NOAA Technical Report NOS CS 26

IMPLEMENTING A LIVE ACCESS SERVER TO ENHANCE ACCESS TO COAST SURVEY DEVELOPMENT LABORATORY DATA - GULF OF MEXICO EXPERIMENTAL NOWCAST/FORECAST SYSTEM

Silver Spring, Maryland
March 2006



noaa National Oceanic and Atmospheric Administration

U.S. DEPARTMENT OF COMMERCE
National Ocean Service
Coast Survey Development Laboratory

Office of Coast Survey
National Ocean Service
National Oceanic and Atmospheric Administration
U.S. Department of Commerce

The Office of Coast Survey (OCS) is the Nation's only official chartmaker. As the oldest United States scientific organization, dating from 1807, this office has a long history. Today it promotes safe navigation by managing the National Oceanic and Atmospheric Administration's (NOAA) nautical chart and oceanographic data collection and information programs.

There are four components of CS:

The Coast Survey Development Laboratory develops new and efficient techniques to accomplish Coast Survey missions and to produce new and improved products and services for the maritime community and other coastal users.

The Marine Chart Division collects marine navigational data to construct and maintain nautical charts, Coast Pilots, and related marine products for the United States.

The Hydrographic Surveys Division directs programs for ship and shore-based hydrographic survey units and conducts general hydrographic survey operations.

The Navigation Services Division is the focal point for Coast Survey customer service activities, concentrating predominantly on charting issues, fast-response hydrographic surveys and Coast Pilot updates.

NOAA Technical Report NOS CS 26

IMPLEMENTING A LIVE ACCESS SERVER TO ENHANCE ACCESS TO COAST SURVEY DEVELOPMENT LABORATORY DATA - GULF OF MEXICO EXPERIMENTAL NOWCAST/ FORECAST SYSTEM

John L. Schattel, Jr.

March 2006



noaa National Oceanic and Atmospheric Administration

U.S. DEPARTMENT
OF COMMERCE
Carlos Gutierrez, Secretary

National Oceanic and
Atmospheric Administration
Conrad C . Lautenbacher, Jr.,
VADM USN (Ret.), Under Secretary

National Ocean Service
John H. Dunnigan
Assistant Administrator

Office of Coast Survey
Captain Steven Barnum

Coast Survey Development
Laboratory
Mary Erickson

NOTICE

Mention of a commercial company or product does not constitute an endorsement by NOAA. Use for publicity or advertising purposes of information from this publication concerning proprietary products or the tests of such products is not authorized.

TABLE OF CONTENTS

LIST OF FIGURES iv

LIST OF TABLES v

LIST OF ACRONYMS vi

1. INTRODUCTION 1

1.1. Background 1
1.2. Required Functionality 1
1.3. LAS Architecture 2

2. LAS FUNCTIONAL DESCRIPTION 4

2.1. User Interface 4
2.1.1. Navigation Bar 5
2.1.1.1. Datasets Link 5
2.1.1.2. Variables Link 6
2.1.1.3. Constraints Link 6
2.1.1.4. Previous Output Link 6
2.1.1.5. Define Variable Link 6
2.1.1.6. About Link 6
2.1.1.7. Contact Link 6
2.1.2. Compare Two Datasets Tab 6
2.1.3. Discovering Datasets 7
2.1.3.1. Data Source Page 7
2.1.3.2. Sub-collection Page 7
2.1.3.3. Dataset Page 9
2.1.4. Dataset Quick Links 10
2.1.5. Variables Page 10
2.1.6. Constraints Page 11
2.1.6.1. Select View 12
2.1.6.2. Select Output 12
2.1.6.3. Select Region 13
2.1.6.3.1. Choosing Predefined Regions 13
2.1.6.3.2. Creating User Defined Region 13
2.1.6.4. Select Time 14
2.1.6.5. Select Depth 14
2.1.6.6. Select Options 15
2.2. Use Cases 15
2.2.1. Data Discovery and Display 15
2.2.2. Data Retrieval 16

3. PREREQUISITE AND SUPPORT SOFTWARE 17

3.1. Automated Customization Scripts 17
3.2. Ferret 17
3.3. Custom LAS Scripts 18
3.4. Software Update Scripts 18

3.5. OPeNDAP Server 19
3.6. OPeNDAP Aggregation Server 19
3.7. Tomcat Servlet Engine 20
3.8. Other Applications 20
3.8.1. Apache Web Server 20
3.8.2. Perl 20
3.8.3. MySQL 20

4. FUTURE DEVELOPMENT 21

5. ACKNOWLEDGEMENTS 22

APPENDIX A: CUSTOMIZATION SOFTWARE INSTALLATION INSTRUCTIONS 23

A1. Introduction. 23
A2. Key Concepts. 23
A3. Development Installation. 23
A4. Production Installation. 24

APPENDIX B: CUSTOMIZATION SCRIPTS DESCRIPTION 26

B1. setupLAS. 26
B2. place_holder.sh 26
B3. copy_NGOM_files.sh 27
B4. createAggregatexml.php 29
B5. createCatalogConfig.php 29
B6. createCATxml.php 30
B7. createDGOMxml.php 31
B8. createDepthxml.php 32
B9. createSigmaxml.php 33
B10. createLASxml.php 34

APPENDIX C: LAS (VERSION 6.5.1.2) PREREQUISITE AND SUPPORT SOFTWARE 37

C1. Prerequisite Software. 37
C1.1. Ferret (Version: 5.8.1) 37
C1.2. Perl (Version: v5.8.0 built for i386-linux-thread-multi) 37
C1.3. MySQL (Version: 4.0.25) 37
C1.4. Expat (Version: 1.95.8) 38
C1.5. Java (Version: 1.4.2.09) 38
C1.6. NetCDF Libraries (Version: 3.6.0-pl) 38
C1.7. Ant (Version: 1.6.5) 38
C2. Support Software 38
C2.1. Apache Web Server (Version: 2.0.46) 38
C2.2. Apache Tomcat (Version: 5.5.9) 38
C2.3. RedHat Linux (Version: Red Hat Enterprise Linux AS release 3 [Taroon Update 6]) 38

APPENDIX D: RUNNING THE LAS (VERSION 6.5.1.2) CONFIGURE SCRIPT 39

D1. Prerequisite Steps 39
D2. Development LAS Configuration 39
D3. Public LAS Configuration 40

LIST OF FIGURES

Figure 1. NGOM Surface Temperature 2
Figure 2. LAS Architecture 3
Figure 3. Product Preparation Process 4
Figure 4. CSDL LAS Navigation Bar 5
Figure 5. Sample Data Collection Hierarchy 7
Figure 6. CSDL LAS Data Source Page 8
Figure 7. CSDL LAS Sub-collection Page 8
Figure 8. CSDL LAS Dataset Page 9
Figure 9. Dataset Quick Links 10
Figure 10. CSDL LAS Variables Page 10
Figure 11. CSDL LAS Constraint Page 11
Figure 12. Sample Output Product 13
Figure 13. Interactive Java Map 14
Figure 14. Aggregation Dataset Architecture 19

LIST OF TABLES

Table 1. NGOM Variables and Corresponding Views and Output Formats 12
Table 2. Custom Ferret Scripts 18
Table 3. Releases Directories..... 24
Table 4. Data File Name Conversion..... 28
Table 5. Standard Depths in Meters..... 33

LIST OF ACRONYMS

CF	Climate and Forecast
CO-OPS	Center for Operational Oceanographic Products and Services
CSDL	Coast Survey Development Laboratory
DODS	Distributed Oceanographic Data System
FTP	File Transfer Protocol
GNOME	General NOAA Oil Modeling Environment
HAZMAT	Hazardous Materials Response Division
HTTP	Hypertext Transfer Protocol
LAS	Live Access Server
MDL	Meteorological Development Laboratory
MMAP	Marine Modeling and Analysis Programs
netCDF	Network Common Data Format
NGOM	NOS Gulf of Mexico Model
NRAP	NOAA Rotational Assignment Program
NOAA	National Atmospheric and Oceanic Administration
NOS	National Ocean Service
NWS	National Weather Service
OAR	Oceanic and Atmospheric Research
OPeNDAP	Open-source Project for a Network Data Access Protocol
OR&R	Office of Response and Restoration
PMEL	Pacific Marine Environmental Laboratory
PORTS	Physical Oceanographic Real-Time System
TMAP	Thermal Modeling and Analysis Project
THREDDS	Thematic Real-time Environmental Distributed Data Services
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

ABSTRACT

The Marine Modeling and Analysis Programs (MMAP) Branch of NOAA/National Ocean Service's (NOS) Coast Survey Development Laboratory (CSDL) has implemented a Live Access Server (LAS) to improve dissemination of its scientific data. The LAS, which is a software package developed at NOAA/Oceanic and Atmospheric Research's (OAR) Pacific Marine Environmental Laboratory (PMEL), acts as a web server that helps public and government users to discover, display, and download CSDL datasets. The LAS facilitates finding a needed dataset by providing the user with a hierarchical structure that organizes a collection of datasets with descriptive titles. Each title indicates what data is available in that branch of the hierarchy. Displaying data is accomplished with Ferret, a data visualization program developed by the PMEL. A key component of the LAS download function is a feature allowing users to retrieve only the spatial and temporal portion of the dataset that they need. This capability reduces communication bandwidth and user storage requirements for customers wanting to leverage CSDL's large datasets.

