

# Implementation Guide

ANDS Project EIF 023

## Publishing Interoperable Data. Experience from the “Underway Data” Project

**Authors:**

**Paul Tildesley (CMAR)**

**Siddeswara Guru (CMAR)**

**Kim Finney (AAD)**

**Miles Jordan (AAD)**

**Version 1.0  
December 2010**



**Australian Government**  
Department of Sustainability, Environment,  
Water, Population and Communities  
Australian Antarctic Division

## **Table of Contents**

Introduction.....	1
The Underway Data Project .....	1
Steps to creating interoperable data.....	3
1. Community agreement.....	3
2. Software and infrastructure requirements.....	4
3. Data provider activities.....	6
4. Activities for the manager of the targeted client tool .....	7
Conclusions .....	7
Appendix 1. Installation and configuration of Geoserver and GeoWebCache .....	8
AAD.....	8
CMAR .....	8
Appendix 2. Installation and configuration of GeoNetwork.....	10
Appendix 3: FTL template for underway data popup window.....	17

### **Document history**

Initial draft	15 December 2010
Version 0.9	20 December 2010
Version 1.0	22 December 2010

## ***Introduction***

This report documents the steps taken by the project “EIF023 Publication of Data from National Research Vessels” to publish near-real-time data to the Australian Ocean Data Network (AODN) [portal](#) and to the Australian Research Data Commons ([ARDC](#)) in 2010. The term “underway data” refers to a set of parameters that are routinely measured by many marine research vessels while they are at sea. Typical parameters in an underway dataset include date and time, location, meteorological data, sea water temperature, salinity and fluorescence. Underway datasets are commonly treated in a different way to specific measurements taken to meet voyage scientific objectives. The aim of the project was to create an easy way for end users to discover information about underway data from two different agencies, visualise the data, interrogate the data at specific points and to download the data in formats that would be useful to them. Making the data from the two agencies interoperable means that a user could combine datasets from the two agencies into one analysis without the need for further processing to match units and other things. The experience of this project in publishing interoperable data from two agencies can be used as a guide by other projects aiming to publish interoperable data when multiple data providers are involved.

## ***The Underway Data Project***

This project was funded by the Australian National Data Service ([ANDS](#)) to assist with automating the publication of data from the CSIRO research vessel RV Southern Surveyor (*RV SS*) and the Australian Antarctic Division vessel RV Aurora Australis (*RV AA*) through the AODN and into the Australian ARDC. The CSIRO vessel is operated by CSIRO Marine and Atmospheric Research (CMAR). Both ships are blue-water national research facilities and are funded by the Australian Government. These two vessels are key elements of Australia’s national research infrastructure, enabling oceanographic, geo-scientific, fishery and ecosystem research in the open ocean of the Australian region and as far as the fixed ice edge of Antarctica.

*RV SS* and *RV AA* are sophisticated data collection platforms capable of acquiring data from the sea surface, the water column and the sea-bed, as well as from the immediate atmosphere. Typically, a suite of data known colloquially as “underway data”, are sampled from a wide range of instruments whilst the vessels are in transit. The type of sampled parameters making up this “underway data” suite, the sampling regimes, sampling rates and formats of the captured data currently varies between vessels as do the data publishing policies.

This project aimed to bring the data publication processes of the two vessels into alignment so that both vessels routinely publish their “underway data” suites and accompanying metadata, in near-real time (automatically) to the AODN, and via this network into the ARDC. This has the benefit of providing Australian researchers and environmental managers with almost immediate access to these data as they are collected, along with appropriate (shallow and deep) metadata. Importantly, data emanating from each vessel can then be integrated because the vessel managers will work together to ensure common data publication protocols and standards.

