# Space Details

| | |
|---|---|
| **Key:** | OMF |
| **Name:** | 708763 Observatory Middleware Framework |
| **Description:** | |
| **Creator (Creation Date):** | kgomes (Jun 09, 2008) |
| **Last Modifier (Mod. Date):** | kgomes (Jun 09, 2008) |

## Available Pages

- OMF Project Space
  - Build And Run SENSORS CTD
  - MRG Development

This page last changed on Nov 25, 2009 by kgomes.

## Abstract and Proposal

Proposal

## Goals of the Project

1. Federate data streams from multiple observatories
2. Create common instrument access and management
3. Provide fundamental mechanism to enforce policies
4. Captures and forwards rich metadata ~~including data provenance~~
5. Provide mechanism to handle data and events in a uniform way
6. Ability to readily integrate and configure instruments, data streams, processing and storage resources with indifference to resource implementations
7. Uniform interface to observatory resources
8. Service integration framework

## Approach

1. Integrate existing sensors, services, and grid functionality using an Enterprise Service Bus (ESB), specifically Mule.
2. Three functional areas where this integration occurs:
    a. Instrument Access and Management
    b. ~~Stream Processing~~
    c. Services (Specifically Governance ~~and Provenance~~)
3. Implementation will be validated using bench tests and through pilot implementations on MOOS and MARS
4. Will use the integration of SIAM, ROADNet, and direct instrument as an example
5. Observer can view and subscribe to data streams
6. Provide a comprehensive suite of functional capabilities integrated across a complete range of installed systems (including simulated systems).
7. We will provide benchmarks
8. Develop specification and implementation of standard instrumentation interfaces between the Sensor Enterprise Service Bus and the Instrument Proxy, including the complete semantics of the message exchange between those services
9. Develop specification of a set of "standard instrument" capabilities that an Instrument Proxy will represent and that can be mapped to the functionality of either specific instruments or observatory representations of its instruments
10. Develop specification of the stream protocols that will carry data, and stream-characterizing metadata, for use by applications
11. Provide evaluation and adoption of a suitable content standard for description of data sources and data provenance, enabling plug-and-play functionality and rich metadata integration of data streams
12. Adopt suitable vocabularies for use within the content standards described above
13. A generic Instrument Proxy that bridges the SESB protocols to a set of typical instrument capabilities
14. Observatory-specific Instrument Proxies that bridge SESB protocols to instrument protocols as presented by existing observatories like MOOS (via SIAM) and US Array (via ROADNet)
15. Modification of the instrument protocols presented by existing observatory infrastructures (like SIAM and ROADNet) to provide the capabilities expected by observatory-specific Instrument Proxies.
16. Governance will be done by controlling the flow of messages and data streams and will be done using:
    a. Internet2 Signet work
    b. Sun XACML implementation
17. Authentication and identity management will be done with integration of technologies from:
    a. Grid Security Infrastructure
    b. Sibboleth (using Kerberos/LDAP)
    c. GridShib (using Kerberos/LDAP)
    d. GridGrouper
    e. JAAS (for Mule ESB)
18. For client tools, we will look at using:
    a. Nagios for hosts and network services

b. Ganglia for monitoring high-performance computing systems
c. ~~StreamBase stream processing engine~~
d. ~~Coral8 Engine for event detection and processing~~

## Requirements

1. Correlate data readings from multiple sensors
2. Establish relationships between the various correlated data
3. Scientist can discover data despite naming differences
4. Immediate access to real-time data streams from different observatories
5. Stream data processing
6. Respond to events by:
   a. increasing or altering data acquisition rates
   b. taking operational actions to ensure robust data collection
   c. allocating additional sensors to take extra observations to characterize the anomaly
7. Promote collaboration among researchers
8. Provide common mechanism for managing data streams
9. Provide common mechanism for automating the creation of derived data from data streams
10. Provide mechanisms for integration of data and models
11. Provide support for workflow automation
12. Create and operate virtual observatory

## Architecture

## MBARI Statement of Work

This statement of Work defines the effort required to demonstrate instrument federation using an ESB. MBARI's work is to

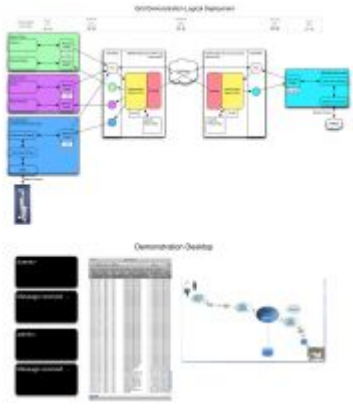
1. build instrument control and data access procedures as part of the MBARI SIAM (Software infrastructure and application for MOOS) infrastructure implementing agreed to standard protocols
2. implement the necessary protocols in low level hardware that resides with individual instruments
3. implement the documented reference implementation on ROADNet/Antelope.