CSDL currently uses its LAS to serve the NOS Gulf of Mexico (NGOM) model[1] data collection. Users can access the NGOM data collection by entering the Uniform Resource Locator (URL) http://dods.ncd.noaa.gov/las/servlets/dataset into any web browser. The data collection contains information on water currents, wind flow, salinity, water temperature, and water surface height. These data are provided on a curvilinear grid covering the Gulf of Mexico and have a sigma vertical coordinate. LAS users can access NGOM output files individually and as a unit aggregated over time. The aggregate provides the user with a continuous set of data covering weeks. The NGOM data collection and other data offerings to be added later, give model developers, hazard assessment and response personnel, commercial and public boating communities, and those interested in the marine environment improved access to CSDL data.

[1] The URL http://nauticalcharts.noaa.gov/csdl/op/dgom.m.html contains more information on the NGOM model.

1. INTRODUCTION

1.1. Background

CSDL produces data that is potentially valuable to entities both inside and outside of NOAA. One organization with a proven need for CSDL model results is NOAA's Office of Response and Restoration (OR&R) Hazardous Materials Response Division (HAZMAT). HAZMAT is responsible for oil and hazardous materials spills in United States waters including the Gulf of Mexico. To fulfill their mission, HAZMAT uses the General NOAA Oil Modeling Environment (GNOME) to determine spill trajectories. To run GNOME, HAZMAT needs input from an ocean model like CSDL's NGOM.

Prior to implementing an LAS, CSDL made NGOM model output files available to HAZMAT via a file transfer protocol (FTP) server. This solution unfortunately meant that large files had to be moved over the internet. A better process would allow HAZMAT personnel to visualize the CSDL NGOM datasets and then retrieve only the portion of the data that they need. It was this HAZMAT requirement that provided the driving use case for the CSDL LAS development project.

1.2. Required Functionality

To satisfy the HAZMAT use case, the LAS serves up CSDL data 24-hours, 7 days-a-week to anyone with an internet connection. The LAS User Interface (UI) organizes dataset files into a hierarchical structure helping users navigate CSDL data collections. To facilitate data retrieval over extended periods of time, the LAS joins individual dataset files into a logical aggregate. These virtual datasets free the user from having to explore and download multiple data files separately.

After locating the desired dataset, a user is able to visualize the information. The initial LAS implementation only provides graphics depicting NGOM model output for the Gulf of Mexico. These images appear similar to Figure 1. The horizontal extent of the images is user selectable using a movable and resizable bounding box. The east-west and north-south components of water currents and surface winds are displayed as vectors. Data with a vertical dimension is depicted on the model's native sigma coordinate as well as being interpolated to standard depths and then displayed.

The final step in the HAZMAT use case allows the customer to retrieve a selected data subset and save it to their computer. The user is able to save a product as either a netCDF file or gif image. The netCDF file would follow the Climate and Forecast (CF) standard[2] to promote ease of use by customer applications such as the GNOME.

[2] The CF convention web site is http://www.unidata.ucar.edu/software/netcdf/conventions.html

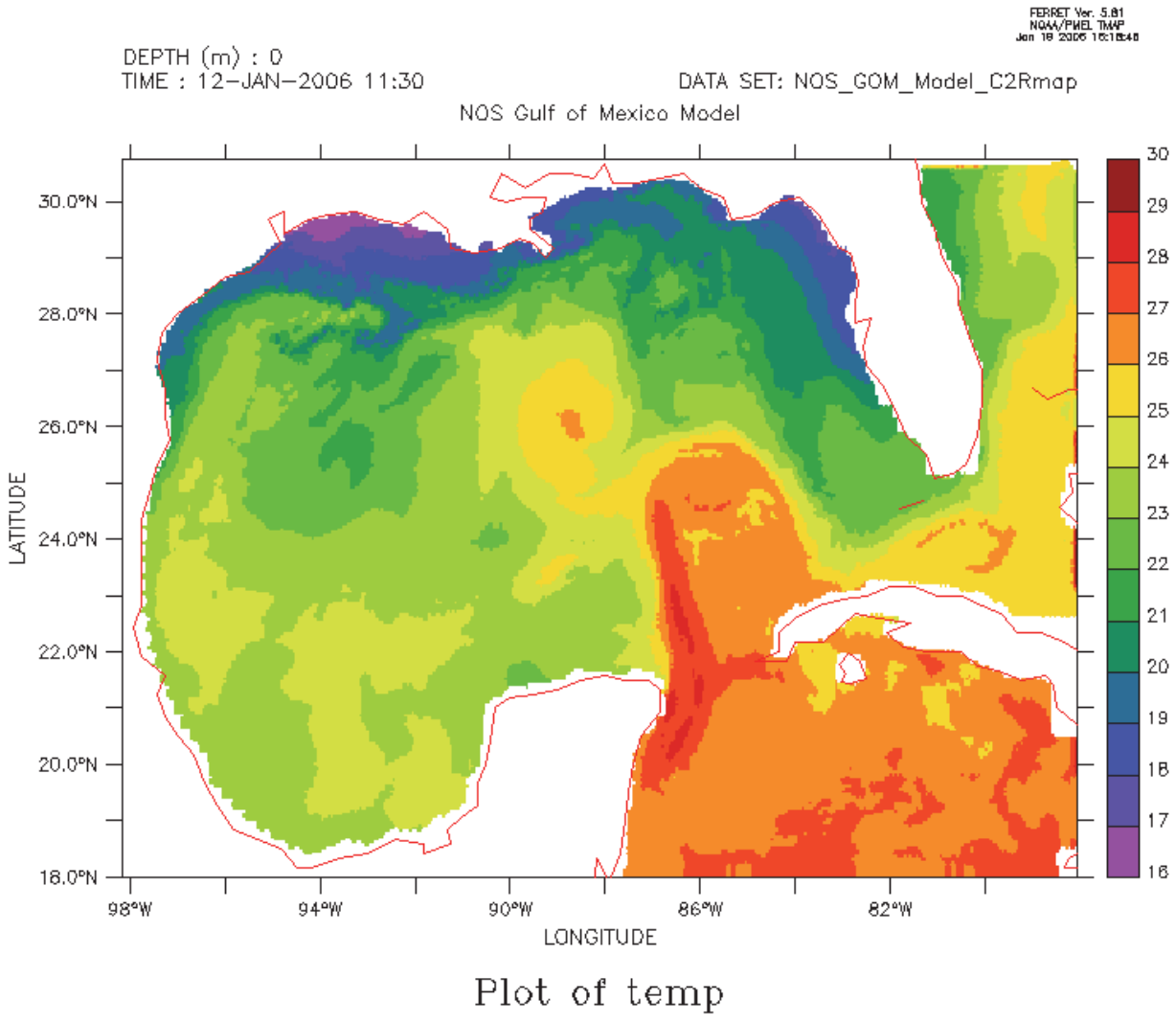


Figure 1. NGOM Surface Temperature

1.3. LAS Architecture

An LAS is composed of a UI server, product server, and a data server (see Figure 2). The UI helps the user by providing them menus and graphical data selection tools. By providing only valid options in the menus, the user needs to possess only a minimal understanding of the dataset. The menu options are determined from configuration metadata files whose content is derived from the dataset. Since the metadata, which is stored as Extensible Markup Language (XML), changes from data file to data file, the configuration files must be dynamically created. For example, the valid times of the data vary from dataset to dataset.

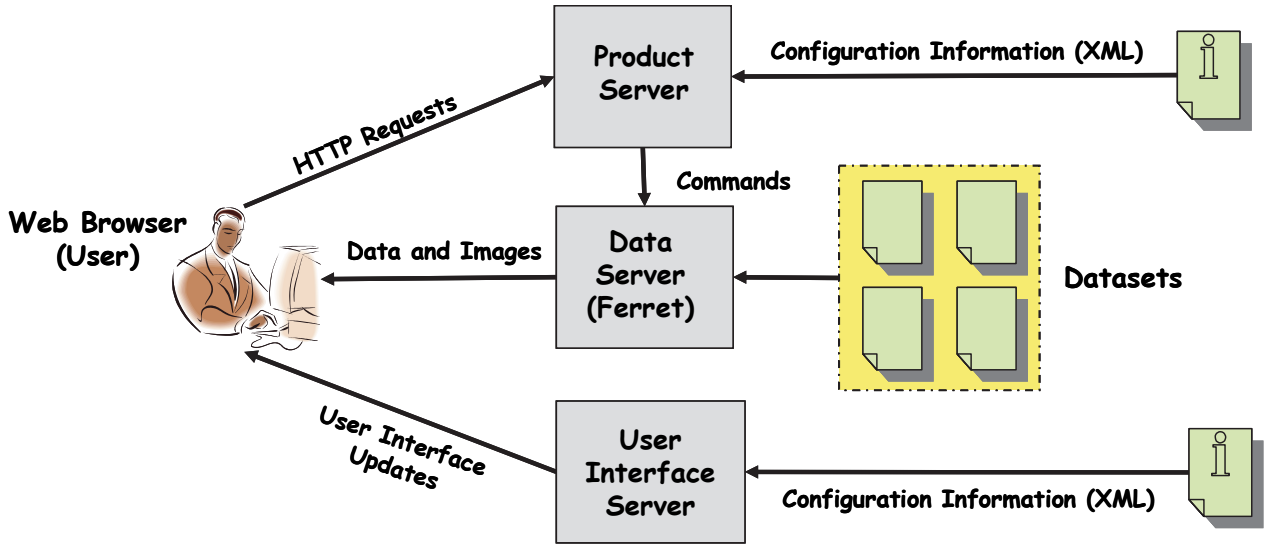


Figure 2. LAS Architecture

To enable user interaction with the UI, Java and JavaScript code is sent to the client computer. Java is used to control the interactive map described in Section 2.1.6.3.2. JavaScript, on the other hand, generates the web pages that the user sees in their browser window (Section 2.1). Configuration files provide the UI with the names and content of available datasets which it then displays in the appropriate web page. As the user selects certain UI fields, HTTP requests are sent to the product server. For example, the user will select the spatial and temporal characteristics of the data they want displayed. The product server translates the user's request, using metadata about the dataset, into instructions that the data server understands.