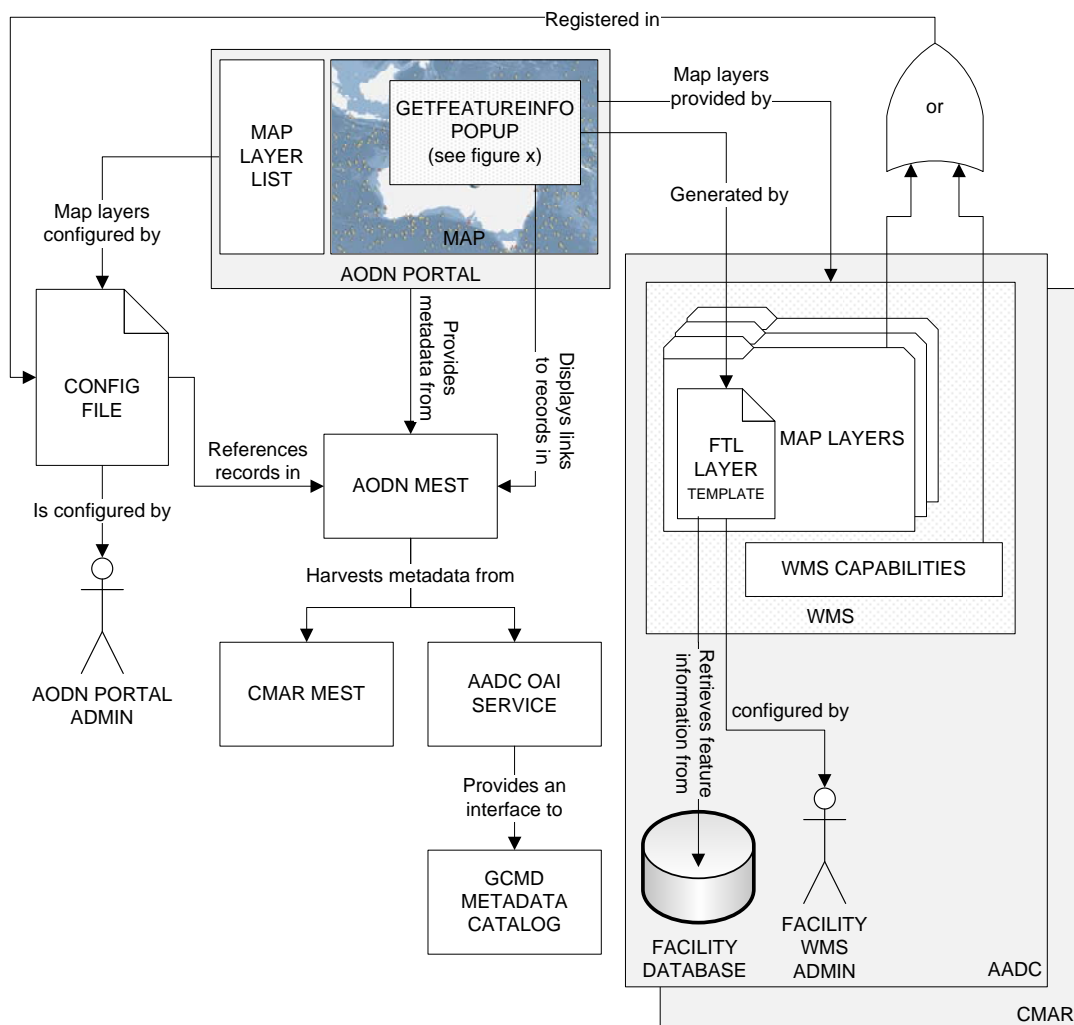
The AODN portal works with Open Geospatial Consortium ([OGC](#)) compliant web services (web map services, web feature services and web coverage services) for machine to machine transfer of maps and data. This project therefore published the underway data using open source software to create these services and tailored the functionality to match the portal. End users can also choose to access the underway

data using other OGC-compliant clients, with detailed functionality depending on the capability of the particular client in use.

In order to publish the data from the two research vessels in a coordinated way so that the different datasets could be combined by end users, the two agencies involved in the project had to work closely together and agree on some conventions that each agency would follow when publishing the data using a set of common software. This can be seen as an exemplar for a community working together to create standards and conventions for publishing data in an interoperable way. The project partners adopted a simple feature model for encoding underway parameters, mainly because the current software used by both agencies to deliver services (i.e. Geoserver) is not yet capable (performance-wise) of delivering complex feature types in application schema.

It will be assumed for the rest of this document that users who might reference this document intend to publish data using OGC-compliant web services.

Figure 1 gives an overview of the components involved in publishing the underway layers and the relationships between them.



**Figure 1. The relationship between the data providers, metadata and the AODN portal**

## ***Steps to creating interoperable data***

### **1. Community agreement**

The project participants (in this case the two agencies) need to agree on a set of conventions that will be used when publishing a particular data type. These conventions include:

