure everything that I had done to date took:
    a. httpd service started automatically ... yeah!
    b. mounts were successful ... yeah!
14. Now in order to expose those directories as http shares so people can access them, I created symlinks to those directories in /var/www/html
15. Once the backup of /data/ssds stuff was done, IS re-enabled the rsync (running on pismo) so that the files from /data/ssds are copied to /ssdsdata/ssds
16. I downloaded jboss-4.0.3SP1 from jboss.org, unzipped and untarred the file on my desktop
17. I moved the newly created jboss-4.0.3SP1 folder to /opt
18. I changed ownership of that directory to ApacheSSDSRO and apache as group
19. I copied the jboss_init_redhat.sh script from the bin directory in jboss to the /etc/init.d directory and renamed to just "jboss"
20. I then edited that script and changed:

    ```
    JBOSS_HOME=${JBOSS_HOME:-"/usr/local/jboss"}
    ```

    to

    ```
    JBOSS_HOME=${JBOSS_HOME:-"/opt/jboss"}
    ```

    and:

    ```
    JBOSSSH=${JBOSSSH:-"$JBOSS_HOME/bin/run.sh -c all"}
    ```

    to:

```
JBOSSSH=${JBOSSSH:-"$JBOSS_HOME/bin/run.sh -b 134.89.2.25"}
```

so it will run the default server and it will bind to 134.89.2.25 (this needed to be done because requests to the naming service would return and IP of 127.0.0.1 which would cause clients to barf).

> **CAUTION on binding**
>
> Originally, I had assigned the bind to 0.0.0.0 as that was supposed to allow it to bind to all IP addresses associated with the server. Normally this seems to work (and it did for the HTTP stuff), but the JMS clients were still getting 127.0.0.1 as the IP back from the server. This was fixed by setting it to the correct IP address. I have not really pushed on it much, but it could also have something to do with the /etc/hosts file which looks like:
>
> ```
> # Do not remove the following line, or various programs
> # that require network functionality will fail.
> 127.0.0.1       new-ssds.mbari.org new-ssds localhost.localdomain localhost
> ::1             localhost6.localdomain6 localhost6
> ```

Also changed:

```
JBOSSUS=${JBOSSUS:-"jboss"}
```

to:

```
JBOSSUS=${JBOSSUS:-"ApacheSSDSRO"}
```

21. I then edited /opt/jboss/bin/run.sh and added the following so that the various HOMES were explicit.

```
export JAVA_HOME="/opt/java"
export JBOSS_HOME="/opt/jboss"
```

22. I edited /opt/jboss/bin/run.conf and changed:

```
JAVA_OPTS="-server -Xms128m -Xmx128m"
```

to:

```
JAVA_OPTS="-server -Djava.awt.headless=true -Duser.timezone=UTC -Xms1024m -Xmx2048m"
```

to makes sure it knows it is not looking for a graphics server and that the timezone to use it UTC and the memory it will use it reasonable.

23. I started up JBoss using the /etc/init.d script and it seemed to start up fine. I could see it from a browser on localhost, but not from another machine (maybe firewall issues?).
24. I then built and deployed the SSDS application on new-ssds.mbari.org (this is a bit involved and not described here).
25. I had to install the mod_jk connector as it was not already installed. I downloaded the binary .so file from the Apache connector project and installed in /usr/lib64/httpd/modules. I also renamed it to just mod_jk.so
26. I then configured mod_jk to server basic port 80 traffic to the SSDS application (see https://oceana.mbari.org/confluence/display/SPEPRJ/Apache+mod_jk)
27. Then I restarted JBoss
28. I.S. had to open some firewall ports for me
29. I then added the jboss script to start up for levels 3 and 5 just like for apache by using:

```
# chkconfig --add jboss
```

30. I was promptly hacked again by leaving the jmx-console and the web-console in place. I removed those by shutting down jboss, removing jmx-console.war and the management directory from the the deploy directory, deleting the tmp and work directories and restarting.
31. As an added security measure, we made root owner of all the files in the jboss directory structure except for the directories listed below which were owned by the process that runs JBoss (ApacheSSDSRO):
    a. JBOSS_HOME/server/default/data
    b. JBOSS_HOME/server/default/tmp
    c. JBOSS_HOME/server/default/work
    d. JBOSS_HOME/server/default/log

32. I then removed the http-invoker.sar from the deploy directory to remove the HTTP invoker for JNDI, EJB and JMX.
33. I then removed the jms/jbossmq-httpil.sar from the deploy directory to remove the HTTP Invoker for JMS.

ProjectRequirements

This page last changed on Oct 28, 2009 by kgomes.

Requirements documents for SSDS development:

1. CIMT Requirements
2. MOOS Science Expermiment Requirements

Consolidated Requirements

Number	Requirement	Description
1.0	User can merge data from different sources	This should be able to be done by the user selecting variables from various sources and a time frame and then SSDS will return a merged data set in various formats. See John's pseudo-code and Perl script for merging script.
2.0	Data conversions can be done by SSDS	SSDS should support user defined data transformations. CTD values to salinity is the golden example of this
3.0	SSDS should connect up/work with AOSN	?
4.0	I want to be able to edit the metadata in the SSDS system.	
5.0	I want a snapshot view of all the currently deployed instruments and what their data stream are doing (this should have links to the raw data, instruments configuration and device information).	
6.0	I want data/plot of variable X (i.e. salinity) from platform Y and time frame.	
7.0	I want data/plot of variable X (i.e. salinity) from geospatial location and time frame.	
8.0	I want a 'shallow' operational view of current platforms and instruments	
9.0	I want to get access to 'all' data from a device	
10.0	I want to find all data of mime type X from a device or platform	
11.0	I want to see all currently deployed devices in a geospatial location (box)	

CIMT Requirements

This page last changed on Apr 17, 2009 by kgomes.

These are the science requirements that were gathered for the CIMT deployment.

Documents

1. We got a document from science about QC plots which is attached here Mooring QC Plot Requirements (Word).
2. This is the same document but John G added categories to try and organize them into group: John G's Categorization (Word)
3. Also a spreadsheet about data processing and analysis for the instruments on CIMT Data Processing and Analysis for CIMT (Word)

Meeting Notes

1. 2004-04-20 CIMT SSDS Meeting Notes

Requirements

2004-04-20 CIMT SSDS Meeting Notes

This page last changed on Apr 14, 2009 by kgomes.

Issues Identified Before and During 2004.04.20 CIMT/SSDS Meeting

1. Q: One of the items is to pass data to/from the SSDS. Does that mean that we (CIMT) are responsible for reproducing the same types of plots and such that MBARI is already creating internally? Or is it possible to combine these two processes?
    a. A: Both processes can be combined. The "passing data from and to" is to enable automated non-SSDS processing.
2. #13 on the prioritization list should be bumped up a bit...if we can't get through the firewall to see the data we are going to get negative feedback from NOAA.
    a. Understood. Is in fact in progress.
3. Are we planning on a watch circle type plot for GPS?
    a. Yes (though note it is in the custom plots category, Priority #16).
4. Is tic mark beginning of day?
    a. Yes, midnight on that day. These formats can be changed.
5. How is development affected by selection of particular plotting package (e.g., JFreeChart) to support SSDS requirements?
    a. It affects how our internal services work (e.g., the callable interface we're using for this demonstration), but users are expected to download data (via TBD interfaces) to work with more advanced applications.
6. How will user know what URL to specify to get what they want?
    a. At first by asking us, but soon enough by filling out a query form (which will then generate a URL, and the URL can be modified and reused).
7. Concern about query page being time-oriented. Can't we make it easy to do this by region too (i.e., lat/lon/depth)? You do need to consider this for CIMT, but a simple (nominal) concept is probably all we should do at first. It doesn't have to be embedded in data, but incorporated in header of a plot. SSDS should eventually be able to query on these 4 parameters: time, lat, lon, depth.
    a. Points well taken. Please note this is hard, and not on the priority list.
8. How do you envision getting data from this system on a recurring basis? This could be used, for example, to connect two data systems together. This is Likely to be common.
    a. We have to design this feature (see Priorities #5 and #8).
9. Would be nice to be able to give feedback about the data, so that the feedback lives with the data.
    a. This is a significant operational challenge across the board at MBARI. But, the architecture we have could be tuned to such a thing. (There are 3 distinct parts to this request; routine automated QC flagging; injecting comments during post-processing, manually or automatically; and user feedback on existing data. The last might be accomplished by publishing the "instrument owner" with the plots.) Note this is not on the priority list.
10. Why can't we query more/better/sooner? How about directly querying the database? Point and click query access can wait, but developer queries must be supported soon. (When will they be available?)
    a. An interface which can support at least some developer queries will be available sometime 'soon' (at its most basic possibly within a few weeks). That kind of interface is implied in Priorities #3, #5, #8, #11, #12, and #16. The query documented by Priority #14 is the point-and-click type.
11. Processing data internally, by moving currently external processing into SSDS, is desirable (e.g., for Metsys and other Bahr data).
    a. We are evaluating all the external processing requirements before proceesing on Priority #8, but my (John G) first approach is to work with existing interfaces as much as possible, for the sake of speed.
12. Add lookup by platform name to device table front end.
    a. Good idea. Have it mind, but not on the priority list.
13. Not clear how to get to a specific data variables/data sets. An interface to 'query for data' is needed, but it isn't on the priority list. (How to describe this need, in the context of the priority list, isn't clear.)
    a. We will be better able to respond to this, and consider what priority it should have, at the next User Story meeting.
14. What time frame are you likely to have some of this work done – do you even know?
    a. We have a decent idea. We expect to be one more step toward a final solution for #8 and #9 by the next User Story meeting on 5/6/04, with a prototype to review. Also at that meeting, we may have significant progress on one or all of the following (current status in parentheses):
    #10: (solution in mind, implementation in progress)
    #11: (prototype demonstrated today, wider access intended by 5/6)

#12: (notional solution in mind)
#13: (PC in house, final architecture still undetermined)
#16: (none currently developed, but this wouldn't be hard, especially with a collaborator)
Status of infrastructure items:
#1: Done for many cases, some cases still in progress.
#2: In progress, 5 MTM instruments described.
#3: Example demonstrated today, some enhancements needed for end users, and still needs to be "released" and possibly put service outside firewall.
#4: Coding can now begin once agreed SIAM metadata interface is documented.
#5: Prototype developed, considerable standardization remains.
#6: In research.
#7: No activity.

Identified/Prioritized User Stories (with caveats).

Items 1-7 are infrastructure services. Most will serve any ingested and described data. Items 8 and beyond are specific CIMT implementations/utilizations of the infrastructure services.

Rough scope is indicated in brackets: A=5 work days, B=5-10 work days, C=10-20 work days.

Done

- Ingest raw SIAM data packets.
- Manage metadata for data packets (basic infrastructure; minor mods needed for CIMT).
- User accesses raw data packets via web (in place for MTM2; tuning needed for CIMT).
- Prototype tool available to confirm consistency of data.

Infrastructure

- Parse ASCII streams A
- Define metadata describing data B
- Create general web plotting capability (to support QC-style, parameter vs time plots only) C
- Simplify SIAM data ingest (internal, no visible behavior change) B
- Add Data notification mechanism or mechanisms (to maximize # of external processes served data) B
- Add data submission mechanism or mechanisms (to maximize # of external processes submitting data) B
- Create a general query interface (expect only one or two basic queries supported for CIMT; see #15 below) B

CIMT-Specific

- Provide data to specific external processes (TBD which ones) B
- Accept data from specific external processes (TBD which ones) A
- Ingest SOON data as a data stream B
- Provide basic plots of raw data (note more advanced plots were deemed not critical by deployment) A
- Merge items according to timestamp, optionally at a given time interval. (Focused on data.)B

Not sure

The following 4 items are on the bubble'; ideally all 4 can be accomplished, but possibly none of them can, depending on speed of the first 12 items. (I expect 2 of these can be done.)