Ferret[3], the data display application used as the CSDL LAS data server, interprets the instructions from the product server, accesses the needed dataset, and creates the requested product. Ferret then returns the product to the UI created browser window. Ferret can return user requested data as an image or a netCDF file. Ferret also invokes custom scripts that allow it to properly process the data including adapting to non-rectilinear data grid structures.

[3] The Ferret home page URL is http://ferret.pmel.noaa.gov/Ferret/

2. LAS FUNCTIONAL DESCRIPTION

2.1. User Interface

The CSDL LAS employs a browser-based UI that steps the user through a series of pages designed to allow them to 1) select one dataset from a collection of datasets; 2) constrain the data in terms of its spatial and temporal extent; 3) choose the output format of the requested data; and, 4) generate the data product. The first step, choosing the dataset (see Figure 3), begins on the LAS home page. This page of the UI presents links to available data collections. The data collection links are the top level of a hierarchical structure grouping datasets into increasingly more specific data collections. By clicking on any of the data collections, the user is taken to the next level of the hierarchy. This process repeats until they reach a dataset page.

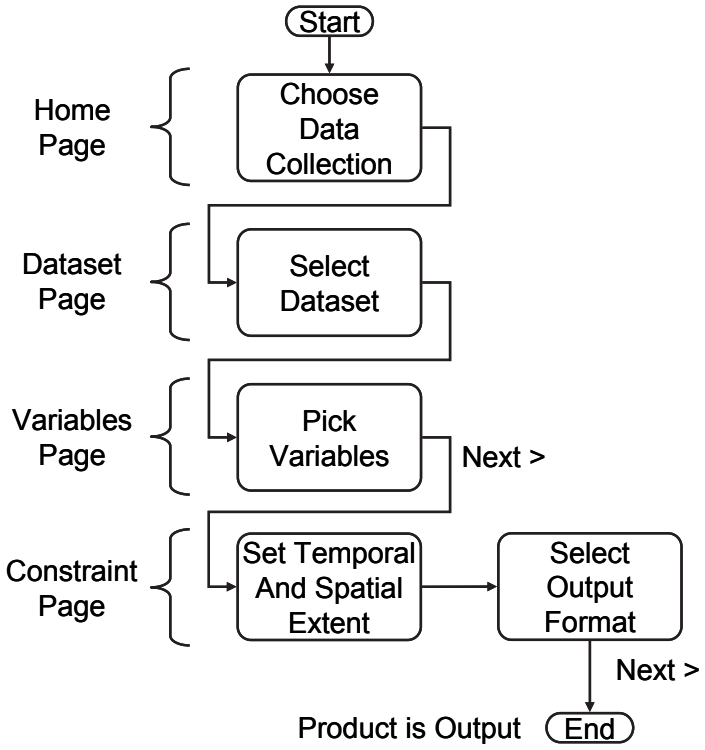


Figure 3. Product Preparation Process

By clicking on a link on the Dataset Page, the user is taken to the Variables Page. This UI page is used to choose one or more variables for inclusion in the output product. The user then clicks on the red *Next* link go to the Constraint Page. The Constraint Page allows the user to specify what part of the dataset they want and what output format the data product should be in. Selecting the red *Next* link generates the requested output. More detail on using the UI to select, constrain, and output data can be found in the Sections 2.1.3 - 2.1.6.

2.1.1. Navigation Bar

The Navigation Bar (Figure 4) offers an alternative product generation method to the one described in Section 2.1. Located on the left side of the LAS UI, the Navigation Bar, with links labeled *Dataset*, *Variables*, and *Constraints*, enables the user to move directly to the UI pages used to specify 1) the desired dataset; 2) variables wanted within a selected dataset; and 3) the spatial and temporal constraints on the data. Once these pages have been used to define the desired product, the user clicks on the Constraints Page's red *Next* link to initiate output generation. The Navigation Bar allows the user to go back to any step in the product generation process to change UI entries.

In addition to the three navigation items discussed above, the Navigation Bar contains the *Previous Output* option which allows the user to review and select from a list of links that create previously generated products (see Section 2.1.1.4 for more details). The Navigation Bar items *Define Variable* and *Compare Two* are not currently used in the CSDL LAS.

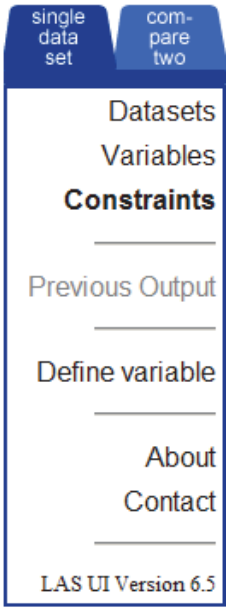


Figure 4. CSDL LAS Navigation Bar

2.1.1.1. Datasets Link

Clicking on the *Datasets* link displays all data collections available to the LAS. Selecting this link is normally the first step in creating a data product. However, the link can be used at any time to change datasets. Only one dataset may be selected at a time. See Section 2.1.3 for more information on the Dataset Page.

2.1.1.2. Variables Link

The *Variables* link is used to go directly to the UI's Variables Page. A user will typically select this link after choosing the dataset. However, it can be used to change variables or to correct a problem involving the selection of too many or the wrong variables for a particular data product. See Section 2.1.5 for more details on the Variables Page.

2.1.1.3. Constraints Link

The C*onstraints* link takes the user straight to the UI's page for setting data constraints. Constraints are highly variable specific, so the user will have had to already select at least one variable to use this page. Subsequent visits to this page allow the user to choose another set of constraints including possibly a different output format. The user can find more information on the Constraints Page in Section 2.1.6.

2.1.1.4. Previous Output Link

This link displays a UI page containing links to all data products generated during the current session. The first link from the top of the page is the most recent product generated. When clicked, the links will cause the LAS to once again create the product. This can be useful if the window containing a previously generated product is mistakenly dismissed.

2.1.1.5. Define Variable Link

This link is not functional in the CSDL LAS. Activating the link is part of CSDL's future development plans.

2.1.1.6. About Link

This link takes the user to PMEL's LAS home page. This page contains information on LAS software, including how to customize and extend it. The FAQ link on this page is a particularly good source for information on customizing the application. A searchable mail archive is also available by clicking the *About* link.

2.1.1.7. Contact Link

The *Contact* link provides e-mail addresses where LAS users can send comments. These addresses are established using the LAS configure script. See Appendix D1 for more details on how to run the configure script.

2.1.2. Compare Two Datasets Tab

This link is not functional in the CSDL LAS. Activating the link is part of CSDL's future development plans.

2.1.3. Discovering Datasets

The CSDL LAS is configured to group datasets into collections sharing a common source and set of variables. The highest level grouping represents all the different sources of data at CSDL. The NGOM model is an example of a distinct source of data. The next level down in the hierarchical structure contains groupings based on other distinctions, such as variables the dataset contains or the manner that the data was produced. Figure 5 illustrates how the NGOM model data source is structured. The figure shows the NGOM collection composed of four sub-collections. These sub-collections represent the model run in 1) forecast mode with variables defined at the ocean surface; 2) nowcast mode with variables defined at the surface; 3) forecast mode and containing variables valid at various depths; and, 4) nowcast mode and containing variables valid at various depths.

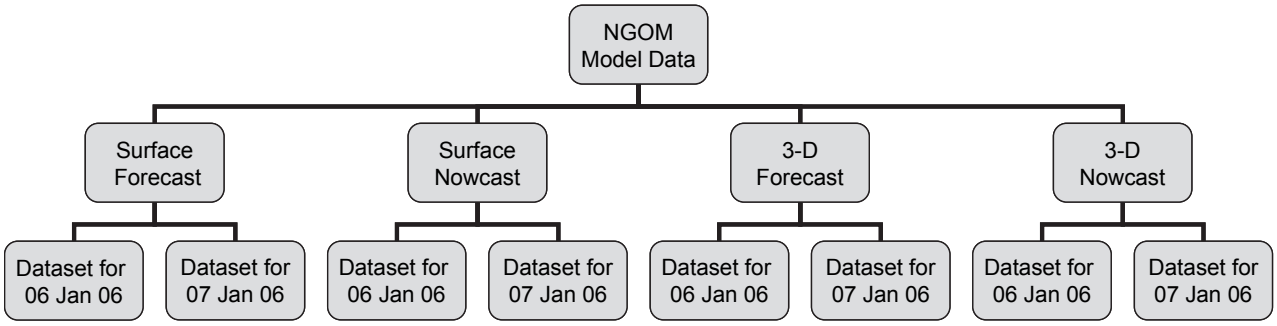


Figure 5. Sample Data Collection Hierarchy

2.1.3.1. Data Source Page

The Data Source Page (see Figure 6) contains links to CSDL model and other sources of data that are available to the public. The page provides the user with information on what CSDL data holdings contain and helps them quickly locate the data they want to explore. Each text field is a link to the next level of a hierarchical structure. The user clicks on the link to go to the next level in the hierarchy.

The text on the Data Source Page is created by the configuration script createCATxml.php and can be altered as needed. Appendix B6 contains more information on createCATxml.php.