- a. the parameters to be delivered (names, units). This should be an exhaustive list of all parameters that may be part of the dataset from any data provider. In most cases, the actual parameters delivered by a data provider will be a subset of the full list. In the case of the underway data project, input was sought from the six agencies in the Australian Ocean Data Network Joint Facility in order to get the best coverage of possible parameters to simplify any future expansion of the harmonised underway data. When deciding on a common set of names for the parameters, the limitations of the various data repositories need to be considered. For example, Oracle has a 30 character limit on parameter names and most special characters are not allowed.
- b. The level of granularity of the data. Most collections of data can be described at a number of levels and the community has to decide what would make sense in the majority of cases for the particular data being considered. For the underway data project, it was decided that describing the data at a voyage level made most sense.
- c. The number and details of metadata records to be created for each data stream. This is linked to the level of granularity and also to the depth that the community wishes to go to. For example it is possible to create a metadata record for each sensor or each instrument or just at the datastream level. The underway project is publishing detail voyage level metadata and the details of parameters measured during the voyage.
- d. The information that will be displayed to the end user when they click on a data point on a map (the “GetFeatureInfo” information). What is displayed here depends on the client software. The AODN portal accepts [FreeMarker](#) Template Language as a way to control the contents of the GetFeatureInfo popup box so the underway project used this to create common content and layout.
- e. The way that data will be delivered to the end user
  - i. File formats (CSV, netCDF, ...)
  - ii. File details (eg the amount of metadata included in data files)

The underway project decided to make the data available in three formats; CSV, KML and netCDF. The first two of these are produced natively by Geoserver and the project wrote some java code and used the Java NetCDF library to create netCDF files after every database update. These netCDF files are compliant with the [Climate and Forecasting](#) conventions, version 1.4 except where a standard name for one of the underway parameters does not exist.

- f. The software that will be used to deliver the metadata and data services. There are a number of open source packages that will produce OGC compliant web services, including Mapserver, Geoserver, Deegree, THREDDS. It is not strictly necessary for all data providers to use the same set of software but each package has different capabilities and it is

easier to harmonise across data providers if they are using similar software. The underway project chose to use Geoserver as it is simple to install, manage and use and provide the necessary functionality. The AODN portal works with [GeoNetwork](#) as its catalogue and so the requirement for the underway project was to provide metadata in a way that could be harvested into the AODN instance of GeoNetwork. The agencies chose to do this in different ways; AAD stores their metadata in a NASA GCMD tool so the AODN installed a GeoNetwork plug-in to be able to harvest metadata records from AAD. CMAR runs a custom built metadata system (MarLIN) that is not directly harvestable and therefore chose to install an externally visible version of GeoNetwork and export their metadata records to that instance. The AODN GeoNetwork can then harvest from there. The AODN GeoNetwork is responsible for exporting records to the Australian Research Data Commons.

## 2. Software and infrastructure requirements

### a. OGC compliant web services

Given the project requirement to use OGC compliant web services to deliver maps and data, there are a number of open source and proprietary packages that could be used. The underway project considered only open source software and there were three contenders:

#### **Mapserver**

This is a stand-alone CGI program built from C code and many open source libraries. It is quite difficult to build and maintain. For each layer that is to be published, the user has to manually create a .map file that specifies all of the details of the layer, from data source to style and behaviour at any magnification. This requires some time and experimentation to get familiar with the intricacies of map files. There is a user guide and several books available for Mapserver and a lot of information on the Internet

#### **Geoserver**

This is a java program that runs in a java servlet container such as [Apache Tomcat](#). Installation is very simple (simply place the .war file in the appropriate place and restart tomcat). A companion package called GeoWebCache can be installed to sit in between the user and Geoserver to reduce the load on Geoserver if there is a reasonable amount of traffic. This does however change the responses that the web services client gets back and so needs to be used with care. Layers are defined in Geoserver using a graphical user interface and this process is straightforward. If the user needs to create a large number of layers, Geoserver has a REST interface plugin that can be used to programmatically build layers. The documentation on Geoserver is reasonably clear and there is a lot of other information available on the Internet, including mailing groups that are responsive.

#### **Deegree**

This is a free comprehensive geospatial software package that implements OGC web services (WMS, WFS, WPS, CSW) as well as a number of other tools for spatial data. It is open source (LGPL), standards compliant (OGC, ISO/TC211) and is written in Java. Deegree offers a

range of OGC services, including WMS, WFS, WCS, CSW, WPS and SOS.

The underway data project chose to use Geoserver because of the ease of installation and maintenance, the ease of definition of layers and the ability to use FreeMarker Template Language to control the content and style of the GetFeatureInfo popup box in the AODN portal. Information on the configuration of Geoserver is shown in appendix 1.

b. Metadata tool

In order for the metadata on the data being served by all data providers to be visible to end users, it needs to be available in one place. This means that the central metadata tool must be able to harvest records from each of the data providers in some way. In the case of the underway project, the central metadata tool is the AODN instance of GeoNetwork and it is harvesting metadata records from both the AAD and CMAR, but doing this in different ways. The AAD uses a NASA GCMD tool to manage their metadata and GeoNetwork has a plugin to harvest from this, using the [OAI-PHM](#) protocol. CMAR has a custom built tool called MarLIN and they installed a local instance of GeoNetwork and exported all MarLIN records to this instance. The AODN GeoNetwork then harvests records from the CMAR GeoNetwork daily.