- Make plots available outsisde the firewall (i..e, publicly available) B
- Provide point-and-click query for data sets according to instrument providing the data. A
- Obtain a subset of the data within a particular time interval (ideally including non-SIAM data). B
- Add customized raw plots (e.g., to QC more sophisticated systems). B

Corrections and modifications are encouraged, now or at any User Story meeting.

2004.04.20 Feedback

1. How to access, by instrument or?

2. Go with something more legible for non-specialist.
3. Operational diagnostics under one heading, science variables under another heading.
4. Minimize number of variables (don't show redundant ones).
5. Preconfigure list of most interesting variables.
6. There really should be a way to do our own groupings, not be limited to fixed layout.
7. A week is reasonable standard duration.
8. Would be really useful to have a contact responsible to guide the layout of the primary page.
9. Like to have date axis on top and bottom (or every so often).
10. How do we get data?
11. Confusing to get it from too many pages? Easy to access if there's a button on the page.
12. Would make more sense if redirected to a query page, allow changes to be made.
13. Provision for providing feedback about data would be valuable. (See Issues doc.)
14. Indicate data gaps. Could just plot points. Give user an option as to how to plot.
15. Would like to know reason for the gap.
16. Post-processing can help.
17. Maybe use sequence number.

This page last changed on Oct 24, 2006 by kgomes.

Requirements For MOOS Science Experiment

1. General Requirements
    a. A website like that of CIMT/MTM
    b. A request for calibrated salinity instead of conductivity was made.
    c. A general comment was made how we should have something in place to notify data users that they need to acknowledge MBARI if data is used.
    d. Make sure ITD is OK with sharing images from camera and if 1 year embargo applies to that data.
2. Instrument specific requirements
    a. MTM-4 Surface Node (**SSDS ID 1446**): There appear to be no real SSDS requirements for this. There is no XML defined for this
        i. Satlantic Radiometer (**SSDS ID 1270**): This is a binary instrument that needs special processing to do anything useful with. We have XML defined for it, but I suspect it is out of date. The XML simply states that the DataStream is binary.
        ii. Triaxys directional wave sensor (**SSDS ID 1286**): This instrument has a fairly well behaved data structure. It does not appear that we have XML defined for this instrument, but we should. I know it has been deployed and we are generating plots for it for MTM-3 which has ID of 1339. Here is an example of a data record:
        `$WC,81,060108,1640,13.37,6000,01020,3.33,,132,0,-0.230,0.000,1.700,0.00,28.6,0.00,249,`
        iii. Heave Sensor (is this the Triaxys above?)
            i. Unknown, assuming it will be just like plots on CIMT
        iv. GPS (**SSDS ID 1287**): This is a standard GPS that we have been dealing with. We have XML defined, but we need an application that user's can configure a watch circle and set an alarm that will send an email if it goes outside the circle. Here is an example data record:
        `$GPRMC,104538,A,3648.2007,N,12147.2515,W,000.0,329.4,291105,014.8,E * 6E`
        v. ASIMET LWR (**SSDS ID 1289**): We have seen these instruments before. We don't have any XML in CVS for it, but it seems like we should. There might be a similar instrument on another deployment that we can take the XML from. Here is an example of the data record
        `294.45 294.46 11614.4 11597.3 -74.0 425.0 33284 33438 32760`
        vi. Radiometer, Long-wave, ASIMET (is this the same as above?)
            i. Unknown, but assuming similar to post processing done on other mooring radiometers
        vii. Radiometer, Short-wave, ASIMET (is this the same as above?)
            i. Unknown, but assuming similar to post processing done on other mooring radiometers
        viii. ASIMET HRH (**SSDS ID 1290**): Again, one we have seen before, but no XML. We should have some elsewhere. Here is an example data record:
        `40.634 21.746 : 38124 40892`
        ix. Relative Humidity - Air Temp., ASIMET (is this the same as above?)
            i. Unknown, but assuming it will be like current mooring ASIMETs
        x. ASIMET WND (**SSDS ID 1291**): Same, no XML, but we have instruments just like it. Here is an example of the data:
        `0.00 0.00 0.0 0.0 0.0 147.8 333.2 -4.8 4.7`
        xi. Wind Speed/Direction, ASIMET (is this the same as above?)
            i. Unknown, but assuming it will be like current mooring ASIMETs
        xii. Power Can (**SSDS ID 1322**): We have XML, but we need to see if anything has changed. A (large) example of the data (consult the data stream for more info):
        ```
        ADC00 A+1.5209e-04 N0000600 L+1.5260e-03 H-1.2208e-03 D-3.0520e-04 E0
        ADC01 A+1.8414e-04 N0000600 L+1.2208e-03 H-9.1560e-04 D-3.0520e-04 E0
        ADC02 A+7.8088e-02 N0000600 L+7.3778e-02 H+8.1754e-02 D+0.0000e+00 E0
        ```
        xiii. Load cell 0-10,000 lbs. version A (Bridle)
            i. Unknown, assuming it will be just like plots on CIMT
        xiv. P2 ADC
            i. Unknown, assuming it will be just like plots on CIMT
        xv. Nitrate Analyzer, ISUS

        i. Assuming that Luke will be creating (or adding to the current) ISUS page for this, we will just link to it.
    xvi. CTD (SBE 37SM)
        i. Unknown
    xvii. Current Meter, ADCP, RDI Long Ranger, 600m range
        i. John R thought the processing would be very similar to the ADCP's we have seen on MTM and M0.
    xviii. SeaHorse Profiler (**SSDS ID 1405**): This is a binary format. I think it is a gzipped set of data for each record, but need to chech with Andy Hamilton about it. There is XML for it, but we just say binary.
    xix. MBARI Medusa (**SSDS ID 1441**): There is not XML for this and I think this is a new instrument. We have not seen any data yet, I only see:
    `ERR: Timeout;Timeout;Timeout;retry limit exceeded`
    xx. ASIMET BPR (**SSDS ID 1443**): There is no XML for this and the data looks like:
    `1025.42 1025.42`
    xxi. Pressure, Barometric, ASIMET (Is this the same as above?)
        i. Unknown
    xxii. MSP430 (**SSDS ID 1447**): Although this instrument does not have XML specifically, we have seen this many times before. Data looks like
    `$PEDATA, P1023.73, T27.63, H43.08, GFL0.00, GFH0.00, C301.46, TC0.00, * 3688`
    xxiii. SOON/PCO2 (**SSDS ID 1448**): Plots of raw data like on CIMT. QC Processing will run and plots of that data as well. No XML, but we have seen this instrument before. No data coming yet, we get
    `ERR: while preparing to sample`
    xxiv. Flourometer/Backscatter, Hobielabs HS2 (5000m)
        i. Unknown
    xxv. Radiometer, Satlantic 350-800nm, Hyperspectral Es(air)
        i. Unknown, but assuming similar to post processing done on other mooring radiometers
    xxvi. Radiometer, Satlantic 350-800nm, Hyperspectral Ed-w (water, downwelling)
        i. Unknown, but assuming similar to post processing done on other mooring radiometers
    xxvii. Radiometer, Satlantic 350-800nm, Hyperspectral Lu-w (water, upwelling)
        i. Unknown, but assuming similar to post processing done on other mooring radiometers
b. MSE Axis BIN Medusa subnode 1 (**SSDS ID 1408**): This one only sends message packets, no data
    i. MSP430 (**SSDS ID 1411**): This is the same as the MSP430 on other nodes.
    ii. MBARI Medusa (**SSDS ID 1441**): This has no XML and no current data, but I went back a couple days and found data that looks like:
    `-> channel=0, temp=48.51, vcc=3.2316, txBiasI=3.9796, txPower=0.4652, rxPower=0.0717, status=0x0 channel=3, temp=51.10, vcc=3.2252, txBiasI=4.4665, txPower=0.5313, rxPower=0.1337, status=0x0`
    iii. Medusa (**SSDS ID 1485**): Looks like the previous (might have actually replaced that one because it has current data).
    `-> channel=0, temp=51.69, vcc=3.2208, txBiasI=4.2839, txPower=-3.3320, rxPower=-10.6312, status=0x0 channel=3, temp=54.25, vcc=3.2142, txBiasI=4.7679, txPower=-2.7678, rxPower=-8.7681, status=0x0`
    iv. Digital Camera
        i. Thumbnail page of all images from the camera (click on image to get full image)
        ii. Sounds like images will come back as encapsulated JPG
c. CTD (SBE 16+)
    i.
        i. Plot raw data like on CIMT
        ii. Process data to calculate salinity
        iii. Plot salinity along with raw
    ii. Current Meter, ADCP, RDI Sentinel
        i. John R thought the processing would be very similar to the ADCP's we have seen on MTM and M0.
    iii. Current Meter, profiling, Aquadaopp HR Profiler, 6000m
        i. This is a complicated vector plot that will be calculated by matlab script. Assume we will just display the image
    iv. Flourometer/Backscatter, Wetlabs ECO BB2F triplett
        i. Unknown, but assuming this would be a similar process that Reiko is running for M0

v. Oxygen, Aanderaa
    i. Unknown
vi. Turbidity, Nephalometer, C-star, 6000m
    i. Unknown

d. MSE Axis BIN Medusa subnode 2 (**SSDS ID 1409**): Again, just message packets.
    i. MSP430 (**SSDS ID 1413**): Again, another MSP. Nothing new and has current data.
    ii. MBARI Medusa (**SSDS ID 1442**): No current data, but recent data looks like
    `-> channel=3, temp=49.40, vcc=3.2308, txBiasI=2.6144, txPower=0.5348, rxPower=0.2863, status=0x0`
    iii. Medusa (**SSDS ID 1486**):
    `-> channel=3, temp=50.25, vcc=3.2476, txBiasI=2.6434, txPower=-2.6995, rxPower=-5.3036, status=0x0`
    iv. Digital Camera
        i. Thumbnail page of all images from the camera (click on image to get full image)
        ii. Sounds like images will come back as encapsulated JPG
    v. CTD (SBE 16+)
        i. Plot raw data like on CIMT
        ii. Process data to calculate salinity
        iii. Plot salinity along with raw
    vi. Current Meter, ADCP, RDI Sentinel
        i. John R thought the processing would be very similar to the ADCP's we have seen on MTM and M0.
    vii. Current Meter, profiling, Aquadaopp HR Profiler, 6000m
        i. This is a complicated vector plot that will be calculated by matlab script. Assume we will just display the image
    viii. Flourometer/Backscatter, Wetlabs ECO BB2F triplett
        i. Unknown, but assuming this would be a similar process that Reiko is running for M0
    ix. Oxygen, Aanderaa
        i. Unknown
    x. Turbidity, Nephalometer, C-star, 6000m
        i. Unknown
    xi. Vertical Profiler, McLane(FSI velocity, Seapoint OBS, Wetlabs FL, CT&P)
        i. Unknown

e. Sediment Trap, Honjo
    i. Unknown
f. Sediment Trap, IRS
    i. Unknown

3. Outstanding Issues:
    a. Instrument questions:
        i. ASIMET LWR (John Ryan): Is there any post processing of the raw data and if so, what does it consist of?
        ii. ASIMET Radiometer, short-wave (John Ryan): Is there any post processing of the raw data and if so, what does it consist of?
        iii. ASIMET Relative Humidity and Temperature (Francisco Chavez): Is there any post processing of the raw data and if so, what does it consist of?
        iv. ASIMET Wind Speed Direction (Francisco Chavez): Is there any post processing of the raw data and if so, what does it consist of?
        v. Power Can (Ed Mellinger): Any new features/graphs
        vi. Load Cell(Lance McBride): Any processing, any new plots?
        vii. Seahorse(Andy Hamilton): Any processing, feedback or data plots from seahorse? Any linked images? Do you want processing and products tracked?
        viii. MBARI Medusa(Mark Chaffey): Do we want plots of any of this data? Any post processing (and resubmit/track)?
        ix. ASIMET Barometric(Chavez or Ryan): Do plots of raw data make sense? Any post processing (and resubmit/track)?
        x. All Satlantic Hyperspectral Radiometers (Chavez/Ryan): What/who is going to do the post processing. What does it look like and can we feedback/link to the results.
        xi. Digital Camera (Jim Barry, Mark Chaffey): is still an unknown as to what will be returned to SSDS. Mark sent request to Jim B for requirements from the camera and that will help answer the questions for SSDS.
        xii. CTD (SBE 16+) (Bill Ussler/Doug Conlin): Seabird software was identified as possibly doing the post processing, need to understand process in detail
        xiii. ADCP RDI Sentinel (Coenen): What is the post processing for this instrument? Can we feedback/link data products?