2.1.3.2. Sub-collection Page

The Sub-collection Page further organizes CSDL data to help the user understand dataset content. This is an optional level in the hierarchy and is only used when the data logically falls into two or more sub-categories. In Figure 7, there are four sub-collections (3-D Forecast, 3-D Update, Surface Forecast, and Surface Update) which provide natural groupings of datasets. Sub-collections help reduce the number of datasets the user has to review and provide a description of the data they contain. Although there is currently only one sub-collection page for each data source, the LAS could use additional sub-collection pages to further categorize complex data.



Figure 6. CSDL LAS Data Source Page

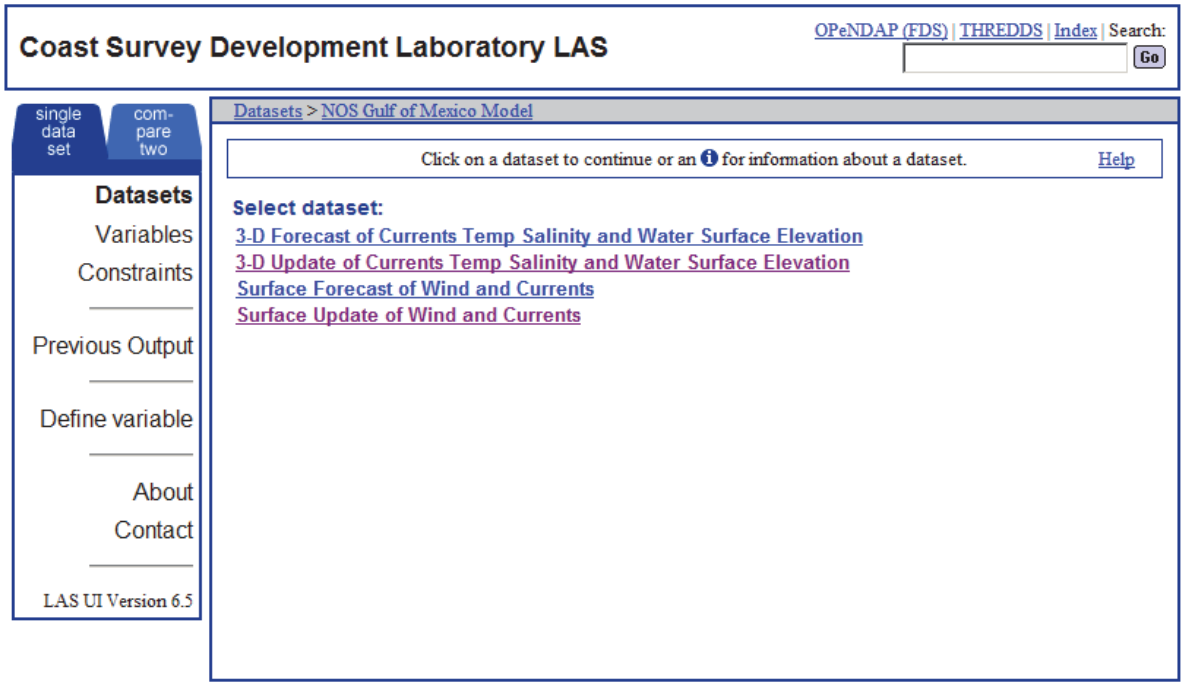


Figure 7. CSDL LAS Sub-collection Page

The text on the Sub-collection Page is created by the configuration script createCATxml.php and can be altered as needed. Appendix B6 contains more information on createCATxml.php.

2.1.3.3. Dataset Page

The Dataset Page (Figure 8) is the lowest level in the data hierarchy. Each link represents a dataset and typically corresponds to a file on disk storage. By clicking on the link, the user tells the LAS that it is to process the contents of that file. This accomplishes the first step of the product generation process. Clicking on the link also takes the user to the Variables Page which continues the product generation process.

In Figure 8 there are three different types of datasets. The first type has names that look like "ng105_06Jan12U3D." These datasets will display data on the NGOM model's sigma vertical coordinate system. The dataset type with names like "ng105D_06Jan12U3D" will result in the model data being displayed on calculated depth surfaces. The physical file for this dataset is the same one used for the sigma coordinate dataset. The final type of dataset is the aggregate and its name is similar to "ngAggregateU3D." This dataset is a virtual dataset consisting of the contents of all the sigma coordinate datasets and is created by the OPeNDAP Aggregation Server.

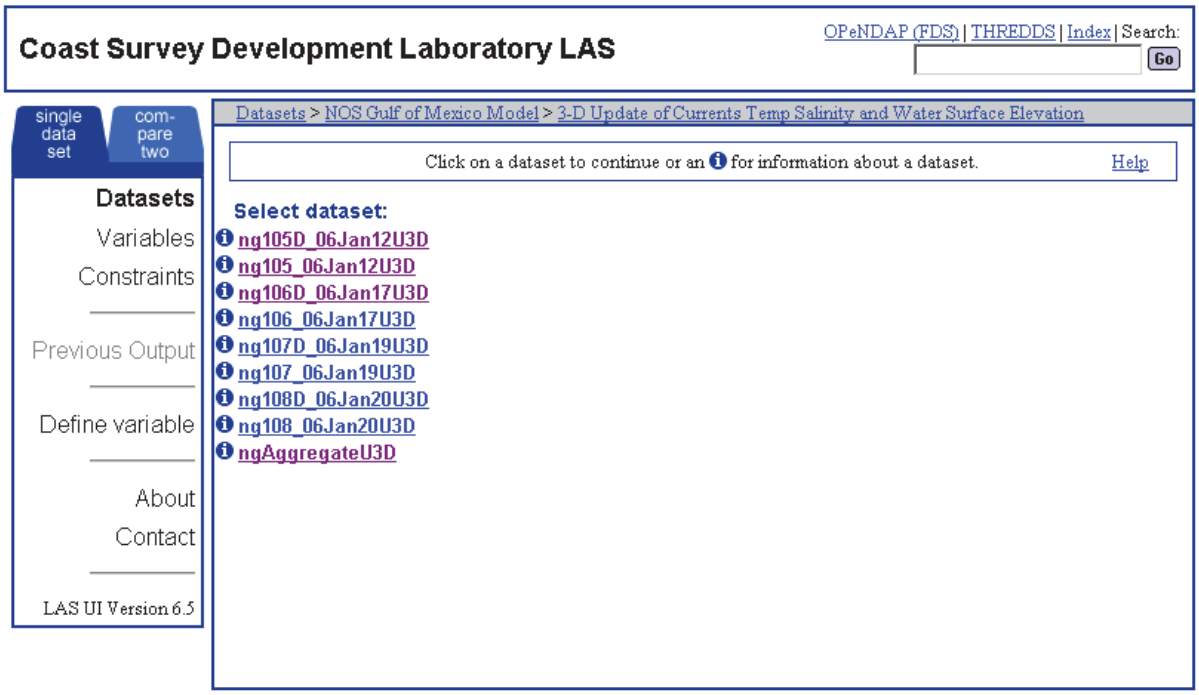


Figure 8. CSDL LAS Dataset Page

The text on the Dataset Page is created by the configuration script createCATxml.php and can be altered as needed. Appendix B6 contains more information on createCATxml.php.

2.1.4. Dataset Quick Links

Located in a gray shaded area below the LAS title bar are the dataset quick links (see Figure 9). There is a link for the Data Source Page, any subsequent sub-collection pages, and finally the currently selected dataset. Clicking on any of these links will take the user back to that level in the data hierarchy.

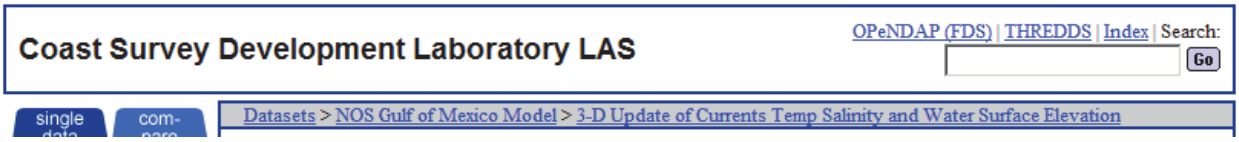


Figure 9. Dataset Quick Links

2.1.5. Variables Page

The Variables Page (see Figure 10) allows the user to select the variable or variables they want the LAS to use for their output product. The user selects or de-selects a variable by clicking on the check box to the left of the variable name. Certain output products that the user can create require a specific number of items to be checked on the variables page. For example, the user must only select one variable if they plan to create a latitude and longitude plot. When creating a netCDF file, the user can select one or more variables. Once the user has selected the appropriate variables, they then click on the red *Next* link to go to the next UI page.