The central metadata repository needs to be able to export in RIF-CS format so that the records can be exported to the ARDC. GeoNetwork can do this and the exporting process is managed by the AODN Development Office.

Information on the installation and configuration of GeoNetwork is shown in appendix 2.

c. File formats for data delivery

Geoserver is able to deliver data in a range of formats, including CSV, KML and GML. If a project wishes to deliver data in a format not supported by Geoserver then they need to write a special program to create the file and then link to the file from the GetFeatureInfo popup window. The underway project wished to give the end user the option of downloading the data as a netCDF file, a popular format in marine science but not a format supported by Geoserver. The underway project therefore created a Java program that extracts data from the database and creates a netCDF file using the [Java netCDF library](#) that is maintained by Unidata. This is a freely available library.

The underway data NetCDF file template format is constructed to comply with the IMOS NetCDF User's Manual, which is based on the [OceanSITES User's Manual](#). The NetCDF data files are created by populating this template file.

Initially the NetCDF is published as a direct downloadable file of the whole voyage dataset. Users would have more flexibility in selecting particular parameters or time range if the netCDF file was made available via an [OPeNDAP](#) server and this will be explored later.

The NetCDF file has one unlimited dimension of time and each underway measurement is considered as a variable. The time value is stored as the

number of minutes from the Unix reference year i.e., 1<sup>st</sup> Jan 1970 00:00:00 UTC.

### 3. Data provider activities

Once there is community agreement on the points listed above, each data provider can then start on the work required to publish the relevant layers. This involves:

- a. creating a data repository that will support the agreed way to publish the data. Possibilities include new database tables, a view of existing database tables or a netCDF directory available to THREDDS. In the underway data example, the repository was a database view for each agency but using different database software (PostgreSQL for AAD and Oracle for CMAR). A specific view was necessary to bring together data that is held in a number of tables.
- b. installing the agreed software to create the metadata and data services. These services need to be visible to the whole Internet and this can be a challenge for some organisations to overcome the security considerations of exposing data outside their firewall. One safe way to do this is to use a [proxy server](#).
- c. setting up a process that creates metadata records as required. This could be a manual or an automated process. In the underway data example, a metadata record is created for each voyage of each research vessel. This is created semi-automatically by using templates. ANZMet Lite is one of the software tool that may be great interest for creating metadata and upload to any MEST instances.
- d. If the data being published is dynamic, then a process will need to be created that adds data to the data repository as it becomes available. This could be a manual or an automated process. In the underway data project, data from the CMAR ship is sent to an ftp area on shore. A Unix cron job regularly checks for new data arriving into the ftp area and adds it to the database. For AAD the data is sent from the ship in an email that is processed as soon as it arrives and added to the database. Once in the database, the data automatically becomes available to the web services.
- e. creating the web services that expose the data in the agreed way as WMS and WFS or WCS. The agreed styling for a layer can be shared between data providers by means of a Style Layer Descriptor file.
- f. For the underway data project, because both agencies are using Geoserver to create the services, a common template was created for the GetFeatureInfo pop-up window. Geoserver uses the open source package [FreeMarker](#) to interpret these templates, which are written in FreeMarker Template Language (FTL). As the pop-ups are supposed to be very similar for different data providers, the output section of the template should be identical across data providers, with differences in the input section of the template that accesses the data. The FTL template used by the underway project and the contents of the popup window are shown in appendix 3.

#### **4. Activities for the manager of the targeted client tool**

A few actions are required from the maintainers of the target tool that clients will use to access the data layers in order to ensure that layers are discoverable and viewable and the data downloadable.

- a. Connect the central metadata tool to the metadata streams of the data providers so that metadata records are harvested and available to the end users.
- b. Connect the central metadata tool to the ARDC so that records harvested from the data providers are passed on and visible at this higher level.
- c. Make the data layers directly visible in the client tool so that end users can see a list of available layers and can choose to visualise one or more and then download the data.
- d. Ideally the infrastructure surrounding the client tool would regularly check the availability of listed layers and notify the data provider if a layer is unavailable for a certain time.