xiv. Aquadopp (Ussler): Contact Leslie to get post processing software.
xv. Wetlabs ECO BB2F Triplett (Ussler): Follow up to try and get example data to figure out what next steps are
xvi. Aanderaa Oxygen (Barry): Need to follow up to see what the data looks like and what post processing will be done.
xvii. Nephalomter (Ussler): Get details on post processing. Is this a SIAM instrument?
xviii. McLane Profiler(Hamilton): Get any details on post processing and if it is to link back to SSDS.
xix. Sediment Traps(Chavez): What will the data look like, are we going to put that in SSDS?

b. General Issues:
c. No decision has been made on MBARI's data policy for this experiment, currently the SSDS team is assuming all data will be open. The public access to data issue was discussed and the next step is to have the scientist's meet to come up with idea of how MSE data should be accessed by the public (if at all) and then give that feedback to the MT for review/approval (note the current default is that data that goes into SSDS is public)
d. Documentation in spreadsheet incomplete – we need to go to each of the reps and collect this data
    i. Don't know driver writers for many nominally SIAM instruments
e. Not sure what the summary/snapshot data (ADCP, OCR, Nortek, Weblabs) will look like and if we are supposed to do anything with it.

Related Data Management Systems and Projects

1. OpenIOOS.org
2. GEON
3. Earth Systems Grid
4. NOAA's OceanExplorer
5. GOMOOS
6. SeaMaven
7. SEACOOS
8. The COOL Room
9. EARTH Observations
10. Strategies.org
11. DAPPER (Look at DAPPER to serve SSDS data)
12. PACOOS
13. CENCOOS
14. EGY
15. OceanSITES
16. LDGO
17. Christoph Waldman (University of Bremen, Germany) (presented at MBARI on Jan 9, 2006)
    a. Sensor interoperability and standardisation
    b. ESONET I and ESONET II + ESONIM, KM2NET, ANTARES, ORION
    c. He is working on standardized data quality and semantics
    d. Quality Control: important to always state accuracy
    e. Using SensorML to describe the sensor and data from sensor (but does not address semantics)

Publishing other non-SIAM data to SSDS

This page last changed on Sep 30, 2009 by kgomes.

These are the instructions for putting non-SIAM infrastructure data streams into SSDS. There are two major steps for getting data into SSDS: Describing the data with metadata and Establishing a data publishing application.

Describe the deployments and data with XML metadata

> **Does not go through ingest**
>
> Note that this process does not send a JMS packet through the normal ingest chain. If you are SSDS savy, this means it will skip the transmogrify-ingest-ruminate step and put this information directly in the database.

1. Devices that are sensors (things that make measurements) and instruments (things that produce data) must first be entered into SSDS so that the metadata author can use the SSDS unique Device IDs in the XML metadata. This may be done with the newDevice.jsp application, specifically: http://new-ssds.mbari.org:8080/ssds/faces/newDevice.jsp.
2. Construct the XML describing the platform, instrument, and sensor deployment. Using an XML schema-aware tool such as Oxygen is recommended for producing well-formed and valid XML. Below is an example XML file (1696.xml) for the Test deployment of the Eye In The Sea platform. Important things to note:
   a. A Deployment with role="platform" must be the outer element.
   b. Give the platform Deployment an appropriate name - this will appear in the SSDS Explorer application and may be used to find the data in SSDS
   c. Other attributes (startTime, nominalDepth, nominalLatitute, nominalLongitude) may be added to the platform Deployment element, though they may be added later to the SSDS Metadata database
   d. Specify the bufferItemSeparator, recordTerminator, and recordParseRegExp in the instrument Deployment RecordDescription to enable automated parsing of the output
   e. Specify the RecordVariables (the minimal attributes are shown in this example)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Header: /home/cvs/puckxml/1696.xml,v 1.4 2008/12/18 20:19:43 mccann Exp $      -->
<!-- Last edited by $Author: mccann $   -->
<Metadata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://ssds.mbari.org/xml/schema/SSDS_Metadata.xsd"
majorVersion="1" minorVersion="2" lastAuthor="$Author: mccann $"
lastUpdate="$Date: 2008/12/18 20:19:43 $">
<Deployment role="platform" name="EITS on MARS (Test)">
<Device id="1697"/>
<!-- Eye In The Sea instrument for MARS2008 -->
<Deployment role="instrument" name="Eye In The Sea combined data from the CTD and ADV">
<Device id="1696"/>
<Deployment role="sensor">
<Device id="1694"/>
</Deployment>
<Deployment role="sensor">
<Device id="1695"/>
</Deployment>
<output>
<DataStream name="EITS Data Logger output of environmental data"
url="http://new-ssds.mbari.org:8080/servlet/GetOriginalDataServlet?deviceID=1696">
<RecordDescription bufferStyle="ASCII" bufferParseType="ordered"
bufferItemSeparator="whitespace" bufferLengthType="variable"
parseable="true" recordType="1" recordTerminator="\n"
recordParseRegExp="\s*([\-\d\.]+)\s*([\-\d\.]+)\s*([\-\d\.]+)\s*([\-\d\.]+)\s*([\-\d
\.]+)\s*([\-\d\.]+)">
<RecordVariable name="Temperature" longName="Water Temperature"
units="deg C" columnIndex="1" format="float">
<StandardVariable name="sea_water_temperature"/>
</RecordVariable>
<RecordVariable name="Salinity" longName="Salinity" units="psu"
```

```
columnIndex="2" format="float">
<StandardVariable name="sea_water_salnity"/>
</RecordVariable>
<RecordVariable name="Depth" longName="Depth" units="meters" columnIndex="3"
format="float">
<StandardVariable name="Depth"/>
</RecordVariable>
<RecordVariable name="CurrentDirection" longName="Current Direction"
units="degrees magnetic" columnIndex="4" format="float">
<StandardVariable name="direction_of_sea_water_velocity"/>
</RecordVariable>
<RecordVariable name="VerticalCurrentVelocity"
longName="Upward Sea Water Velocity" units="m/s" columnIndex="5"
format="float">
<StandardVariable name="upward_sea_water_velocity"/>
</RecordVariable>
<RecordVariable name="HorizontalCurrentSpeed" longName="Sea Water Speed"
units="m/s" columnIndex="6" format="float">
<StandardVariable name="sea_water_speed"/>
</RecordVariable>
</RecordDescription>
</DataStream>
</output>
</Deployment>
</Deployment>
</Metadata>
```

3. It's recommended that the final XML is to be checked into the puckxml module in MBARI's CVS on moonjelly.
4. Submit the Metadata to SSDS using the SSDSLoads application (available at http://new-ssds.mbari.org/ssds-docs/client/. The -h option provides a usage note):

```
java -jar ssdsLoads-new-ssds.jar -d 1696.xml -x      # Use '-x' to test and then '-s' to
submit
```

   a. If a mistake is made in the metadata (e.g. forgetting the platform Deployment) it may be easier to undo the submission and start over - do a deep delete on the parent deployment if this is the case.
   b. Minor attribute fixes may be done by editing the database once the deployments have been loaded
5. Check that the metadata has been successfully loaded using the Explorer application.

Establish data publishing application

1. This step assumes that there will be some application reading data from the deployed instrument. Perhaps it is a legacy application which reads the data to load into a custom data storage or visualization system. For this application to publish data to SSDS it must have visibility of each record the instrument produces as that record needs to be "packaged" into a SensorDataPacket and "published" to SSDS.
2. An example Java application that will package and publish a record is below:

```
import java.io.IOException;
import moos.ssds.jms.PublisherComponent;
import moos.ssds.transmogrify.SSDSDevicePacket;

/**
* <p>
* Publish instrument data records to the SSDS database. The client must provide
* SSDS device ID, timeStamp, sequence number, and payload.
* </p>
* <hr>
*
* @author : $Author: mccann $
* @version : $Revision: 1.17.2.7 $
*          <hr>
*          <p>
```

```
*          <font size="-1" color="#336699"><a href="http://www.mbari.org"> The
*          Monterey Bay Aquarium Research Institute (MBARI)</a> provides this
*          documentation and code &quot;as is&quot;, with no warranty, express
*          or implied, of its quality or consistency. It is provided without
*          support and without obligation on the part of MBARI to assist in its
*          use, correction, modification, or enhancement. This information
*          should not be published or distributed to third parties without
*          specific written permission from MBARI.</font>
*          </p>
*          <br>
*          <font size="-1" color="#336699">Copyright 2008 MBARI.<br>
*          MBARI Proprietary Information. All rights reserved.</font><br>
*          <hr>
*          <br>
*/

/**
* @author mccann
*
*/
public class SsdsPublisher {

/**
* Publish data record from an instrument to SSDS
*
* @param deviceID
*                       is the SSDS Device ID for the instrument that produces the data in
payload
* @param epochMilliseconds
*                       time of payload sample in milliseconds since 1/1/1970 0000 GMT
* @param sequenceNumber
*                       an incrementing number for each packet
* @param payload
*                       data from instrument
*/
public static void publish(long deviceID, long epochMilliseconds, long sequenceNumber,
String payload) {

// Create a publisher component
PublisherComponent pc = new PublisherComponent();

// Create a new SensorDataPacket
SSDSDevicePacket packetToSend = new SSDSDevicePacket(deviceID, payload
.getBytes().length);

// Set the packet type to data (0 = Metadata, 1 = Data, 2 = Message)
packetToSend.setPacketType(1);

// Assign the time
packetToSend.setSystemTime(epochMilliseconds);

// Set the parentID to 0 for a parentless deployment
packetToSend.setPlatformID(0);

// Set the metadataref number to 0, if the data format changes and we can
// refer to a different metadata packet then this number will change.
packetToSend.setMetadataRef(0);

// Set the sequence number
packetToSend.setSequenceNo(sequenceNumber);

// Set the payload
packetToSend.setDataBuffer(payload.getBytes());

// Set the record type to 1 as this is the most common situation
packetToSend.setRecordType(1);
```

```
try {
pc.publishBytes(SSDSDevicePacket
.convertToPublishableByteArray(packetToSend));
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}

/**
* Test of SsdsPublisher.publish()
*
* @param args
*            No arguments are taken.  Test values hard coded.
*/
public static void main(String[] args) {

/*
* Example packet for EITS instrument
*/
long eitsDeviceID = 1696; // 1696 is actual SSDS deviceID for EITS
long sampleTime = 1228431283209L; // Sample time: 2008-12-04 22:54:43
long sampleSequenceNumber = 1; // May set it to ID from DB insert

// Payload values must match the RecordVariables in SSDS
String samplePayload = "12.2, 4.3, 768.2, 2.3, 1.2, 0.2"; // Temp, Cond, Pres, U, V,
W

SsdsPublisher.publish(eitsDeviceID, sampleTime, sampleSequenceNumber, samplePayload);
}

}
```

3. An example of a shell script (the getM1 OASIS download) calling a perl script ssdsSubmit.pl to publish download statistics records to SSDS:

```
# Record start of download
set starttime_es = `perl -e 'print time, "\n"'`

# Download the data and record error status
$DOWNLOAD $DOWNLOAD_OPTS  >& $DATAFILE
set rtnsts = $status

# Record the end of download
set endtime_es = `perl -e 'print time, "\n"'`

# Get the raw data filesize
set filesize = `ls -l $DATAFILE | sed -f $BIN/fs.sed`

# Log download statistics and submit to SSDS - IDs are unique for mooring deployment
set deviceId = 1698
set parentId = 1306
/oasis/bin/ssdsSubmit.pl $deviceId $parentId "$starttime_es,$endtime_es,$filesize,$rtnsts"
```

4. If using Java incorporate the above class into a Java application that reads the data from the instrument and calls a method like the main() example for each record
5. Set up the application to execute the loads for the duration of the deployment
6. Configure DStoNetCDF.pl cron job execution as is done for the OASIS data processing within the ssdsadmin account on elvis. (For MARS deployments this is done for the DataProducerGroup MARS2008; see the processCurrentMooringDataStreams.sh script in /u/ssdsadmin/dev/DPforSSDS/cimt/).

Generate a packet from a file and publish to SSDS

Under Construction

This section is still under construction, pay no attention

1. Download a jar file
2. Create the properties file like the one here:

```
# This file contains the information that will be used to construct a data packet to publish
to SSDS.