#### Detailed tasks

1. Build up InstrumentProxyStrategy commands for instrument control (e.g. turn on/off, etc.).
2. Augment the InstrumentProxy classes to handle the instrument control commands.
3. Perform the SIAM and ROADnet integration to handle those commands.
4. Document the reference implementation.
5. Develop software to run InstrumentProxy on Digi Connect to demonstrate plugging a device directly onto the ESB.

## InGrid Demonstration

One target for this project was to develop a demonstration that could be presented at the InGrid conference. Here is a logical



## Related Resources

1. OMF Wiki
2. Meeting Notes
3. Demo Plans
4. Post-Demo Thoughts
5. OMF Mule Architecture
6. OMF Project Kickoff

## Development Notes

1. SVN URL (you need an account and permissions): https://svn.ncsa.uiuc.edu/svn/omf
2. Build And Run SENSORS CTD

## MRG Project

As a side project for the OOI CyberInfrastructure Team, we worked on moving the code for utilizing the ESB to the QPID messaging framework that is running under RedHat's MRG software. You can find that work documented here:

MRG Development

## Build And Run SENSORS CTD

This page last changed on Jul 08, 2009 by kgomes.

The goal of the MBARI portion of the OMF project is to get an instrument connected to the Enterprise Service Bus. In order to do that, we setup a PC-104 stack, running Debian Linux that was then connected to a Sea-Bird SBE 37-SM. We ran SIAM as the instrument service and then connected SIAM up to the ESB using messaging. To get the system running:

1. Put the PC-104 stack together and install Debian. Please refer to the FOCE documentation on how this was done
   a. FOCE Electronics Page
   b. Installing Debian on PC104
2. Check out SIAM2 code base from CVS under the ops account
3. Check out the omf code base from NCSA's repository
4. Created a new device in the Shore Side Data System and retrieved the ID from SSDS.
5. Edited the ~/siam2/make/Makefile and added a section to define a puck jar for this device

```
omfpuck:
mkpuck org.mbari.siam.devices.seabird.sbe37.SBE37 'Seabird-37SM s/n 37SM34119-3374' 1701 \
$(JAVA_DEV_ROOT)/ports/Seabird-1701.jar $(JAVA_DEV_ROOT)/puckxml/1701.xml 'sampleSchedule =
600' \
'powerPolicy = always' 'commPowerPolicy = sampling' 'defaultSkipInterval = -1' \
'safeSampleIntervalSec = 600' 'currentLimitMa = 3000' 'timeSynch=true' 'log=false' \
'UUID = d691724b-ebef-11dd-98dd-f9ada8061848';
```

6. Checked out the puckxml project from CVS
7. Created a 1701.xml file in that project and modeled it after another SBE-37 XML doc (1613.xml)
8. Checked that file back into CVS.
9. Create a ~/siam2/puckxml directory
10. Copied the new XML file (1701.xml) to that directory
11. When to the command line and ran 'make' (took a long time to build everything).
12. Then ran 'make omfpuck' (this creates a Seabird-1701.jar in the ports directory)
13. I then created a siamPorts.cfg file in the properties directory
14. I then typed 'gosiam' and then ran the foce script by running './utils/foce/foce omftest -publish -sensors &'
15. Once the application is running, you can check the service by opening another command window, typing 'gosiam' and then running 'listPorts omftest -stats' which should show the ports, the status and the Samples, errors, and retries stats

# MRG Development

This page last changed on Nov 24, 2009 by kgomes.

**Statement of Work**

Statement of Work
OOI-CI
Pilot prototype RedHat MRG framework

MBARI and NCSA propose to prototype an implementation of their Observatory Middleware Framework on the RedHat MRG framework as a risk-reduction pilot for the OOI-CI project.

MBARI and NCSA propose to split equally the $150,000 subaward for this prototyping effort, with $75,000 going to MBARI and $75,000 going to NCSA. Detailed proposed budgets are being prepared by the grant offices of MBARI and NCSA.

The duration of the work spans September 1, 2009 through December 31, 2009.

NCSA's contribution to the OMF-OOI pilot project will be to develop a Security Proxy implementation for the RedHat MRG framework and subsequently integrate an equivalent to OMF Authorization Service Unit (ASU) with the Security Proxy. This effort is pending the availability of MRG software, to be provided from RedHat and facilitated by UCSD. The MRG ASU implementation may be a simplified version of the current Java-based ASU, in particular if a Python implementation of XACML cannot be identified.

MBARI's contribution to the OMF-OOI pilot project will be to build an Instrument Proxy Implementation: instrument control and data access procedures as part of the MBARI SIAM infrastructure implementing agreed-to standards from the OGC suite of Sensor Interface standards. MBARI will implement the necessary protocols in low-level hardware that resides with individual instruments and/or in software simulators (that MBARI creates).