#### ***Conclusions***

The EIF023 project to publish interoperable underway data has demonstrated that it is possible to take existing data streams from different data providers and publish them in a way that is interoperable for the end user. This requires community agreement on the data content and description and then for each data provider to implement the agreement. In most cases, it will be possible to publish interoperable data without significant changes to existing data flows.

## **Appendix 1. Installation and configuration of Geoserver and GeoWebCache**

### **AAD**

Installation procedures for Tomcat and Geoserver at the AAD are shown in separate documents, “Installing Tomcat 6 on Ubuntu 8.04 (Hardy)” and “Upgrading Geoserver on Tomcat 6 and Ubuntu”. These documents are available at the website for the underway project, <http://imos.org.au/rtunderway.html>.

### **CMAR**

For the underway project, CMAR installed Geoserver on a SuSE Linux system. All software was installed in the /opt/apache-tomcat-7.0.4 directory (directories below are relative to this).

#### **Tomcat installation:**

```
tar -zxvf apache-tomcat-7.0.4.
```

Change the port numbers in the conf/server.xml file

<b>parameters</b>	<b>old port numbers</b>	<b>new port numbers</b>
shutdown	8005	7005
redirectport	8443	7443
connector	8080	7080
AJP connector	8009	7009

Add JAVA\_HOME, JRE\_HOME, CATALINA\_HOME in bin/catalina.sh

Command to start server: sh startup.sh

#### **Geoserver Installation:**

Geoserver is deployed using war installation. Place the war file in an accessible directory, start the tomcat manager, point to the war file and click on “deploy”.

Change the value of GEOSERVER\_DATA\_DIR in the webapps/geoserver/WEB\_INF/web.xml

For this installation GEOSERVER\_DATA\_DIR = /opt/tomcat7-geoserver/data

Copy jar files for oracle jdbc, REST config and application schema into webapps/geoserver/WEB\_INF/lib/

#### **GeoWebCache Installation:**

GeowebCache is also deployed as war installation, in a similar way to Geoserver.

Change the WMS getcapabilities url in /webapps/geowebcache/WEB-INF/geowebcache-core-context.xml

Change the WFS url in /webapps/geowebcache/WEB-INF/geowebcache-wfsclient-context.xml

Change the username and password in webapps/geowebcache/WEB-INF/users.properties

**Issues to be resolved**

The geoRSS context has been disabled in GeoWebCache because the server hangs.

## **Appendix 2. Installation and configuration of GeoNetwork**

GeoNetwork is a catalogue application to manage spatially referenced resources. It provides powerful metadata editing and search functions as well as an embedded interactive web map viewer. It is currently used in numerous Spatial Data Infrastructure initiatives across the world, including the Australian Ocean Data Network (AODN) and the Integrated Marine Observing System (IMOS). General information on GeoNetwork can be found at <http://geonetwork-opensource.org/>

GeoNetwork uses a database to store its data. This can be any JDBC capable database, but it has been tested against Oracle, MySQL and PostgreSQL. CMAR is using GeoNetwork to publish metadata and it has an Oracle datastore.

The main source of GeoNetwork documentation is the open source project documentation page - <http://geonetwork-opensource.org/docs.html>

The installation is described here - <http://geonetwork-opensource.org/manuals/2.6.1/users/quickstartguide/installing/index.html>

Simon Pigot is an Australian developer of GeoNetwork and there are three published presentation from Simon on implementing GeoNetworks:

<http://www.osdm.gov.au/Metadata/GeoNetwork/Resources/ImplementingGeoNetworkBasics.pdf/?id=1014>

and to a lesser extent:

<http://www.osdm.gov.au/Metadata/GeoNetwork/Resources/ImplementingGeoNetworkAdvanced.pdf/?id=1015>

and another intro/background one:

<http://www.geospatial.govt.nz/assets/Resources/Presentations--Project-Data/implementing-geonetwork.pdf>

Below is a modified version of the original Oracle install script, that fixes some index syntax, replaces long datatypes with the clob datatype and references an index tablespace called indx (the last is optional, and specific to the CMAR situation).