# This is a SIAM property that (in theory) would allow multiple message formats
# that can be handled.  Right now, there is only one and it should always be 0.
DevicePacketVersion=0

# This is the ID of the source of the data (i.e. the device that generated the packet
# being sent.
SourceID=1

# This is the timestamp on the packet which normally represents when the packet
# was created.  In the system it is normally epoch seconds, but for clarity sake,
# you can enter it in ISO 8601 format (http://en.wikipedia.org/wiki/ISO_8601)
Timestamp=2009-09-30T15:47:00Z

# This is the sequence number that will be on the packet. It should reflect the
# index in the order of generation of packets from the source device.
SequenceNumber=1

# This is the reference number that points to the sequence number of the packet
# which contains the metadata that describes the contents of this packet.
MetadataRef=0

# This is the ID of the parent device (if one exists) that the generating device
# (see SourceID above) was connected to when it generated the packet
ParentID=0

# This is the type of record that is being produced.  Source devices can produce
# multiple types of records of the same type. For example, a device can produce
# data records in multiple formats.  This allows you to identify which format
# is used for the generated packet.  It serves to link the pre-defined metadata
# to the packet and allows machines and humans to understand the contents of the
# packet.
RecordType=0

# This value determines what type of packet (SIAM equivalent) will be constructed
# using the information in this file.  The available options are:
# 1. METADATA
# 2. SENSOR_DATA
# 3. DEVICE_MESSAGE
SecondStreamID=METADATA

# This indicates which version of the above type of packet will be constructed.
# At the time of this authoring, the only option available is 0 for all three
# types listed above. In theory this would allow you to have multiple forms of
# any of the above types of packets, but it has not been used to date.
SecondPacketVersion=0

# This is the name of the file (located with this properties file) that contains
# the payload of the first buffer.  It will be read as straight bytes into a
# byte buffer. NOTE: with METADATA packets the first buffer contains something
# that is called 'cause' which is not used much by SSDS.
FirstBuffer=test-first-buffer.txt

# This is the name of the file (located with this properties file) that contains
```

```
# the payload of the second buffer.  It will be read as straight bytes into a
# byte buffer. NOTE: with METADATA packets the second buffer is usually where the
# main metadata is stored (for example, device XML).
SecondBuffer=test-second-buffer.xml
```

3. Create the two buffer files
4. Run using command:

```
the command
```

Republishing Data From SIAM Node

This page last changed on Jan 13, 2009 by kgomes.

This is a quick note about how to republish packets from a SIAM Node. This was taken from and email that Kent Headley sent out after he republished some data to SSDS as well as some help from Bob Herlien.

1. Login to the machine where the siam service is running and the log files for the device that you want to republish data from
2. type 'gosiam'
3. cd to the directory where the data logs are
4. Run log publish as per the following example for device 1419

```
logPublish -start 1/5/2009T16:10:00 -stop 1/5/2009T18:00 1419 .
```

Services

This page last changed on Oct 28, 2009 by kgomes.

This is the index page that lists all the various services in the SSDS and links to their respective documentation:

1. Data Producer Services
    a. createDuplicateDeepDeployment
2. Data Services
    a. getDataStreamProperties

Data Producer Services

This page last changed on Jan 22, 2009 by kgomes.

createDuplicateDeepDeployment

Background

The purpose of this service was to have a way to make copies of deployments in SSDS. Often times we want to duplicate an instance of a deployment, but we want to copy all of its sub-deployment children as well, hence the "deep" part of the copy. This method takes in a <code>DataProducer</code> that must be of type "deployment" and then creates "deep" copy of it and persists the new copy. It pulls in the options specified in the incoming parameters to create a unique copy of the DataProducers and DataContainer (outputs). All the rest of the object should be linked to the ones that already exist in the peristent storage. The new ID of the duplicate deployment is returned.

The Interface

Return	Parameters
The ID of the newly generated deployment	• DataProducer to Copy • Date the copy should start at • A boolean to indicate if the old (original) should be closed • Date the old (original) should use as an end date (if to be closed) • String that will be the name of the new deployment (copy) • String that is the base of the data stream URI which should be `http://new-ssds.mbari.org/servlet/GetOriginalDataServlet`

So the API interface looks something like:
createDuplicateDeepDeployment

```
public Long createDuplicateDeepDeployment(
DataProducer deploymentToCopy,          // DataProducer to copy
Date newStartDate,                      // Date to start copied DataProducer on
boolean closeOld,                       // Whether or not old (original) should be closed
Date oldEndDate,                        // Date to use for end date if original is closed
String newHeadDeploymentName,           // Name of the new DataProducer (copy)
String baseDataStreamUri                // Base URI for raw data access (http://new-
ssds.mbari.org/servlet/GetOriginalDataServlet)
)
```

Java EJB client

Under construction

REST Style client

To use this service via HTTP (REST Style), the call would be constructed using this format:
REST Style format

```
http://new-ssds.mbari.org/servlet/MetadataAccessServlet?
objectToInvokeOn=DataProducerAccess&method=createDuplicateDeepDeployment
&p1Type=DataProducer&p1Value=DataProducer|id=1938292
&p2Type=Date&p2Value=2008-12-01T00:00:00Z
&p3Type=boolean&p3Value=true
```

```
&p4Type=Date&p4Value=2008-11-30T00:00:00Z
&p5Type=String&p5Value=New copy of Deployment
&p6Type=String&p6Value=http://new-ssds.mbari.org/servlet/GetOriginalDataServlet
```

And the return might looks something like this:

```
Result(s):
java.lang.Long|29985
```

Web Services client

Under construction

makePersistent

Background

The purpose of this service is to insert (or update) a DataProducer in the SSDS. If the DataProducer already exists (the equal criteria is by ID and then if no ID specified, it uses the outputs) it will update it, if not it will create a new one.

The Interface

Return	Parameters
The ID of the created or updated DataProducer	• DataProducer to persist

So the API interface looks something like:
createDuplicateDeepDeployment

```
public Long makePersistent(
DataProducer dataProducerToPersist         // This is the DataProducer that will be updated or
created
)
```

Java EJB client

Under construction

REST Style client

To use this service via HTTP (REST Style), the call would be constructed using this format:
REST Style format

```
http://new-ssds.mbari.org/servlet/MetadataAccessServlet?
objectToInvokeOn=DataProducerAccess&method=makePersistent
&p1Type=DataProducer&p1Value=DataProducer|name=MUCE - January 2009 test|description=MUCE Test
Deployment|dataProducerType=Deployment|startDate=2009-01-10T12:00:00Z
```

And the return might looks something like this:

```
Result(s):
java.lang.Long|29991
```

Web Services client

Under construction

Data Services

This page last changed on Jan 07, 2010 by kgomes.

searchDataStream

Background

A service was need to search data streams for some specific text. This service does not apply to all instruments as not all instruments output ASCII in their packets, but it was deemed handy to have for those that do output ASCII.

Parameter	Description	Options	Default Value	Required
Device ID	The SSDS ID of the device whose data stream the user wants to search	Any SSDS ID	N/A	Y
Expression	This is the regular expression (Java style) that will be used to search for	Any valid regular expression	N/A	Y
PacketOrLine	This is an option that specifies if the service should return the whole packet contents if a match is found, or if just the line out of the packet is found	*PACKET* or *LINE*	*PACKET*	N

The Interface

searchDataStream

```
// Packet or line enumerations
public final static String RETURN_PACKET = "packet";
public final static String RETURN_LINE = "line";

searchDataStream(
Long deviceID,                    // The ID of the Device whose data stream will be searched
String expression,                // The (Java style) regular expression to look for in the
// data stream
packetOrLine                      // The flag to indicate if the user wants the whole packet
// from the match returned, or just the line
)
```

The call will return a collection of **SSDSDevicePackets**

Java EJB client

Under construction

REST client

If you want to use the REST-style interface the HTTP call would looks something like:

```
http://new-ssds.mbari.org:8080/servlet/DataAccessServlet?
objectToInvokeOn=SQLDataStreamRawDataAccess&method=getDataStreamProperties
&p1Type=Long&p1Value=1300
&p2Type=Long&p2Value=1
&p3Type=Boolean&p3Value=true
&p4Type=Date&p4Value=2008-12-00T00:00:00Z
&p5Type=Date&p5Value=2008-01-00T00:00:00Z
&p6Type=String&p6Value=timeGap
&p7Type=Long&p7Value=5000
&p8Type=String&p8Value=userSpecified
&p9Type=Long&p9Value=10
&p10Type=Date&p10Value=
&p11Type=Date&p11Value=
&p12Type=Long&p12Value=10000
```

And the return might look something like

```
java.util.Properties{}
```

Web Service Client

Under construction

getDataStreamProperties

Background

The desire it to have a service that can characterize a DataStream from an instrument. This would allow for easier monitoring of instruments on the network.

Parameter	Description	Options	Default Value	Required
Device ID	The SSDS ID of the device that the user wants information about	Any SSDS ID	N/A	Y
Number Of Samples To Average	The number of samples back that the service is to use to calculate the average sampling interval	Any Number	10	N
Check for gaps	Is a flag that tells the service to try to find data gaps based on some average sample interval	True/False	False	N
Gap Interval	Is the number of milliseconds to use as the sampling interval to search for gaps. If this is specified it will override the average gap	Any number of milliseconds	N/A	N

	calculated by the service			

The Interface

The things that the user would want to know are:

Return	Parameters
Date and time of last packet received	• Device ID • RecordType to search for (nothing/default means most recent packet)
Total Number Of Records	• Device ID • RecordType to search for (nothing/default specified means all packets)
Data Gaps	• Device ID • RecordType • Time window over data to search for gaps • Gap criteria ◦ Type of gap - Time only - Sequence number only - Time and sequence number ◦ Ways to specify gap - Margin on gap in milliseconds (anything longer than gap + margin will be considered a possible gap) - Let service calculate gap constraints (calculate average time between sample) - Number of points to use (points back from most recent packet) - Or time window to use (start to end time) - Specify gap constraints - Gap in milliseconds

So the API interface looks like:
getDataStreamProperties

```
// Type of criteria to use for finding gaps
public final static String TIME_ONLY_GAP = "timeGap";
public final static String SEQ_ONLY_GAP = "seqGap";
public final static String TIME_SEQ_GAP = "timeSeqGap";

// The method of specifying the gap
public final static String SERVICE_CALCULATED = "serviceCalculated";
public final static String USER_SPECIFIED = "userSpecified";

getDataStreamProperties(
Long deviceID,                    // The ID of the Device to get the properties for
Long recordType,                  // The RecordType that will be singled out (devices
// can send out more than one RecordType)
// 0 = Metadata Packets
// 1+ = Device specific record types
Boolean checkForGaps,             // A Boolean that indicates if the caller wants to
// have the service check for data gaps (true means
// the service will check for gaps and false/null
// means it will not
Date startGapCheckWindow,         // The start date of the window over which to search for
```