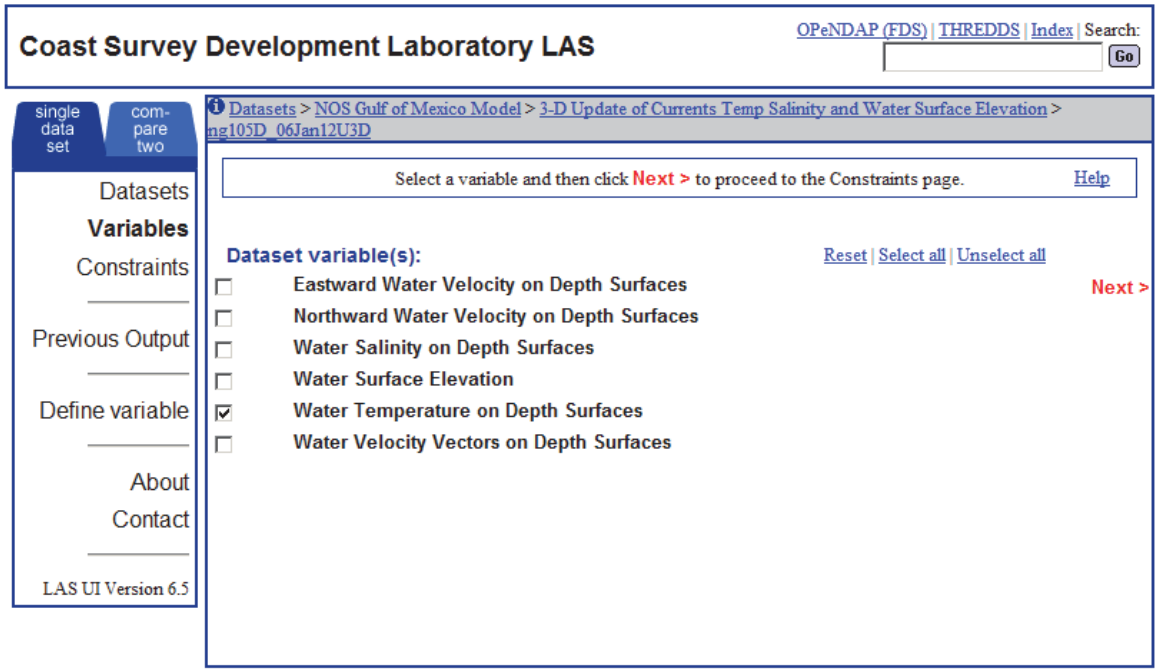


Figure 10. CSDL LAS Variables Page

Variables appearing on the Variables Page are derived from the dataset's XML configuration file located in the LAS server directory. These files are dynamically created by the script createDGOMxml.php. Appendix B7 describes the script in more detail.

2.1.6. Constraints Page

The Constrains Page (see Figure 11) is perhaps the most complicated of the UI pages and is used to control which part of the dataset is displayed or saved. The user controls the product content by using the *Select view*, *Select output*, *Select region*, *Select time*, and *Select depth* menus. The *Select region* menu is augmented by a Java map applet that allows the user to select the horizontal area for which they would like data. Each of these menus is discussed further in Sections 2.1.6.1 – 2.1.6.5. The *Select Option* menu is used to control the appearance of the output. This menu is not used by the CSDL LAS currently but is in the plan for future development. Once the user has either accepted the default settings or selected an item from each pull down menu or the map, they then click on the red *Next* link to generate the desired product. The product will appear in a new window on the user's computer.

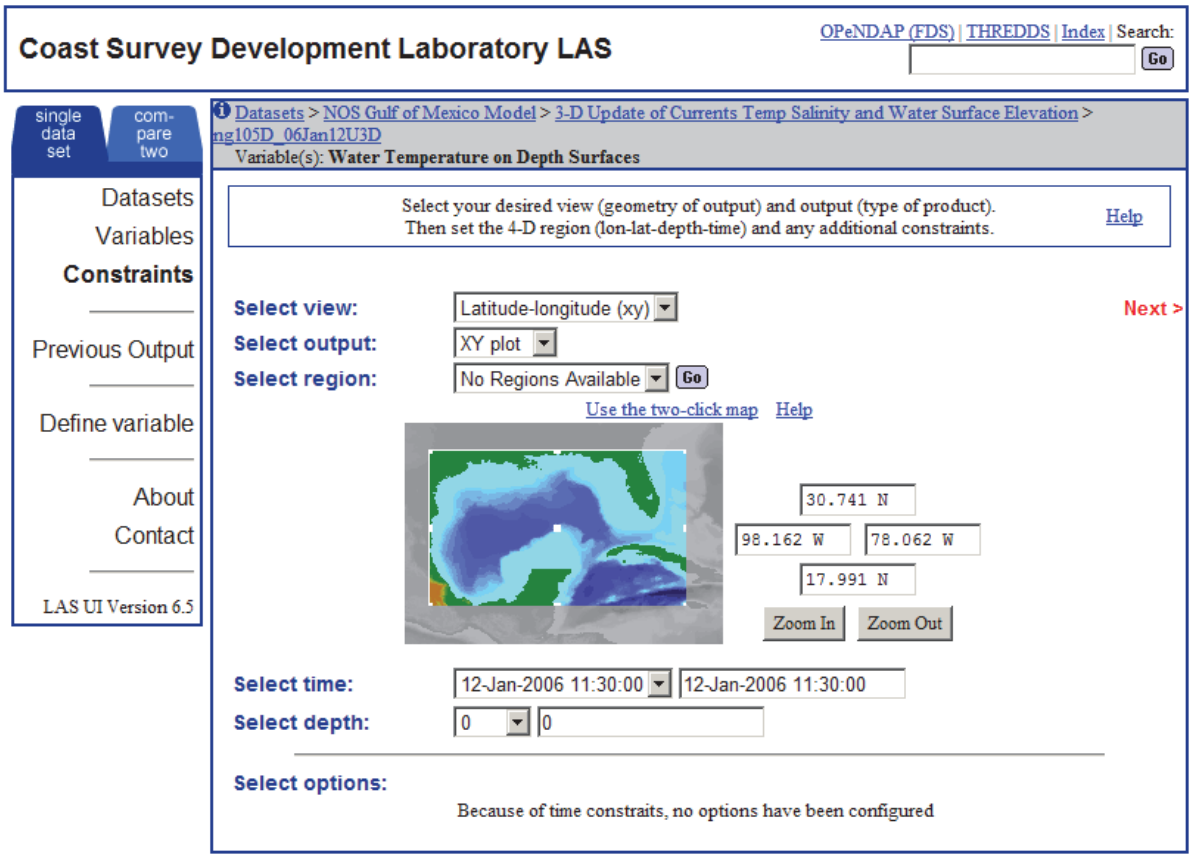


Figure 11. CSDL LAS Constraint Page

2.1.6.1. Select View

The *Select view* menu allows the user to choose from a list of predefined slices of the data. Each variable has its applicable set of views. For example, Eastward Air Velocity can be visualized along the data's 1) latitude and longitude axes, 2) latitude, longitude, and time axes, and 3) time axis. Table 1 shows the slices available to each NGOM variable. Selecting a particular view will result in changes to items available in other menus on the page. Therefore, it is best to define the view first.

Table 1. NGOM Variables and Corresponding Views and Output Formats

Variables	Views	Output Formats
Eastward Wind Component Northward Wind Component Eastward Current Component Northward Current Component	Longitude-Latitude Longitude-Latitude & Time Time	.gif image netCDF file
Wind Vectors Current Vectors	Longitude-Latitude	.gif image
Salinity Temperature Water Surface Height	Longitude-Latitude Longitude-Latitude & Time Time Longitude-Latitude & Depth	.gif image netCDF file

The views available to each variable can be modified in the UI configuration file HAZMAT_ui.xml. To alter the options appearing in the menu, add or remove xml tags like <menu type="views">. One tag may be added for each data axis or combination of data axes. The file HAZMAT_ui.xml is located in the LAS server directory /usr/local/las/server.

2.1.6.2. Select Output

The *Select output* menu allows the user to select from appropriate output formats. As Table 1 shows, the allowable formats vary with each variable. The .gif image is initially displayed in a browser window. The LAS does not provide a mechanism to save the image. However, the browser displaying the image may have a *Save Picture As…* option. The other output option allows the user to save the data in a CF compliant netCDF file. When the product is generated, the user is prompted with a dialog box offering them the option to save the file to their computer. Figure 12 shows a sample image product.

Output formats associated with a particular data variable can be changed by software maintainers. Changing these formats is accomplished by editing the UI configuration file HAZMAT_ui.xml. To alter the options appearing in the menu, add or remove xml tags like <menu type="ops">. For each added xml tag, there needs to be a corresponding entry in the UI configuration file HAZMAT_operations.xml. The files HAZMAT_ui.xml and HAZMAT_operations.xml are located in the LAS server directory /usr/local/las/server.

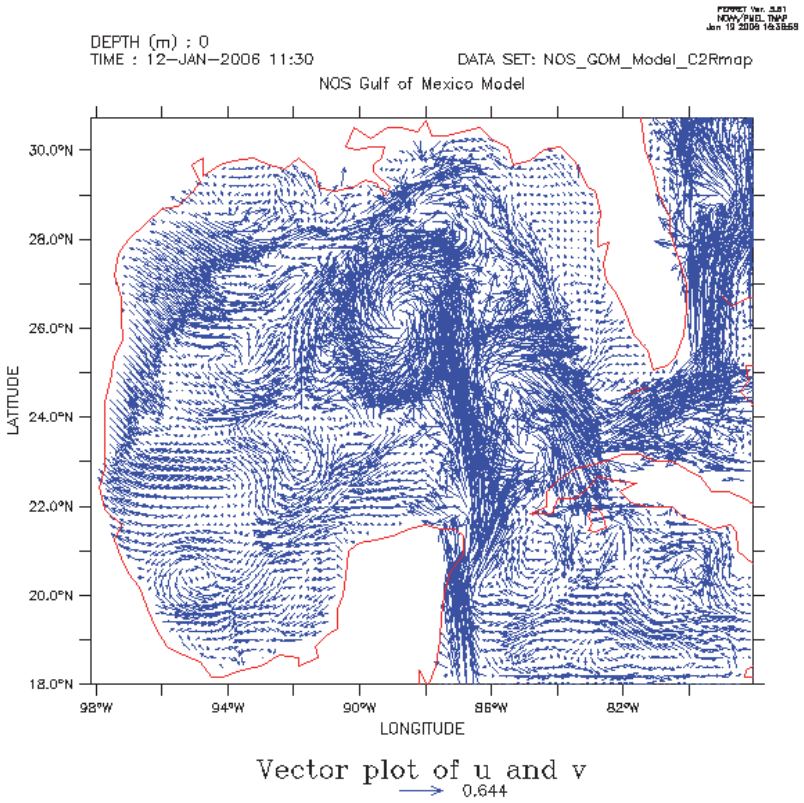


Figure 12. Sample Output Product

2.1.6.3. Select Region

2.1.6.3.1. Choosing Predefined Regions

The CSDL NGOM data collection has no predefined regions.