This script (create-db-oracle\_modified.sql) would need to replace the original which is located in \$GEONETWORK\_HOME/gast/setup/sql/create-db-oracle.sql

```
-----  
-- ===  Sql Script for Database : Geonet  
-- ===  
-- === Build : 148  
-- modified 2010-10-05 edg029  
-----
```

```
CREATE TABLE Relations
(
  id int,
  relatedId int,
  constraint relations$pk primary key(id,relatedId) using index tablespace indx
);
```

```
CREATE TABLE Categories
(
  id int,
  name varchar(32) not null,
  constraint categories$pk primary key(id) using index tablespace indx,
  constraint categories$uk unique(name) using index tablespace indx
);
```

```
CREATE TABLE Settings
(
  id int,
  parentId int,
  name varchar(32) not null,
  --value long,
  value clob,
  constraint settings$pk primary key(id) using index tablespace indx,
  foreign key(parentId) references Settings(id)
);
```

```
CREATE TABLE Languages
(
  id varchar(5),
  name varchar(32) not null,
  constraint language$pk primary key(id) using index tablespace indx
);
```

```
CREATE TABLE Sources
(
  uuid varchar(250),
  name varchar(250),
  isLocal char(1) default 'y',
  constraint sources$pk primary key(uuid) using index tablespace indx
);
```

```
CREATE TABLE IsoLanguages
(
  id int,
  code varchar(3) not null,
```

```
constraint isoLanguages$pk primary key(id) using index tablespace indx,  
constraint isoLanguages$uk unique(code) using index tablespace indx  
);
```

```
CREATE TABLE IsoLanguagesDes  
(  
  idDes int,  
  langld varchar(5),  
  label varchar(96) not null,  
  constraint isoLanguagesDes$pk primary key(idDes,langld) using index tablespace  
indx,  
  constraint isoLanguagesDes$fk1 foreign key(idDes) references IsoLanguages(id),  
  constraint isoLanguagesDes$fk2 foreign key(langld) references Languages(id)  
);
```

```
CREATE TABLE Regions  
(  
  id int,  
  north float not null,  
  south float not null,  
  west float not null,  
  east float not null,  
  constraint regions$pk primary key(id) using index tablespace indx  
);
```

```
CREATE TABLE RegionsDes  
(  
  idDes int,  
  langld varchar(5),  
  label varchar(96) not null,  
  constraint regionsDes$pk primary key(idDes,langld) using index tablespace indx,  
  constraint regionsDes$fk1 foreign key(idDes) references Regions(id),  
  constraint regionsDes$fk2 foreign key(langld) references Languages(id)  
);
```

```
CREATE TABLE Users  
(  
  id int,  
  username varchar(32) not null,  
  password varchar(40) not null,  
  surname varchar(32),  
  name varchar(32),  
  profile varchar(32) not null,  
  address varchar(128),  
  state varchar(32),  
  zip varchar(16),  
  country varchar(128),
```

```

email    varchar(128),
organisation varchar(128),
kind     varchar(16),
constraint users$pk primary key(id) using index tablespace indx,
constraint users$uk unique(username) using index tablespace indx
);

```

CREATE TABLE Operations

```

(
  id      int,
  name    varchar(32) not null,
  reserved char(1)   default 'n' not null,
  constraint operations$pk primary key(id) using index tablespace indx
);

```

CREATE TABLE OperationsDes

```

(
  idDes int,
  langId varchar(5),
  label  varchar(96) not null,
  constraint operationsDes$pk primary key(idDes,langId) using index tablespace
indx,
  constraint operationsDes$fk1 foreign key(idDes) references Operations(id),
  constraint operationsDes$fk2 foreign key(langId) references Languages(id)
);

```

CREATE TABLE Groups

```

(
  id      int,
  name    varchar(32) not null,
  description varchar(255),
  email   varchar(32),
  referrer int,
  constraint groups$pk primary key(id) using index tablespace indx,
  constraint groups$uk unique(name) using index tablespace indx,
  constraint groups$fk foreign key(referrer) references Users(id)
);

```

CREATE TABLE GroupsDes

```

(
  idDes int,
  langId varchar(5),
  label  varchar(96) not null,
  constraint groupsDes$pk primary key(idDes,langId) using index tablespace indx,
  constraint groupsDes$fk1 foreign key(idDes) references Groups(id),
  constraint groupsDes$fk2 foreign key(langId) references Languages(id)
);

```

```

CREATE TABLE UserGroups
(
  userId int,
  groupId int,
  constraint userGroups$pk primary key(userId,groupId) using index tablespace
indx,
  constraint userGroups$fk1 foreign key(userId) references Users(id),
  constraint userGroups$fk2 foreign key(groupId) references Groups(id)
);

```

```

CREATE TABLE CategoriesDes
(
  idDes int,
  langId varchar(5),
  label varchar(96) not null,
  constraint categoriesDes$pk primary key(idDes,langId) using index tablespace
indx,
  constraint categoriesDes$fk1 foreign key(idDes) references Categories(id),
  constraint categoriesDes$fk2 foreign key(langId) references Languages(id)
);