```
// gaps.
Date endGapCheckWindow,            // The end date of the window over which to search for
gaps.
String typeOfGap,                  // One of three types: "timeGap", "seqGap", "timeSeqGap"
Long marginMillis,                 // This is the number of milliseconds that are used as
'slop'
// around the specification for a gap. In other words, if
// this is > 0, the service will consider any time between
// samples that is less than the specified gap plus this
margin,
// it will assume that it is not a gap condition.  This is
to
// prevent false positives when the sample timestamps
aren't exactly
// on the interval.
String gapSpec,                    // There are two ways to specify a gap:
// "serviceCalculated" or "userSpecified"
Long numberOfRecords,              // If the call specifies SERVICE_CALCULATED and this
// is greater than 0, the service will use
'numberOfRecords'
// most recent records of the specified RecordType
// in calculating the average time between samples
Date intervalCalcStartWindow,      // This is the date that starts the window over which the
data
// will be used to calculate the average time between
samples
// NOTE: If numberOfRecords is specified, this is ignored.
Date intervalCalcEndWindow,        // This is the date that ends the window over which the
data
// will be used to calculate the average time between
samples
// NOTE: If numberOfRecords is specified, this will be used
as
// the endtime and then the service will use the
numberOfRecords
// before this time as the data to calculate the average
// time interval.
Long gapInMillis                   // If the gapSpec is USER_SPECIFIED, then the service will
use
// this number of milliseconds as the gap for identifying
gaps.
)
```

With a return that has the format of:
Properties Objects with properties:

Property Name	**Value**
lastPacketDateTime	This is the date and time of the last packet received
totalNumberOfRecords	This is the total number of records for the parameters specified
numberOfFuturePackets	This is the number of packet that appear in the future. This should be zero and if they are not, there could be bad data
averagSampleIntervalInMillis	This is the number of milliseconds that the service used to find data gaps
marginInMillis	This is the number of milliseconds as a margin that the service used to find data gaps

numRecordsSearchedForGaps	This is the number of records that were searched through while trying to find gaps using the gap criteria
dataGap1Start	This is the date and time of start of the first possible gap in the data
dataGap1End	This is the date and time of end of the first possible gap in the data
.	.
.	.
.	.
dataGapNStart	This is the date and time of start of the Nth possible gap in the data
dataGapNEnd	This is the date and time of end of the Nth possible gap in the data

Java EJB client

Under construction

REST client

If you want to use the REST-style interface the HTTP call would looks something like:

```
http://new-ssds.mbari.org:8080/servlet/DataAccessServlet?
objectToInvokeOn=SQLDataStreamRawDataAccess&method=getDataStreamProperties
&p1Type=Long&p1Value=1300
&p2Type=Long&p2Value=1
&p3Type=Boolean&p3Value=true
&p4Type=Date&p4Value=2008-12-00T00:00:00Z
&p5Type=Date&p5Value=2008-01-00T00:00:00Z
&p6Type=String&p6Value=timeGap
&p7Type=Long&p7Value=5000
&p8Type=String&p8Value=userSpecified
&p9Type=Long&p9Value=10
&p10Type=Date&p10Value=
&p11Type=Date&p11Value=
&p12Type=Long&p12Value=10000
```

And the return might look something like