2.1.6.3.2. Creating User Defined Region

Rather than using predefined regions, the user employs the interactive Java map to define their area of interest (Figure 13). Selecting an area is accomplished by moving any of the eight small white squares located on the white box's perimeter. This should allow the user to resize the white box to fit the area for which they desire data. The user can also use the white square in the middle of the white box to translate the box around the map area.

The image displayed by the Java applet can be configured in the UI configuration file HAZMAT_ui.xml by modifying the *url* attribute of the <images> tag. The following example shows the current <images> tag.

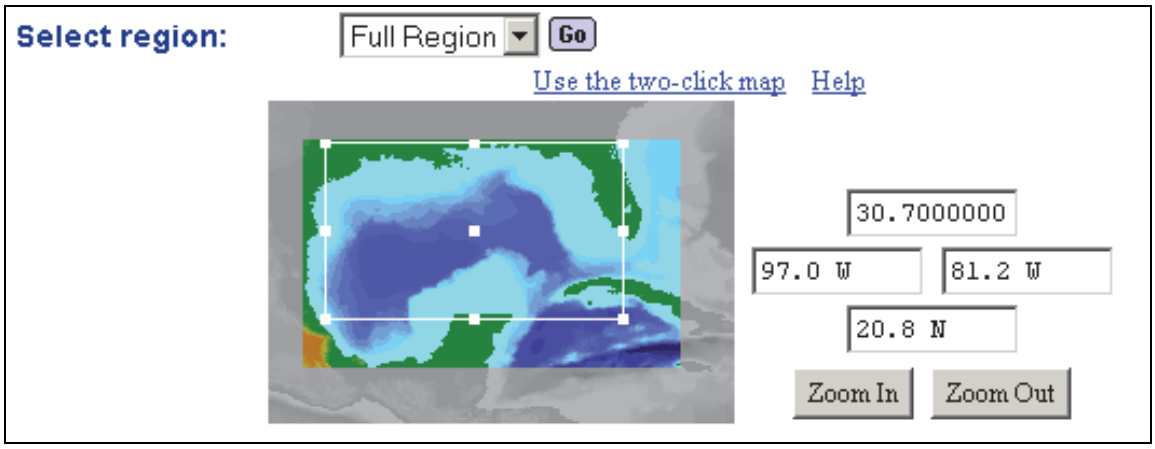


Figure 13. Interactive Java Map

```
<images>
  <image name="Image_HAZMAT_GOM" url="gifs/hz_map2.gif" bounds="-100,-75,15,33"/>
</images>
```

The file hz_maps2.gif needs to be in the directory /usr/local/las/las_servlet/web/classes/gifs/.

2.1.6.4. Select Time

The *Select time* menu allows the user to tell the LAS which valid times to use. For views where time is not one of the axes, the *Select time* menu allows the user to pick only one time value. But when time is an axis in the view, the *Select time* allows the user to choose both a start and end time.

Time values that appear in the menu are dynamically created by the script createDGOMxml.php. The script createDGOMxml.php enters values in the dataset configuration file's time axis definition. A sample definition follows:

```
<ngom05Nov033D_nc_time type="t" units="seconds">
  <v>03-Nov-2005 13:30</v>
  <v>03-Nov-2005 16:30</v>
  <v>03-Nov-2005 19:30</v>
</ngom05Nov033D_nc_time>
```

2.1.6.5. Select Depth

The *Select depth* menu allows the user to choose the vertical location of the data they want the LAS to process. For views where the vertical axis is not part of the slice, the *Select depth* menu only allows the user to select one vertical location. If the vertical axis is part of the slice, then the *Select depth* allows the user to pick the start and end vertical locations. If the selected variables do not have a vertical dimension, then the *Select depth* menu is omitted from the page.

Vertical coordinate values in the menu are dynamically created by the script createDGOMxml.php. There are two types of vertical coordinates that are generated by the script createDGOMxml.php. The model's sigma coordinate is one vertical axis type. Depth from the ocean surface is the other vertical axis type. The script createDGOMxml.php first enters values in the sigma-based configuration file's vertical axis definition. The script next creates another configuration file where native sigma coordinate values have been replaced with standard depths in meters. These configuration files point to the same physical file as the sigma coordinate version and are similar in all other respects except for the vertical coordinate.

When the user selects a dataset with a depth vertical coordinate, the LAS knows to use a specialized Ferret script to process the data. The dataset's configuration file tells the LAS to use non-standard Ferret scripts to process the data. This use of custom software is controlled by the following XML tags:

```
<ferret>
  <script_prefix>HAZMATstd_</script_prefix>
</ferret>
```

The prefix "HAZMATstd_" is used by the LAS to determine the names of Ferret scripts it should use to process the dataset. These custom scripts are in the directory /usr/local/las/server/custom. See Section 3.3 for more details on custom Ferret scripts.

2.1.6.6. Select Options

This menu is not functional in the CSDL LAS. Activating the menu is part of CSDL's future development plans.

2.2. Use Cases

2.2.1. Data Discovery and Display

A user of CSDL data will want to locate and explore datasets available on the LAS. They will begin at the LAS home page (Figure 6) which is also called the Data Source Page. Using the data source's title, the user will select a source that seems appropriate for their interests. The UI will then present them with a more precise description of the contents of the data collection (Figure 7). The user will select a link which takes them to the Dataset Page (Figure 8). Selecting a link on the Dataset Page takes the user to the Variables Page (Figure 10). This page shows the user all the variables in the dataset. Selecting one of the check boxes on the Variables Page tells the LAS that it is to use that variable in subsequent actions. Once the desired box is checked, the user then selects the red "*Next*" link to transition to the Constraints Page (Figure 11). Here the user selects from pull down menus that tell the LAS how to subset the dataset and which product is desired. If the user has selected to output a vector plot, clicking on the red "Next" link brings up a new window containing the requested image (see Figure 1 for an example).

2.2.2. Data Retrieval

To download a subset of a dataset, the user will follow the steps outlined in Section 2.2.1 for locating the dataset of interest. Once the user has reached the Dataset Page, they will select one or more variables for inclusion in the returned file. Clicking on the red *Next* link, the user is taken to the Constraints Page where they use the *Select view* pull down menu to choose either the *XYT volume* or *XYZT volume*, depending on whether the data is two- or three-dimensional. Following this action, the user should see *NetCDF* in the *Select output* pull down menu. The user can then further constrain the data using the *Select region*, *Select time range*, and possibly (if the data is three-dimensional), "*Select depth*". By clicking on the red *Next* link, the user is presented with a window allowing them to save the requested data to their computer.

3. PREREQUISITE AND SUPPORT SOFTWARE

The CSDL LAS derives considerable functionality from a number of software applications and libraries. This code is installed on the LAS server and configured according to the software's documentation. Each version of the LAS is usually tested against one version of this support software, so care should be exercised when upgrading them outside an LAS upgrade. A complete list of software needed by the LAS (version 6.5.1.2), including appropriate version numbers and where the software can be obtained, is available in Appendix C1 and C2.

3.1. Automated Customization Scripts

These scripts were written by MMAP and dynamically create LAS configuration files. The configuration files are set up to reflect unique dataset information such as the date and time when the data is valid. The scripts are located in the LAS directory /user/local/las/scripts. These scripts are run twice daily using a UNIX crontab. The crontab is run as user root.

When the main script, setupLAS.sh executes, NGOM data files are copied to the LAS data directory (/data/dgom). The script ensures that only datasets that are newer than a certain threshold (i.e. 30 days) will get copied and be made available to the public. Configuration files for these copied files are created on the fly and stored in the server directory (/usr/local/las/server).

In addition to creating LAS dataset configuration files, setupLAS.sh also creates configuration files that establish the LAS hierarchical dataset structure presented to the user. Figure 6 shows the top level of the hierarchy of links. To logically merge separate datasets into a combined dataset, setupLAS.sh creates OPeNDAP Aggregation Server configuration files. These files tell the Aggregation Server which datasets to include in the virtual grouping and how to join them. The CSDL LAS joins all the nowcast runs of the NGOM model into a single virtual file that spans the data's time axis. See Appendix B4 and B5 for more details on individual configuration scripts.

3.2. Ferret

Ferret[4] is the data visualization backend of the LAS (see Figure 2). It was developed by the Thermal Modeling and Analysis Project (TMAP) group at PMEL. The Ferret software is located on the LAS server in the directory /usr/local/ferret. The application is invoked by the LAS Data Server when the user requests a product. The data server passes the appropriate commands to Ferret, which then sends the visualization or data file to the user's browser. Ferret is designed to be OPeNDAP aware. As a result, the LAS provides Ferret with the data location specified as an OPeNDAP URL. Ferret is a highly configurable and capable graphics package. The Ferret User's Guide[5] can help exploit these capabilities.