```

```

CREATE TABLE Metadata
(
  id int,
  uuid varchar(250) not null,
  schemaId varchar(32) not null,
  isTemplate char(1) default 'n' not null,
  isHarvested char(1) default 'n' not null,
  createDate varchar(24) not null,
  changeDate varchar(24) not null,
  --data long not null,
  data clob not null,
  source varchar(250) not null,
  title varchar(255),
  root varchar(255),
  harvestUuid varchar(250) default null,
  owner int not null,
  groupOwner int default null,
  harvestUri varchar(255) default null,
  rating int default 0 not null,
  popularity int default 0 not null,
  constraint metadata$pk primary key(id) using index tablespace indx,
  constraint metadata$uk unique(uuid) using index tablespace indx,
  constraint metadata$fk1 foreign key(owner) references Users(id),
  constraint metadata$fk2 foreign key(groupOwner) references Groups(id)
);

```

```
--CREATE INDEX MetadataNDX1 ON Metadata(uuid); --already indexed above
```

```
CREATE INDEX MetadataNDX2 ON Metadata(source) tablespace indx;
```

```
create index metadatandx20 on metadata(title) tablespace indx;
```

```
CREATE TABLE MetadataCateg
```

```
(
  metadataId int,
  categoryId int,
  constraint metadataCateg$pk primary key(metadataId,categoryId) using index
  tablespace indx,
  constraint metadataCateg$fk1 foreign key(metadataId) references Metadata(id),
  constraint metadataCateg$fk2 foreign key(categoryId) references Categories(id)
);
```

```
CREATE TABLE OperationAllowed
```

```
(
  groupId int,
  metadataId int,
  operationId int,
  constraint operationAllowed$pk primary key(groupId,metadataId,operationId) using
  index tablespace indx,
  constraint operationAllowed$fk1 foreign key(groupId) references Groups(id),
  constraint operationAllowed$fk2 foreign key(metadataId) references Metadata(id),
  constraint operationAllowed$fk3 foreign key(operationId) references Operations(id)
);
```

```
CREATE TABLE MetadataRating
```

```
(
  metadataId int,
  ipAddress varchar(32),
  rating int not null,
  constraint metadataRating$pk primary key(metadataId,ipAddress) using index
  tablespace indx,
  constraint metadataRating$fk foreign key(metadataId) references Metadata(id)
);
```

```
-----  
--cleanup script-----  
-----
```

```
/*  
drop table MetadataRating;  
drop table OperationAllowed;  
drop table MetadataCateg;  
drop table Metadata;
```

```
drop table CategoriesDes;  
drop table UserGroups;  
drop table GroupsDes;  
drop table Groups;  
drop table OperationsDes;  
drop table Operations;  
drop table Users;  
drop table RegionsDes;  
drop table Regions;  
drop table IsoLanguagesDes;  
drop table IsoLanguages;  
drop table Sources;  
drop table Languages;  
drop table Settings;  
drop table Categories;  
drop table relations;  
*/
```

## Appendix 3: FTL template for underway data popup window

The following is an abbreviated version of the template used by Geoserver to create a popup window when a user clicks on a part of the track of the CMAR vessel. The coding for most of the parameters has been removed for brevity but the style of the coding is the same for each parameter. The window gives the user the location of the closest point on the track and the values of all underway parameters at that point. There are also links for downloading the full underway dataset from that voyage in a number of formats.

```
<@compress single_line=true>

<#setting number_format="0.###">

<h3>RVSS Underway Voyage Data</h3>

<#list features as feature>
  
  <div class="feature">
    <h4>RV Southern Surveyor</h4>
    ${feature.DATE_TIME_UTC.value?string} UTC
    <ul>
      <li><b>Latitude:</b>
<#attempt>${feature.LATITUDE_DEGNORTH.value?number}&deg;N<#recover>no
t available</#attempt></li>
      <li><b>Longitude:</b>
<#attempt>${feature.LONGITUDE_DEGEAST.value?number}&deg;E<#recover>no
t available</#attempt></li>
    </ul>
    <b>Download (entire voyage):</b>
    <ul>
      <li><a
href="http://www.cmar.csiro.au/geoserver/wfs?request=getfeature&servi
ce=wfs&version=1.0.0&outputformat=CSV&typename=${type.name}"
target="_blank">CSV</a></li>
      <li><a
href="http://www.cmar.csiro.au/geowebcache/service/kml/CSIRORVSS:${ty
pe.name}.png.kml">KML (Google Earth)</a></li>
      <li><a href="javascript:alert('This download type is not yet
available');return false;">NetCDF</li>
    </ul>
    <div style="clear:both; border-bottom: 1px bevel #aaa; margin-
bottom: 10px;">
      <a
href="http://www.cmar.csiro.au/geonetwork/srv/en/metadata.show?id=${f
eature.METADATA_ID.value}" target="_blank">View metadata for this
dataset</a>
    </div>
    <div style="clear:both">
      <h5>Sensor measurements at this location</h5>
      <table>
        <tr>
          <td>Ship Course Over Ground:</td>
        </tr>
      </table>
    </div>
  </div>
</#list>
```

```

999.9'>
    <#if feature.SHIP_COURSE_OVER_GROUND_DEG.value == '-
        not available
    <#else>
        ${feature.SHIP_COURSE_OVER_GROUND_DEG.value}&deg;
    </#if>
</td>
</tr>
<tr>
    <td>Ship Heading (Gyro):</td>
    <td>
        <#if feature.SHIP_HEADING_GYRO_DEG.value == '-999.9'>
            not available
        <#else>
            ${feature.SHIP_HEADING_GYRO_DEG.value}&deg;
        </#if>
    </td>
</tr>