```
java.util.Properties{}
```

Web Service Client

Under construction

SQL 2008 Upgrade and Move to Dione

This page last changed on Jul 15, 2011 by kgomes.

In July of 2011, the database server that SSDS was running against was a an older version of SQL Server (2000) and SSDS was filling up the disk on Solstice. For this reason, it was decided to create a whole new server in the DMZ running SQL 2008 and to move the SSDS databases to that machine. Here are the steps taken during that work:

1. 7:41 AM: SSH'd into pismo as kgomes
2. 7:44 AM: edited the crontab using 'crontab -e' and commented out the entries that ran the data checker and the graphing routines.
3. 7:45 AM: Verified the updatebot was runnning using 'ps -ef'

   ```
   root      4203     1  0 Jul07 ?        00:00:15 /usr/java/jdk1.6.0_06/bin/java -
   Duser.timezone=UTC -Xms512m -Xmx1024m -classpath /opt/ssds/updatebot/ssds-updatebot-client-
   new-ssds.jar moos.ssds.clients.updateBot.UpdateBotRunner
   ```
4. 7:45 AM: stopped the updatebot service using 'sudo /sbin/service updatebot stop'
5. 7:49 AM: ssh'd into new-ssds.mbari.org as kgomes
6. 7:50 AM: in another windows, ssh'd into bob.shore.mbari.org as kgomes
7. 7:50 AM: verified JBoss was running on bob, using 'ps -aux' and getting:

   ```
   root   11926   0.4 -4.8  1325572 100336  ??  S    27Jun11 1313:48.61 /usr/bin/java -server -
   Xmx1024M -Djboss.server.temp.dir=/var/tmp/jbosstmpdata11922 -classpath /Library/JBoss/3.2/bin/
   run.jar org.jboss.Main -c deploy-standalone
   ```
8. 7:50 AM: on bob.shore.mbari.org, shut off the JBoss instance using:

   ```
   sudo serveradmin stop appserver
   ```
9. 7:54 AM: could not shutdown JBoss on new-ssds.mbari.org using 'sudo /sbin/service jboss stop' and figured out that I had to use:

   ```
   cd /opt/jboss/bin
   ./shutdown.sh -S -s 134.89.2.25
   ```
10. 8:16 AM: bob.shore.mbari.org had some oddities with iagadmin account, so I rebooted bob remotely using

    ```
    sudo shutdown -r now
    ```

    and then stopped the jboss service again.
11. I then copied all the pertinent files from bob.shore.mbari.org, new-ssds and pismo to my local machine so I could edit them locally.
12. While I don't think this is really an issue, all the editing of files that I do here is done with vi from the command line due to the fact that I am running on OS X and I don't want any editing software doing anything funny.
13. I decided that in order to keep my changes consistent with what was in the source code repo, I used the new-ssds.mbari.org build I had on my laptop and edited the new-ssds.mbari.org configuration properties and just ran the build again.

Testing TransmogrifyMDB and Ingest

This page last changed on Jan 04, 2010 by kgomes.

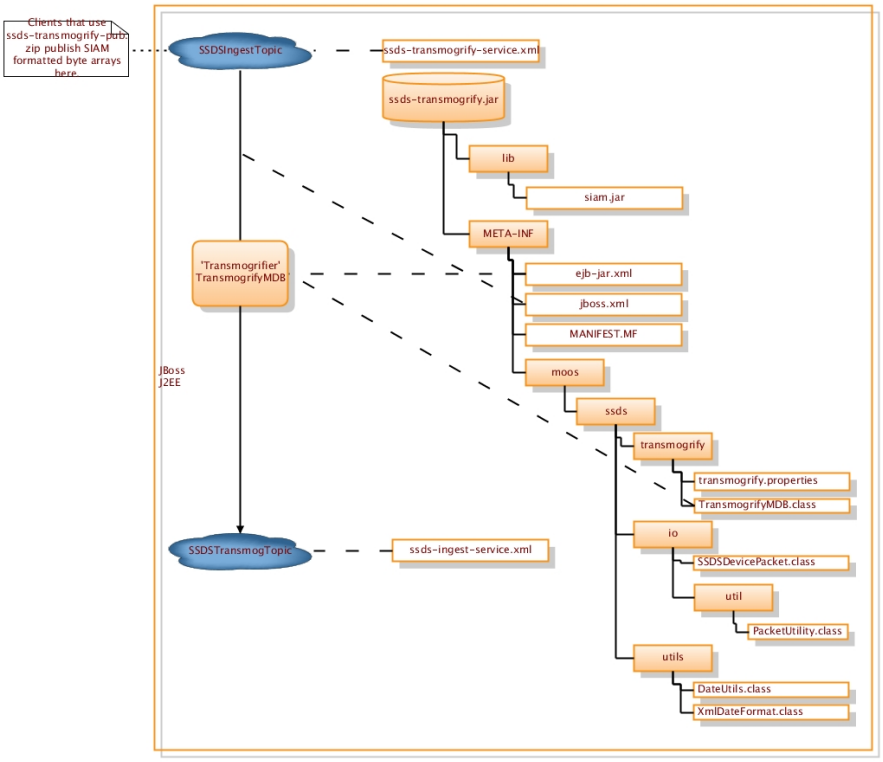
In order to test the TransmogrifyMDB class, it needs to be done in the context of a J2EE container. You can see how the TransmogrifyMDB is deployed in a J2EE container on
this page

Transmogrify and Ingest Deployment

This page last changed on Jun 08, 2010 by kgomes.

Transmogrify Component

The SSDS is a J2EE application that uses Java Messaging Service (JMS) to ingest data and metadata from clients. There are multiple stages of the ingest process. The first step is the Transmogrify service. This service configuration is shown here:



Java clients use a jar file that contains the JBossMQ client utilities and then publishes messages to the topic that feeds into the TransmogrifyMDB class. This MDB does some conversions to make sure the incoming messages are in a format SSDS can understand and then republishes them on to the topic that the IngestMDB is listening to.

Ingest Component

UserInterfaces

This page last changed on Jun 09, 2010 by kgomes.

This page contains information related to the design of the user interfaces for SSDS.

Graphical User Interfaces

Technology for the SSDS User Interfaces

I had been using Java Server Faces for the web application work and have been less than thrilled with it. It just is not that straightforward to do hard stuff. For this reason, I started to look around at RIA options. An evaluation of various technologies and how they are configured to work in a J2EE environment can be seen here:

RIA Technologies for the SSDS

Here is a list of the GUI/Pages that were developed to meet the user's requirements:

1. Device Inspector
2. DataContainer Importer

Programmatic Interfaces (APIs)

Matlab Integration

Users can interact with the SSDS programatically using Matlab. Instructions for doing this are found on the page Matlab 2008a Integration.

Other Resources

For a list of other resources used in designing user interfaces to the SSDS, please see Related Resources

DataContainer Importer

This page last changed on Jun 09, 2010 by kgomes.

The purpose of this component is to allow users to upload a file to the SSDS and have the SSDS parse the file as best it can and then present a form to the user so they can fill in the necessary metadata about the file.

Device Inspector

This page last changed on Nov 02, 2009 by kgomes.

This particular user interface is provided to the user so that they can explore a particular device to find out more about it and its current operation. Here is the information to be conveyed to the user by this interface:

1. Device Manufacturer
2. HTTP Links to external device information
3. Current location
4. Current parent
5. List of all deployments (user should be able to select deployment and show location on map)
6. XML Template for SSDS Metadata for deployment
7. Data streams coming from device (show variables, raw data)
8. Quick look plots of data
9. Statistics on data stream from instrument (gaps, last time heard from)
10. Associated metadata

Here is a mock up of what the device inspector might look like:

edit this mockup

Matlab 2008a Integration

This page last changed on Jun 17, 2008 by brian.

From Mike McCann:

> I thought I'd share this little bit of success.
> With Matlab 2008a on Windows we are able to interact with SSDS Metadata via the client jar file.
> The path to the jar file does need to be added to Matlab's static javaclasspath. Do this by adding the path to the jar file to C:\Program Files\MATLAB\R2008a\toolbox\local\classpth.txt, e.g. '$matlabroot/work/java/ssds-services-metadata-client-new-ssds.jar'. Then restart Matlab.
> Here's some example commands:
>
> ```
> % Import SSDS package
> import moos.ssds.services.metadata.*
>
> % Get Home interface
> home = moos.ssds.services.metadata.DataProducerAccessUtil.getHome();
>
> % Get Access object
> dpAccess = home.create();
>
> % Call methods on the Access object
> dpAccess.countFindParentlessDeployments
> dpAccess.countFindByName('Back', logical(0))
>
> % Get a specific instrument deployment and print out some properties
> dList = dpAccess.findByName( ...
> 'Backscatterometer (2006-10-17T00:53:30Z - 8) UUID=edb274f1-9277-11da-
> a72b-0800200c9a66', ...
> logical(1), 'id', 'ascending', logical(1));
> it = dList.iterator;
> d = it.next;
> d.getName
> d.getStartDate
> d.getDevice.getMfgSerialNumber
> ```
>
> (You may ignore the FileNotFoundException for \opt\ssds\logs\ssds-client.log - Kevin says he'll fix that.)

Related Resources

This page last changed on Oct 28, 2009 by kgomes.

1. **Data Search and Access** - This section focuses on finding (and maybe getting) the data. Within each category, the examples are roughly organized from more traditional to more innovative.
    a. MBARI's Cruise (expd) Interface
    b. MBARI's Samples Database
    c. Mike Godin's AOSN/MB06 interface for finding data via metadata
2. **External Oceanography Examples**
    a. SeaCOOS typical IOOS Regional Association site
    b. CaroCOOPS nice display of mooring sites
3. **External General Example**
    a. Google Maps points overlaid on lat/long (2 dimensions)
    b. Google Earth latest cool view of the world (2 1/2 dimensions)
4. **Data Visualization** - This section addresses interfaces for viewing the data.
5. Workflow/Automated
    a. Kepler project Project that can automate science data workflows, including visualizations

RIA Technologies for the SSDS

This page last changed on Oct 28, 2009 by kgomes.

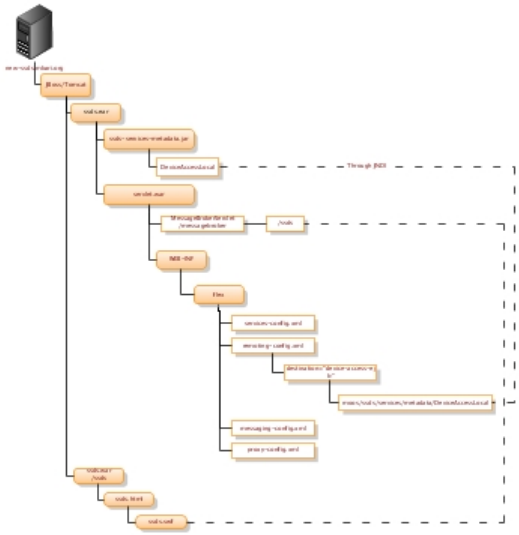
I have been using Java Server Faces for the web application work and have been less than thrilled with it. It just is not that straightforward to do hard stuff. For this reason, I started to look around at RIA options. Here are some:

1. Google Web Toolkit (GWT)
2. Flex 3 and BlazeDS

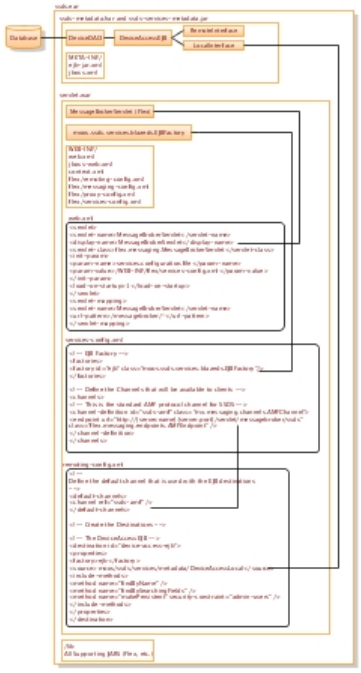
GWT

Flex 3 and BlazeDS

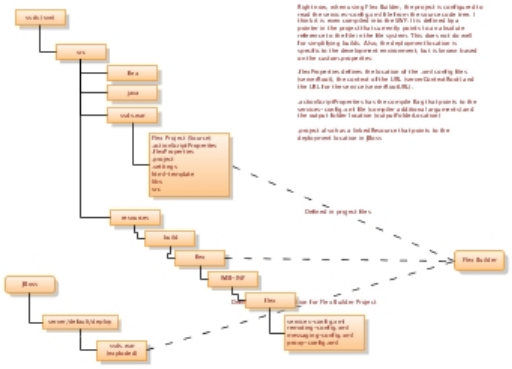
I looked at Flex 3 because I have seen some very compelling uses of it and it seems to integrate well with Java development (ant, J2EE, etc.). BlazeDS is a piece that goes on the server to expose Java objects as Flex services. So, for SSDS, we can expose the EJB's to flex clients by setting up the system in the following way:



A more detailed view with examples of configuration files can be seen here:



The development environment must work in both Ant and FlexBuilder. Here is the diagram that explains how all that works:



Tasks

This page last changed on Oct 21, 2008 by kgomes.

This list no longer updated

This task list is being kept for posterity sake and that it has some desired tasks that were shelved for a later date. The task that are currently being worked on are managed in the project plan located here

The current work (tasks) for SSDS are basically targeted at these main areas:

1. Cleanup Configuration Management
2. Internal Application Consolidation
3. Metadata Integrity Checking\-Repairing\-Enhancing
4. Improve Testing
5. Enhance Access Interfaces
6. Improve Data Ingest Mechanisms
7. Documentation
8. Prepare for Opening to Community

900823 SSDS Hardening Original Tasks List (Proposal)

1. Cleanup Configuration Management **(4 days - 3 KG, 1 MM)**
    a. (2 days) Finish build overhaul (done on Mac) to make sure SSDS can be built easily (common sense defaults) and that all components work with JBoss/MySQL combination.
    b. (.5 day) Setup javadoc deployment as part of build task
    c. (.5 day) Verify that wrapper generator unit test are on during test target of build.
    d. (.5 day) Create some template startup scripts and document
2. Internal application Consolidation **(10 days SE - 5 KG, 5 MM, 10 days I.S.)**
    a. (1 day) Look at SSDS_Metadata database for all urls that point to ssds.shore.mbari.org and see if we can point those to new-ssds or DODS.
    b. (1 day) Look at SSDS_Metadata database for all urls that point to dods.shore.mbari.org and see if we can point those to dods.mbari.org.
    c. (.5 day) Shutdown web server on predator (dods too).
    d. (.5 day) Look at all web applications on predator and see if we simply CNAME ssds.shore.mbari.org to new-ssds, that will work OK.
    e. (.5 day) Shutdown jboss on predator.
    f. (.5 day) Plan shutdown time for predator.
    g. (1 day) Backup files off predator. (should we migrate UpdateBot and Graphing stuff to elvis?).
    h. (0 day for SSDS-I.S. request) Have Pat upgrade predator to RHE.
    i. (.5 day) Reinstall updateBot and graphing software and restart.
    j. (2 day) Figure out what ssds.mbari.org is right now, move content to web application and CNAME ssds.mbari.org to new-ssds.mbari.org (or rename new-ssds.mbari.org to ssds.mbari.org and CNAME new-ssds to ssds.mbari.org)
    k. (.5 day - I.S.) Remove Microsoft SQL Server on SSDSPub
    l. (.5 day) Remove data directories on SSDPub
    m. (.5 day) Clean everything up and look at making SSDSPub just a Tomcat installation to house web applications
    n. (.5 day) Verify data replication from tornado to ssdspub is turned off (remove data from ssdspub too)
    o. (.5 day) Verify the GetOriginalDataServlet on ssds.shore and ssdpsub simply forward on to the same servlet on new-ssds
    p. (.5 day) Remove the SSDS database from Solstice (backup first)
    q. (.5 day) Remove the SSDS database from Fog (backup first)
    r. (.5 day) Backup and remove all DTS's except on Fog for Solstice-SSDS_Metadata->Fog-SSDS_Metadata
3. Prepare for opening to community **(3 days - 3 KG)**
    a. (.5 day) Put Copyright in all SSDS source code and zip up and make externally available.
    b. (2 days) Setup Source on public repository
4. Metadata Integrity Checking/Repairing/Enhancing **(33 days - 11 KG, 10 MM, 12 RS)**
    a. (3 days) Have updateBot check all resources and if they do not have a ResourceType, use the file extension to map it to a MIMEType and create a ResourceType (this will allow queries for specific resources)

b. (1 day) Have updateBot crawl all resources and update contentLength if not specified.
c. (10 days) Lookup standard variables and standardUnits from UCAR and have UpdateBot mechanism add them to those without (for NetCDF, try to pull StandardVariable and update SSDS).
d. (2 days) Implement a mechanism that updates the true end date/time on the DataContainer for data streams to show the latest data for each stream (should also update the number of records).
e. (5 days) Look into having SSDS create "README" type files in the same location as certain DataContainers.
    i. These could/should be in FGDC format
f. (12 days) Refactor and reinstate the SQL integrity checks Rich wrote.

5. Enhance Access Interfaces **(42 days - 20 KG, 22 MM)**
    a. (2 days) Develop web page to allow administrators to configure plot creation
    b. (5 days) Finish implementing all DAOs
        i. Make sure all methods have associated count method
        ii. Make sure all methods have boolean option for return full graph
        iii. Make sure all methods have capability to specify a sort by field