[4] The Ferret Home Page can be found at http://ferret.pmel.noaa.gov/Ferret/
[5] The Ferret User's Guide is available at http://ferret.pmel.noaa.gov/Ferret/Documentation/ferret_documentation.html

3.3. Custom LAS Scripts

CSDL NGOM data is based on a curvilinear grid. Unfortunately, the LAS UI is predicated on a rectilinear input grid. To bridge the differences in spatial representations, CSDL developed custom Ferret scripts to convert the LAS rectilinear latitude and longitude values into curvilinear space. For visualization products, the image is displayed in the LAS's native rectilinear coordinate system. However, netCDF file products are written with the data in curvilinear space. The Ferret scripts are found in the directory /usr/local/las/server/custom. Table 2 shows the custom scripts and which products they support.

Table 2. Custom Ferret Scripts

Script	Description	Product
HAZMAT_plot_xy.jnl curvi_region_subset.jnl	Displays a latitude/longitude image in rectilinear coordinates using a curvilinear grid as input. The vertical axis is on model native sigma levels.	Image (.gif) [sigma]
HAZMATstd_plot_xy.jnl curvi_region_subset_STD.jnl	Displays a latitude/longitude image in rectilinear coordinates using a curvilinear grid as input. The vertical axis uses standard depths found in Table 5.	Image (.gif) [depth]
HAZMAT_list_cdf.jnl	Creates a netCDF file by translating LAS rectilinear latitude and longitude values to their corresponding curvilinear values.	File (netCDF) [sigma]
HAZMATstd_list_cdf.jnl	Creates a netCDF file by translating LAS rectilinear latitude and longitude values to their corresponding curvilinear values. The vertical axis uses standard depths found in Table 5.	File (netCDF) [depth]
HAZMAT_std_gif_vectorref.jnl vector_subset.jnl	Displays a latitude/longitude vector image in rectilinear coordinates using a curvilinear grid as input. The vertical axis is on model native sigma levels	Image (.gif) [sigma]
HAZMATstd_std_gif_vectorref.jnl vector_subset_STD.jnl	Displays a latitude/longitude vector image in rectilinear coordinates using a curvilinear grid as input. The vertical axis uses standard depths found in Table 5.	Image (.gif) [depth]

3.4. Software Update Scripts

As it becomes necessary to modify customization and custom Ferret scripts, software update scripts are used to implement these changes using a systematic and repeatable process. The scripts are located in the directory /usr/local/las/install. For more information on installing custom scripts, see Appendix A1.

3.5. OPeNDAP Server

The LAS runs on the same machine as the CSDL OPeNDAP server. Since Ferret is OPeNDAP aware, the LAS has been configured to make its dataset access attempts using OPeNDAP. This integration with the OPeNDAP server is depicted in Figure 14.

The OPeNDAP server is a stand alone capability that customers can use independent of the LAS. A user with a browser can explore CSDL data offerings at http://dods.ncd.noaa.gov/index.php without ever using the LAS. The user can also use the OPeNDAP server to retrieve CSDL datasets. However, the LAS becomes an important alternative to the stand alone OPeNDAP server when the user is not familiar with OPeNDAP URLs or does not have an OPeNDAP aware graphics package. More information can be found on OPeNDAP at its home page http://www.opendap.org/index.html.

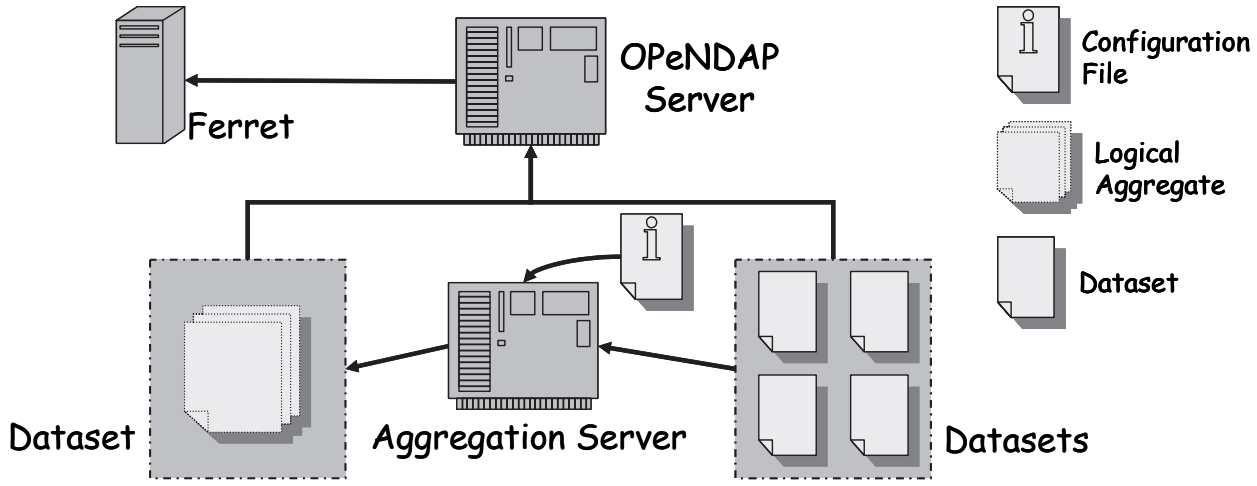


Figure 14. Aggregation Dataset Architecture

3.6. OPeNDAP Aggregation Server

Unidata expanded on the OPeNDAP server by providing a mechanism to catalog datasets. This service, called Thematic Real-time Environmental Distributed Data Services (THREDDS), allows the CSDL LAS to aggregate many datasets into a single virtual dataset (see Figure 14). The LAS then displays the appropriate date and time information as if these files were one file.

The Aggregation Server is configured by the customization script createAggregatexml.php. This script creates the aggregation configuration file that identifies which datasets are to be aggregated, what the aggregation is to be called, and that they are to be unified along the time axis. For more information on createAggregatexml.php, see Appendix B4.

3.7. Tomcat Servlet Engine

The Tomcat Servlet Engine[6] is used as an interface between the Apache Web Server and the Java code in the LAS and Aggregation Server. While there can be more than one LAS, each should use the same Tomcat servlet. This is accomplished by using the LAS configure script to setup the LAS and is designed to eliminate the overhead associated with multiple Tomcat servlets. See Appendix D1 for information on LAS setup. Tomcat is run at http://dods.ncd.noaa.gov:8080/.

3.8. Other Applications

3.8.1. Apache Web Server

The Apache Web Server communicates with the user's web browser using Hypertext Transfer Protocol (HTTP). The Apache Web Server then makes the user request available to the LAS. The Apache configuration file /etc/httpd/conf/httpd.conf contains alias and proxy information for the public and development LAS as well as the Aggregation Server. For more information on the Apache Web Server, see http://httpd.apache.org/.

3.8.2. Perl

Perl is the programming language used to write the LAS UI and Data Server. There are a number of specialized modules that the LAS needs. See Appendix C1.2.1 - C1.2.7 for a list of these modules. For more information on Perl, see http://www.perl.org/.

3.8.3. MySQL

The MySQL database is used to hold LAS server information. When the LAS configuration script runs it is important to know the MySQL host, user, and password that the LAS will use to communicate with the database.

---

[6] The Tomcat Home Page is http://tomcat.apache.org/

4. FUTURE DEVELOPMENT

Due to the limited amount of time provided by his rotational assignment, the MMAP LAS developer was only able to configure the application to expose the NGOM data collection. In the future, CSDL will add additional existing data collections to the LAS's public offerings. These data collections will include digital elevation model and bathymetric information.

In addition to expanding the data available via the LAS, CSDL plans to extend the software's functionality to include the ability to define custom user variables, compare variables between two CSDL datasets, and provide the customer more options for controlling the format of output images.

Finally, CSDL plans to use the LAS to provide access to CO-OPS PORTS operational model data. This will require serving up remotely held data via DODS[7].

[7] See LAS Frequently Asked Question http://ferret.pmel.noaa.gov/Ferret/LAS/FAQ/remote_data.html

5. ACKNOWLEDGEMENTS

This software development project was made possible by an employee assigned to MMAP through the NOAA Rotational Assignment Program (NRAP). We appreciate the support of NOAA/NWS/MDL for their participation in the NRAP. Additionally, invaluable LAS technical assistance and custom software development was provided by members of NOAA/PMEL/TMAP (especially Kevin O'Brien and Ansley Manke). Finally, we want to express our appreciation to NOAA/NOS/CO-OPS for sharing with us their OPeNDAP server index script.

APPENDIX A: CUSTOMIZATION SOFTWARE INSTALLATION INSTRUCTIONS

A1. Introduction.

The LAS requires numerous XML configuration files to function. These files contain information that must be dynamically created using the content of the datasets being configured. This on-the-fly formatting of configuration files is accomplished by a series of bash and PHP scripts. The scripts will from time to time need to be modified, tested, and installed. This appendix documents the procedures to develop and install changes to the customization scripts as well as other user developed components of the LAS.

A2. Key Concepts.

A2.1. There are two LASs implemented on the same host (dods.ncd.noaa.gov). The public LAS is located in the directory /usr/local/las/ and the development version is located in the directory /usr/local/las_dev. Changes to customization scripts, user developed Ferret scripts, and other user developed components in the development LAS directory will not affect the LAS used by CSDL customers. The two LASs do, however, share a common Tomcat servlet. This is done for processing efficiency reasons.

A2.2. Installation scripts are controlled by environmental variables in the file myLAS.env located in each LAS home directory (/usr/local/las and /usr/local/las_dev). Care should be exercised to ensure that these files remain up-to-date and contain appropriate paths for their respective file areas.

A2.3. The installation scripts located in the directories /usr/local/las/install and /usr/local/las_dev/install are not part of the release and as a result it is the developer's responsibility to ensure that the latest versions of these scripts are in both the development and production install directories.