```

(lines removed)

```

<tr>
    <td>Voyage Name:</td>
    <td>
        <#attempt>
            ${feature.VOYAGENAME.value?string}
        <#recover>
            not available
        </#attempt>
    </td>
</tr>
</table>
</div>
<#break>
</div>
</#list>
</@compress>

```

This produces the following content in a pop-up window

## RVSS Underway Voyage Data



### RV Southern Surveyor

16/10/2010 11:52:00 UTC

- **Latitude:** -32.209°N
- **Longitude:** 153.19°E

### Download (entire voyage):

- [CSV](#)
- [KML \(Google Earth\)](#)
- [NetCDF](#)

[View metadata for this dataset](#)

### Sensor measurements at this location

Ship Course Over Ground: 16.8°  
 Ship Heading (Gyro): 19.3°

Ship Heading (GPS):	19.8°
Ship Speed (Log):	9.42 kn
Ship Speed Over Ground (GPS):	9.57 kn
Altitude:	23.055 m
Atmospheric Pressure:	1010.462 hPa
Fluorescence:	0.09 AFU
Radiation - infrared (portside):	292.1 W/m <sup>2</sup>
Radiation - infrared (starboardside):	294.24 W/m <sup>2</sup>
Radiation - PAR:	6 µE/m <sup>2</sup> /s;
Radiation - solar (portside):	-1.08 W/m <sup>2</sup>
Radiation - solar (starboardside):	-2.09 W/m <sup>2</sup>
Accumulated Rain (Foremast):	0 mm
Accumulated Rain (Mainmast):	0 mm
Rain Rate (Foremast):	0 mm/hour
Relative Humidity (portside):	45 %
Relative Humidity (starboardside):	45.7 %
Sea Water Salinity - TSG:	35.489 PSU
Sea Water Temperature - TSG:	19.91 °C
Sensor Temperature - TSG:	20.107°C
Water Flow Rate - TSG:	33 l/min
Air Temperature (portside):	14.3 °C
Air Temperature (starboardside):	14.4 °C
Water Depth:	2362.7 m
Centre Beam Depth:	473.077 m
Wind Direction (foremast, uncorrected):	260.6°
Wind Direction (foremast, corrected):	245.1°
Wind Direction (mainmast, uncorrected):	257.92°
Wind Direction (mainmast, corrected):	259.34 °
Wind Speed (foremast, uncorrected):	12.2 kn
Wind Speed (foremast, corrected):	16.3 kn
Wind Speed (mainmast, uncorrected):	26.49 kn
Wind Speed (mainmast, corrected):	29.53 kn
Maximum Wind Gust:	31.1 kn
Water Temperature Equilibrator - PCO2:	20.02 °C
XCO2 - PCO2:	362.38 ppm
Water Vapour - PCO2:	1.4 mmol/mole
Water Flow - PCO2:	2.8 l/min
Vent Flow - PCO2:	2.74 ml/min
Licor Flow - PCO2:	98.88 l/min
Licor Pressure - PCO2:	1010.4 hPa
Equilibrator Pressure - PCO2:	0 hPa
Condenser Temperature - PCO2:	6.7 °C
Measurement Type - PCO2:	0
Pump Speed - PCO2:	5 (1-255)
Voyage Name:	ss2010_v09

---