        iv. Verify returned DataContainer collections should be sorted by start date as default
        v. Verify implemented query for DataContainer by DataContainerGroup
    c. (.5 day) Verify PC02 plots are working after M0 turnaround
    d. (.5 day) Add links to CVS XML on device pages
    e. (1 day) Need a mechanism for users to upload and store "Resources" in SSDS (some secure/backed up network location) and link them to other resource aware objects.
    f. (.5 day) In Explorer, truncate long deployment names
    g. (5 days) Implement more queries in Explorer
        i. Desire in long term for common 'status reporting' system for instruments (something like an ops portal)
        ii. Find all post products from deployment
        iii. Find all resources of certain types (graphics, log files, calibration files, etc.)
        iv. "I want the Air Temperature (StandardVariable.name) from the DataProducer named "M1" from such and such to such and such a time."
        v. Given a deployment, can I have a query mechanism that can automatically walk up the tree to find the parent DataProducer of type Deployment?
        vi. Given a deployment, can I walk up the parent tree to look for a parent with a certain role?
        vii. "Give me all QC'd data and realtime data up to now". This would need to pull the best QC'd data for some sampled parameter and then integrate it with the realtime, un-QC'd data from streaming data.
    h. (5 days) Migrate HOOVES to new architecture
    i. (22 days) Add HOOVES improvements
        i. Full edit pages for deployment information
        ii. Tree structure for dataset variables that are functions of depth
        iii. SGT-driven contour plots of ZT data (e.g. ADCP, CTD String)
        iv. Faster variable list generation by using DODS rather than netCDF API
        v. More consistent use of resourceType contentType info (MIME types)
        vi. Top-level data set display for platform level deployment nodes
        vii. Additional queries:
            i. by standard variable name
            ii. by lat/lon rubber band box via mini maplet gui interface
        viii. Fix Bugs:
            i. Window sizing on startup
            ii. thread/hash problem with multiple plots
            iii. Numerics not showing for some data sets
6. Develop admin application to edit all metadata objects and their relationships **(11 days - 11 KG)**
    a. One function should be able to change the start time on a DataProducer and have an option to update all the child deployment (deep update) to that same start time.
    b. Build web pages that allow user to send messages to different topics in the ingest component
    c. Build web page that will allow a user to load data packets into the database from the file storage mechanism. (user selects device and time window).
    d. Replace instrument monitoring to read open deployments from SSDS and have configuration options.
7. Improve Data Ingest Mechanisms **(8 days - 8 KG)**
    a. (.5 day) Try to change OASIS to make mooring turns less painful (documentation basically)
    b. (5 days) Build non-JMS mechanism for users to send data/metadata to SSDS.

    c. (1 day) Put true exception handling in SQLIngest to throw messages when the DB can't be reached.
    d. (1 day) Change PacketSQLOutput/Input to work with any database (not just MS SQL)
8. Improve Testing **(5 days - 3 KG, 2MM)**
    a. (.5 day) Verify (unit tests) that the RecordDescription level parse regular expression works
    b. (.5 day) Make sure XML dates that are coming out of XMLBuilder/ObjectBuilder cycle can be parsed by the XMLDateFormat.
    c. (.5 day) Check Object Builder/XML builder to make sure it handles URI/URL/UriString/ DODSUrl/X/Y/Zoffsets correctly.
    d. (.5 day) Write valid unit test for Object and XMLBuilders
    e. (.5 day) Make sure object Schema/ObjectBuilder/XMLBuilder only use one DataContainer/ DataProducer for output, input, consumer, tags.
    f. (.5 day) Write tests for ResourceBLOB->ObjectBuilder for byte array and verify that it is working correctly.
    g. (2 days) Improve Wrapper tests
9. Documentation **(6 days - 3 KG, 3 MM)**
    a. (.5 day) Put UML diagram of data model on developer section of web app.
    b. (.5 day) Finish documenting data packet structure on web pages.
    c. (5 days) Document Explorer, Admin app and HOOVES

Tasks that were descoped from 900823 SSDS Hardening Proposal

1. Setup SPLUNK to monitor SSDS system logs (JBoss, Apache, UNIX system)
2. Can we setup SSDS so that if a certain process/data container were deemed bad, that you could prevent any reprocessing of that data using that process from happening?
3. Mike M had new idea about extending our model classes to implement remote lazy loading ... interesting idea and I would like to add that functionality later.
4. Can I embed the business logic documentation as JavaDoc tags in the source code and then export that somewhere? This is specifically related to the EJB services.
5. Add end of line terminator as separator in parsing packet records (not files)
6. Implement fixed format (length) parsing
7. Implement way to handle multi-record data
8. Allow user's to "annotate" DataContainer by storing comments
9. Design architecture for plug-in type functionality for servlet type access (i.e. inline filters for data like calculate salinity on the fly)
    a. Add a standard capability to SSDS to do things like mx+b, quadratics, using multiple variables in calculations (these should be able to be defined in the XML and SSDS do them automatically)
10. Refactor navigation menus and scheme for SSDS site and post processed pages (left menu)
    a. Need to add rest of SSDS web page content to main SSDS pages
11. Web pages to help with automated workflows
12. Create web page to do variable/variable plots (with some query capability)
13. Create page to merge data from different sources (could we use Francisco's help/resources:By Platform,By Instrument,By Variable,By Lat/Lon Box,map with AUV track and moorings/ instruments,data from AUV/Moorings/Ships merged by time (plots and raw data)
14. For application that show lots of graphics, implement thumbnails (click to enlarge)
15. SIAMRawDataAccess Page -> The list of instruments and parents doesn't show what is current. Maybe I can put a time/date of last update next to the instrument.
16. Build web form to help developers build URLs that they can use in their code to access data from SSDS
17. Somehow need to tell SSDS user's that they need to specify an acknowledgement of MBARI when users utilize the data from SSDS.
18. Implement mechanism (web page, email notification) that allows valid users to approve or map additional relationship and then notify the user of that change so they can change their source. This should be tied into UpdateBot so that it knows what associations it can make between RecordVariable and StandardVarible, for example.
    a. StandardVariables
    b. StandardUnits
    c. StandardKeywords
    d. StandardDomain
    e. StandardReferenceScale
    f. DeviceType
    g. ResourceType

    h. DataProducerGroup
    i. DataContainerGroup
19. GoogleMaps/GoogleEarth/Worldwind integration
20. Instead of using command line argument for timezone->UTC, look at [http://www.hibernate.org/100.html Hibernate code] for UTC dates.
21. Put statistics on plots (min/max of past 24 hrs, latest read value), use max, mins to drive axis range (no autoranging)
22. Where data are dense enough to make for easy viewing, use scatter instead of line plots, leaving gaps for missing data (if lines must be used, put breaks where data gaps are)
23. Setup DeviceQCPlotCreator to create plots with multiple lines on one chart (and separate axes).
24. Make any direction plot (wind, heading, etc.) plot as points, not lines
25. Put nominal lattitude and longitude in plot titles
26. Have capability to turn on/off autoscale on plots and specify range
27. Create servlet that dynamically generates schema and puts in tokens from class constants and database entries. For example, DeviceTypes, ResourceTypes, StandardXXXXXs
28. Build application to allow users to add QC flags and comments to data packets in SSDS_Data
29. Convert ingest to pull metadata revision number from XML instead of metadata sequence number from packet
30. Add mechanism to notify users of new data arrival (method TBD)
    a. Look into NRSS (RSS for data streams)
31. In PacketOutputManager, implement a cleanup watchdog (You could do this by keeping another hashmap that had the key deviceID_metadataRevision_recordType_platformID and then a value of last time accessed. Have the watchdog loop through those and PacketOutputs that have not been accessed in some interval get closed and removed from the packetOutputs hashmap)
32. Add software to crawl SQL packets, read in corresponding GPS data and attach to packet
    a. (5 days) Remove Deployment info from PUCK XML and move all to new schema and validate (due to the amount of work to do this, it will be done on an as needed basis)
33. Could we move applications on SSDSPub to another machine with Tomcat and CNAME ssdspub to that machine?
34. Follow up on PUCK configuration tool (ACE)
35. **Build services to read data from DataContainers that are files through the query interface (not just from packets).** (This is really valuable, but not REALLY needed)
36. Look into implementing paging in services (Hibernate supports this).
37. Load historical OASIS data into SSDS (data and metadata). **This is important but too big for this, separate project**
    a. Figure out how we will integrate data files offloaded from instruments when they are recovered. (generalize to any data file and this might be a location where the file can be place with appropriate metadata and it will be loaded into SSDS. SSDS can even do some thinking like with NetCDF files). **IMPORTANT EVEN THOUGH DESCOPED**

Cleanup Configuration Management

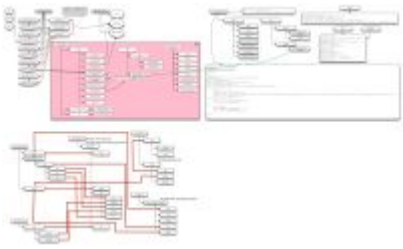
This page last changed on May 30, 2008 by kgomes.

The idea of this work was to cleanup the build process so that it was very manageable and had a great number of common sense defaults setup so that the property configurations were not difficult. This would allow for easier installation as well as development. When I was doing this work, I setup the build so that the user could more easily switch between MySQL and MSSQL. So a large part of this work over-lapped with the work to get the system ready for opening to the community.

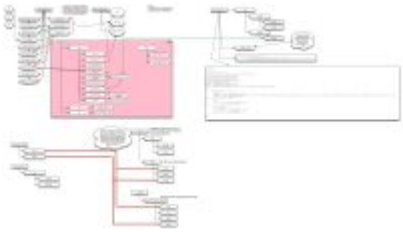
Internal Application Consolidation

This page last changed on Jun 09, 2008 by kgomes.

In order to get our local (MBARI) installation of SSDS in a manageable state, I went through an application consolidation phase to try and clean up a bunch of stuff. The first thing to do was to create a layout of how things are now.



And then a diagram of how they will look after the cleanup:



And then the steps on how to make that transition:

ssdspub.mbari.org

The easiest place to clean first, was the machine ssdspub.mbari.org. Currently it is basically just serving the purpose of a tomcat container. There are still services out there, but they are not really serving any purpose since they are pointed to a database that is defunct. To clean up, I did the following:

1. I first removed the axis.war file from the deploy directory.
2. I then removed the omse.war and the mse.war web applications.

> **Move MSE to the inside?**
>
> I am wondering if I shouldn't move the mse.war pages to the new-ssds.mbari.org server so they are at least available.

3. I then shutdown Jboss, removed access.war, ssds-data-mssql-ds.xml, ssds-mssql-ds.xml and ssds-services-ssdspub.jar

> **access.war wasn't so simple**
>
> When I removed access.war, it messed up some people who were using the old GetOriginalDataServlet and the forwards from the old /access/*.jsp's were broken. I put an access.war back out there, but removed the servlets and put notes on the other pages that said either the pages were no longer available or where they could go to get to them.

4. I then deployed access.war and cimt.war on to new-ssds.mbari.org (to prepare for the CNAME change)
5. I then restarted JBoss
6. I also updated the index.html page in the apache installation to point to the cimt web application so that if people go to ssdspub.mbari.org they will see something.
7. I had Neil shut off the replication jobs that were rebuild the SSDS database on ssdspub each day.
8. I also had Todd and Neil shut off the replication jobs that were copying the raw data files from bob.shore.mbari.org, iagdata share on tornado, and the ssdsdata share on tornado out to SSDSPub as they are no longer needed.

9. I then set the MSSQLServer and SQLServerAgent service to 'Manual' and shut them off.

> ⚠ **Get rid of SSDSPUB?**
>
> In theory, I should now be able to remove ssdspub.mbari.org if I CNAME it to new-ssds.mbari.org

predator.shore.mbari.org

1. Next, I could do a similar cleanup of predator.
2. First, I removed axis.war
3. Then I removed mtm3.war
4. Now, my current thinking is that instead of going through the database and changing everything under the sun, can I just change the CNAME of ssds.shore.mbari.org to point to new-ssds.mbari.org. In order to do that, I need to:
    a. Change all references from predator.shore.mbari.org to ssds.shore.mbari.org in DataContainer.uriString, Resource.uriString and Software.uriString and make sure those entities exist.
        i. First I queried to find all the DataContainers with predator in their URIString. I got back 23 rows of DataContainers whose uriStrings are no longer valid. Since this is the case, there will be no harm in just changing them with the following SQL:

```
UPDATE ssdsdba.DataContainer SET uriString = REPLACE(uriString, 'predator.shore',
'ssds.shore') WHERE uriString like '%predator.shore%'
```

        ii. Next thing was to do it for the Resources. Now, here there was a small snag. Some of the old NetCDF logs have an analogous entry for ssds.shore already so when the update was tried, I got duplicate unique key constraint violations. So, first I just searched for entries that pointed to the ssds/xml directory.

```
SELECT * from ssdsdba.Resource where uriString like '%predator.shore.mbari.org/ssds/
xml%'
```

            This returned 47 rows and they seemed to be valid uriStrings even though they were from really old stuff. So, I simply changed the uriString to point to ssds.shore instead of predator with the following:

```
UPDATE ssdsdba.Resource set uriString =
REPLACE(uriString,'predator.shore','ssds.shore') where uriString like
'%predator.shore.mbari.org/ssds/xml%'
```

        iii. After that, I queried for the other resources with predator in the name using:

```
SELECT * from ssdsdba.Resource where uriString like '%predator.shore%'
```

            and it returned 24 rows of things that do not exist. Since they don't exist at the uri's and renamed hit unique key constraints, I just decided to remove them by first removing references to them in the assocResource tables.

```
select * from ssdsdba.DataContainerAssocResource where ResourceID_FK in (select id
from ssdsdba.Resource where uriString like '%predator.shore%')
select * from ssdsdba.DataProducerAssocResource where ResourceID_FK in (select id
from ssdsdba.Resource where uriString like '%predator.shore%')
select * from ssdsdba.DeviceAssocResource where ResourceID_FK in (select id from
ssdsdba.Resource where uriString like '%predator.shore%')
select * from ssdsdba.SoftwareAssocResource where ResourceID_FK in (select id from
ssdsdba.Resource where uriString like '%predator.shore%')
```

            The only one that found anything was for DataProducers (48 rows), so I removed all assoc records using:

```
delete from ssdsdba.DataProducerAssocResource WHERE ResourceID_FK in (select id from
ssdsdba.Resource where uriString like '%predator.shore%')
```

            Now that all the links to the resources with uriStrings with predator are removed, remove the resources themselves with:

```
delete from ssdsdba.Resource WHERE uriString like '%predator.shore%'
```

That removed 24 rows

iv. There were no uriStrings in the Software table that have references to predator.shore, so I did not do anything

b. Now that the predator name has been removed from the uriStrings, let's make sure there are no dods references in the uriStrings. I can search for those using:

```
SELECT * from ssdsdba.DataContainer where uriString like '%nph-dods%'
```

That returned a whopping 1590 records, but there are basically two roots of the URLs that are of importance, they are:

```
http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/data/
```

and

```
http://ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd/
```

Since the auvctd ones are mapped through to the auvctd share on Tornado and the dods.mbari.org auvctd is the same, we can simply map the ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd to the dods.mbari.org machine using

```
UPDATE ssdsdba.DataContainer set uriString = REPLACE(uriString,'http://
ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd','http://dods.mbari.org/cgi-bin/nph-nc/data/
auvctd') where uriString like 'http://ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd%'
```

Since the rest of the DataContainers that have uriStrings with nph-dods in them are pointing to old data and I can't rename them (they would create duplicate uriStrings because we used to put parallel dods and http file uris in there), I am just going to let them be and have broken links (for now). So there are 1255 records like that with broken links.

c. Verify all DODS urls are accessible through dods.mbari.org
   i. Currently, here is the list of DODS URLs that are available through ssds.shore.mbari.org:
      i. http://ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd/ (which is the mount of AUVCTD on Tornado)
      ii. http://ssds.shore.mbari.org/cgi-bin/nph-dods/clients/ (which is a broken link)
      iii. http://ssds.shore.mbari.org/cgi-bin/nph-dods/data/ (which is the mount to the data volume on bob.shore.mbari.org).
      iv. http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/data/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/data/)
      v. http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/rawpackets/ (which is a link through the 'data' mount to the rawpacket on bob.shore.mbari.org)
      vi. http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/rss/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/rss/)
      vii. http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/transmogrify/ (which is a link through the 'data' mount to the transmogrify directory on bob.shore.mbari.org)
      viii. http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/xml/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/xml/)
   ii. Let's look at these on a case-by-case basis
      i. The AUVCTD mount on ssds.shore is the same as the one on dods.mbari.org. So the following URLs should be equivalent:

         ```
         http://ssds.shore.mbari.org/cgi-bin/nph-dods/auvctd/
         ```

         equals:

         ```
         http://dods.mbari.org/cgi-bin/nph-nc/data/auvctd/
         ```
      ii. For the clients URL, since it is broken, there is no equivalent
      iii. For the /data which is a mount to bob.shore.mbari.org, there is no equivalent URL on dods.mbari.org. That might be fine, we will find out in a minute.
      iv. The /ssds/data URL on ssds.shore points to the ssds share on iagdata which is accessible through dods.mbari.org from the /data/ssds share. So these are equivalent:

         ```
         http://ssds.shore.mbari.org/cgi-bin/nph-dods/ssds/data
         ```

equals:

```
http://dods.mbari.org/cgi-bin/nph-nc/data/ssds/
```

There is a problem though that on the dods.mbari.org side, there is a permissions denied in trying to access it. However, I don't think we really need this share and it would be nice to remove it if possible.

v. That last one also applies to the rss and xml directories
vi. The ssds/rawpackets and transmogrify urls point to the raw packet and transmogrify share on bob and is not available through dods.mbari, but that should be OK. I will find out shortly.

iii. Now that we have an idea of how they are mapped, let's take a look at the DataContainer's and their base uriStrings to see if they point to any nph-dods urls. Since these are the same broken linked files that I found above and they cannot be mapped due to duplicate uriString constraint, I will just leave the uriStrings for DataContainers alone.
iv. For the DataContainer dodsUrlString, I can query to find any current dods urls that point to ssds.shore using:

```
select * from ssdsdba.DataContainer where dodsUrlString LIKE '%nph-dods%'
```

Since this returned no results, we should be fine on the data container side of things (I think we did that move earlier).

v. We need to do the same for any resources we find and search the uriString for nph-dods:

```
select * from ssdsdba.Resource where uriString LIKE '%nph-dods%'
```

Which returned no results so we are good there.

vi. Also check software

```
select * from ssdsdba.Software where uriString LIKE '%nph-dods%'
```

Which also returned no results.

d. Now, we have all nph-dods urls that point to ssds.shore removed (except for the broken 1255) and a CNAME change should work if we point ssds.shore to new-ssds. Before we do that though, we must make sure all HTTP accessible shares on predator are available on new-ssds at the same base URL (i.e. new-ssds.mbari.org/ should be the equivalent of ssds.shore.mbari.org from an HTTP directory sharing standpoint. So, the following HTTP shares are available on ssds.shore:
    i. http://ssds.shore.mbari.org/auvctd/ (which is the mount of AUVCTD on Tornado)
    ii. http://ssds.shore.mbari.org/clients/ (which is a broken link)
    iii. http://ssds.shore.mbari.org/data/ (which is the mount to the data volume on bob.shore.mbari.org).
    iv. http://ssds.shore.mbari.org/ssds/data/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/data/)
    v. http://ssds.shore.mbari.org/ssds/rawpackets/ (which is a link through the 'data' mount to the rawpacket on bob.shore.mbari.org)
    vi. http://ssds.shore.mbari.org/ssds/rss/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/rss/)
    vii. http://ssds.shore.mbari.org/ssds/transmogrify/ (which is a link through the 'data' mount to the transmogrify directory on bob.shore.mbari.org)
    viii. http://ssds.shore.mbari.org/ssds/xml/ (which is a link through a mount to tornado.shore.mbari.org/iagdata/ssds/xml/)
e. So if we look at them one-by-one:
    i. http://ssds.shore.mbari.org/auvctd/ does not have an equivalent on new-ssds, but I have a trouble ticket into I.S. to get that mounted.
    ii. http://ssds.shore.mbari.org/clients/ since it is a broken link, I am not worried about making it available through new-ssds.
    iii. http://ssds.shore.mbari.org/data/ I am hoping to not have any links pointing to this, so hopefully I can not make that share available.
    iv. http://ssds.shore.mbari.org/ssds/data/ I am hoping I can get rid of this
    v. http://ssds.shore.mbari.org/ssds/rawpackets/ I am hoping I can get rid of this
    vi. http://ssds.shore.mbari.org/ssds/rss/ I am hoping I can get rid of this
    vii. http://ssds.shore.mbari.org/ssds/transmogrify/ I am hoping I can get rid of this
    viii. http://ssds.shore.mbari.org/ssds/xml/ I am hoping I can get rid of this

f. So let's start with the DataContainer uriStrings (I am going to ignore DODS URLs since they were done). I ran the following search:

```
select * from ssdsdba.DataContainer where uriString LIKE 'http://ssds.shore.mbari.org/auvctd%'
```

I get 4692 results. As long as I can get the auvctd mount working on new-ssds, the CNAME should fix these.

```
select * from ssdsdba.DataContainer where uriString LIKE 'http://ssds.shore.mbari.org/clients%'
```

This returned 0 results, so we are good to get rid of it.

```
select * from ssdsdba.DataContainer where uriString LIKE 'http://ssds.shore.mbari.org/data%'
```

Again, 0 results.

```
select * from ssdsdba.DataContainer where uriString LIKE 'http://ssds.shore.mbari.org/ssds/data/%'
```

Returned 1278 entries. All the other /ssds/* urls returned nothing so we are good there. Looking at the Resource table, it looks like there are uriStrings that point to /ssds/data and /ssds/xml, but they all look very out of date. There were no uriStrings in the Software table that pointed to the ssds.shore url so we are good there. So the big question becomes can we just remove all references to those old shares from the metadata since I think most of those have been reprocessed anyway? I have contacted Mike McCann about it. If that is the case I can get rid of:

i. ssds share/url on dods.mbari.org
ii. All the dods and http share/urls from ssds.shore.mbari.org
iii. All of the data housed in the iagdata/ssds share on tornado
Great, got the OK from Mike, so I can do all this and then we can go back and clean out the DB of any DataContainer, DataProducers, and Resources that are associated with these URLs. COOL!
iv. One small change, there are a handful of Resources that are XML files for data streams. Those might be useful, so I could copy those over the current Ruminate xml share on new-ssds and update the URLs to point to them there. Actually it looks like they have already been copied, probably when I moved to new-ssds, so I just need to update the URLs with:

```
UPDATE ssdsdba.Resource set uriString = REPLACE(uriString,'http://ssds.shore.mbari.org/ssds/xml','http://new-ssds.mbari.org/data/ssds/ruminate/xml') where uriString like 'http://ssds.shore.mbari.org/ssds/xml%'
```

5. OK, now with that all cleaned up, I needed to move the existing updatebot, graphing and data checking perl script to the machine

```
pismo.shore.mbari.org
```

a. Pat was able to construct the same directory structures and permissions on pismo as on predator, so I just copied the files over to /opt/ssds and changed the scripts to point to the correct java locations.
b. With the data checking perl script, I changed it to point to the new-ssds GetOriginalDataServlet so that it would be reading the packets from the database and is a much better test as we can see the packets at the database and not just the file system.

6. Now with all things moved to pismo, we should be able to change the CNAME of ssds.shore.mbari.org to point to new-ssds.mbari.org instead of to predator.shore.mbari.org. Pete made that change and I shutdown the jboss (ssds) service and the httpd service on predator. I also commented out the various mounts in /etc/fstab so that the shares would no longer be mounted on predator.

new-ssds.mbari.org

1. Now, I currently have ruminate running on new-ssds as a message driven bean that is writing the XML files to a local directory /data/ssds/ruminate/xml. This really should be stored on the /

tornado.shore.mbari.org/ssdsdata/ssds/ share under something like: /tornado.shore.mbari.org/ssdsdata/ssds/ruminate/xml. This means that I need to get a read-write share mounted from /tornado.shore.mbari.org/ssdsdata/ssds/ruminate that I can mount on new-ssds. If I can do this, I can then point any urls to the /ssdsdata/ssds/ruminate url on new-ssds and turn off the http share to the local /data/ssds/ruminate directory. The same goes for the /data/ssds/generated/gps directory.

a. There were security concerns (rightly so) about setting up a write share through the firewall, so instead, we setup a copy to run every 10 minutes and copy all files from the /data/ssds/ruminate/xml to the tornado /ssdsdata/ssds/ruminate/xml directories. Also, a similar copy was setup for /data/ssds/generated/gps. This means that any URLs that used to point to:

   ```
   http://new-ssds.mbari.org/data/ssds/ruminate/xml
   ```

   should point to

   ```
   http://new-ssds.mbari.org/ssdsdata/ssds/ruminate/xml
   ```

   and

   ```
   http://new-ssds.mbari.org/data/ssds/generated/gps
   ```

   should point to

   ```
   http://new-ssds.mbari.org/ssdsdata/ssds/generated/gps
   ```

   The SQL for that to happen is:

   ```
   UPDATE ssdsdba.Resource set uriString = REPLACE(uriString,'http://new-ssds.mbari.org/data/ssds/ruminate/xml','http://new-ssds.mbari.org/ssdsdata/ssds/ruminate/xml') where uriString like 'http://new-ssds.mbari.org/data/ssds/ruminate/xml%'
   ```

   and

   ```
   UPDATE ssdsdba.Resource set uriString = REPLACE(uriString,'http://new-ssds.mbari.org/data/ssds/generated/gps','http://new-ssds.mbari.org/ssdsdata/ssds/generated/gps') where uriString like 'http://new-ssds.mbari.org/data/ssds/generated/gps%'
   ```

   The second query was not necessary as it did not have any entries. After running those, I rebuilt the ssds-ruminate.jar with the updated url bases and deployed to new-ssds. Since this effectively removes all need of the http://new-ssds.mbari.org/data link, I removed that share from the http server on new-ssds as well.

   While I was there

   While I was updating ruminate, I changed the jboss.xml that deploys with ruminate and changed the entry:

   ```
   <MaximumSize>15</MaximumSize>
   ```

   to

   ```
   <MaximumSize>1</MaximumSize>
   ```

   Which should effectively make the RuminateMDB a singleton which should alleviate our deadlock issues that we were having (at least at Ruminate step).