A3. Development Installation.

A3.1. Creating the Release. Once changes to user controlled software have been tested, a script named create_LAS_release, located in the directory /usr/local/las_dev/install, is run. The script is executed, as shown below, using the sudo command and a single argument for the release's version number (*version*).

```
cd /usr/local/las_dev/install
sudo ./create_LAS_release version
```

This script creates an archive of the files contained in the two directories found in Table 3. The archive is placed in the directory /usr/local/las_dev. The script chronicles the steps it takes in a log file located /usr/local/dods_logs/ directory. The log file name is of the form

LAS_DEVELOPMENT_Install_1214-15:24.

Table 3. Releases Directories

Directory	Contents
/usr/local/las_dev/scripts	Contains the scripts used to dynamically create LAS configuration files.
/usr/local/las_dev/server	Contains the LAS configuration files and user created custom Ferret scripts.

A3.2. Installing the Release. Installing the release can be tested in the development area by executing the following commands:

cd /usr/local/las_dev/install
sudo ./install_LAS *version type_install*

The script takes as input the version number (*version*) assigned in Section A3.1 and the install type (*type_install*) which in this case is "DEVELOPMENT". When the script runs it logs its actions in the directory /usr/local/dods_logs/ with a file name something like LAS_DEVELOPMENT_Install_1214-15:24. The log file should indicate that the previous contents of the two directories in Table 3 were backed up to directory /usr/local/las_dev/back_up/.

A3.3. De-installing the Release. If the installation or testing of the release does not go well, the installation can be undone using the de-install script. This script restores the contents of the two directories in Table 3 using the archive file in the backup directory (/usr/local/las_dev/back_up/). The de-install script is run as follows:

cd /usr/local/las_dev/install
sudo ./deinstall type_install

The de-install script takes as input the install type (*type_install*) which in this case is "DEVELOPMENT". The script will run and log its actions in the directory /usr/local/dods_logs/ with a file name something like LAS_DEVELOPMENT_Deinstall_1212-11:51.

A4. Production Installation.

A4.1. Creating the Release. The release is created as part of the development process described in Section A3.1. The development release is used for the production installation as described in Section A4.2.

A4.2. Installing the Release. Installing the release is accomplished by executing the following commands where *version* is the number assigned in Section A3.1:

cd /usr/local/las
cp /usr/local/las_dev/LAS_*version*.tar **.**
cd install
sudo ./install_LAS *version type_install*

The installation script takes as input the version number (*version*) and the install type (*type_install*) which in this case is "PRODUCTION". When the script runs, it logs its actions in the directory /usr/local/dods_logs/ with a file name that looks something like LAS_PRODUCTION_Install_1209-14:18. The log file should indicate that the previous contents of the scripts and server directories were backed up to directory /usr/local/las/back_up/. Customization scripts are installed into the directory controlled by the environmental variable las_bin_dir found in the file /usr/local/las/myLAS.env.

A4.3. De-installing the Release. If the install or testing of the release does not go well, the installation can be undone using the de-install script. This script restores the contents of the directories using the archive file in the backup directory (/usr/local/las/back_up/). The de-install script is run as follows:

cd /usr/local/las/install
sudo ./deinstall *type_install*

The deinstall script takes as input the install type (*type_install*) which in this case is "PRODUCTION". The script will run and log its actions in the directory /usr/local/dods_logs/ with a file name something like LAS_PRODUCTION_Deinstall_1212-11:51.

APPENDIX B: CUSTOMIZATION SCRIPTS DESCRIPTION

B1. setupLAS.

B1.1. Purpose. This is the main script that processes all CSDL data sources for inclusion into the LAS. The processing includes moving files from their archived locations to a public server and the creation of configuration files needed by the LAS software. The processing of each data source is handled by a script specifically tailored to its particular dataset content.

B1.2. Input.

B1.2.1. The LAS environmental variables file (/usr/local/las/myLAS.env).

B1.2.2. Dataset names for each data source (i.e. NGOM).

B1.3. Processing.

B1.3.1. Create the log file in the directory /usr/local/dods_logs. The log file will capture what the script did and any error message that may have occurred. The log file name will look something like MMAP_LAS_2006Jan19_143951.

B1.3.2. Execute each driver program which will process a particular data source (i.e. NGOM model output).

B1.3.3. Read in the dataset names that the driver program wrote to the file /usr/local/las/server/dataset_names.txt.

B1.3.4. After processing all the data sources, create the LAS hierarchical data structure by calling the program createCATxml.php. A sample of the hierarchy's first level (the LAS Data Source Page) can be seen in Figure 6. The sub-collection level of the hierarchical structure can be seen in Figure 7.

B1.4. Output.

B1.4.1. Log file in directory /usr/local/dods_logs/.

B1.4.2. Other outputs are described in the driver programs (see B2 - B3).

B2. place_holder.sh

B2.1. Input.

B2.1.1. The LAS environmental variables file (/usr/local/las/myLAS.env).

B2.2. Processing.

B2.2.1. Create log file in the directory /usr/local/dods_logs. The log file will capture what the script did and any error message that may have occurred. The log file name will look something like future_2006Jan19_143952.

B2.2.2. Output several fictitious data collection and dataset names. This is done since the LAS will only display a data collection if there are more than one name in the collection.

B2.3. Output.

B2.3.1. Log file in directory /usr/local/dods_logs/.

B2.3.2. Fake names for what would be the source collection and dataset names. These are place holder names used in creating the hazmat_cat.xml categories file.

B2.3.3. Rebuild the LAS to incorporate the newly created configuration files.

B3. copy_NGOM_files.sh

B3.1. Purpose. This bash shell script is the driver for the NGOM PHP scripts that dynamically configure the LAS. The script's primary function is to move data files to the OPeNDAP server and to create LAS XML configuration files.

B3.2. Input.

B3.2.1. The LAS environmental variables file (/usr/local/las/myLAS.env).

B3.2.2. NGOM model data located in the source directory /disks/NASWORK/dgom/model-transfer

B3.3. Processing.

B3.3.1. Create the log file in the directory /usr/local/dods_logs. The log file will capture what the script did and any error message that may have occurred. The log file name will look something like ngom_2006Jan19_143951.

B3.3.2. Determine which files in the source directory were created within the last 30 days. Only files matching the patterns in the predetermined expressions are retrieved (see Table 4 for possible patterns, matched files, and converted file names).

B3.3.3. For each file determined in Section B3.3.2, create a dataset XML configuration file (i.e. ng288_05Oct04U1D.xml). The content of these files tells the LAS UI what variables the dataset contains and the details of the data's horizontal, vertical, and temporal axes.

Table 4. Data File Name Conversion

Pattern	Source File Name	New File Name
GOMFD*HAZMAT.nc	GOMFD-2006.3S.101.HAZMAT.nc	ng101_06Jan02F1D.nc
GOMFD*3D.nc	GOMFD-2006.3.102.3D.nc	ng102_06Jan03F3D.nc
GOMUD*HAZMAT.nc	GOMUD-2006.3S.101.HAZMAT.nc	ng101_06Jan01U1D.nc
GOMUD*3D.nc	GOMUD-2006.3.100.3D.nc	ng100_05Dec31U3D.nc

B3.3.4. Copy each file to the OPeNDAP server data directory (i.e. /data/dgom). As the file is copied, it is renamed to indicate the valid time of the first value. This naming convention allows the user to determine from the dataset name if it contains the time period they are looking for.

B3.3.5. For each dataset requiring an aggregate dataset, create the aggregate's XML configuration file (i.e. ngAggregateU1D.xml). Only model output from the nowcast runs are continuous in time and are therefore suitable for creating a temporal aggregate.

B3.3.6. Create the aggregate server catalog XML configuration file (catalogConfig.xml).

B3.3.7. Create the LAS XML configuration file (las.xml).

B3.3.8. Write the dataset names (i.e. ng105_06Jan17F3D) to the file provided by the calling program. These names are used to create the LAS categories configuration file (hazmat_cat.xml).

B3.3.9. Clean up old netCDF files in the data directory and XML files in the server directory.

B3.4. Output.

B3.4.1. Log file in directory /usr/local/dods_logs/.

B3.4.2. Dataset XML configuration files in the directory /usr/local/las/server/. File names are of the form ng288_05Oct04U1D.xml.

B3.4.3. Renamed netCDF files with names of the form ng101_06Jan01U3D, copied to the /data/dgom directory.

B3.4.4. Configuration files for aggregate datasets in the directory /usr/local/las/server/. File names are of the form ngAggregateU1D.xml.

B3.4.5. Aggregate server catalog configuration file catalogConfig.xml in the directory /usr/local/las/las_servlet/jakarta/content/thredds/dodsC/.

B3.4.6. LAS configuration file las.xml in the directory /usr/local/las/server/.

B3.4.7. Data source dataset names for use in creating the hazmat_cat.xml categories file.

B4. createAggregatexml.php

B4.1. Purpose. Creates the configuration files for a logical aggregate dataset. File names are of the form ngAggregateU1D.xml. These files tell the LAS UI what variables the dataset contains and the details of the data's horizontal, vertical, and temporal axes. The routine must accumulate date/times from the individual files that make up the aggregate.

B4.2. Input.

B4.2.1. Name of the aggregates to be created.

B4.2.2. Names of the source data files that are in each aggregate.

B4.2.3. The name of the directory that holds the source data files.

B4.2.4. A list of date/time strings corresponding to the files to be aggregated.

B4.3. Processing.

B4.3.1. For each file to be aggregated, use /usr/local/bin/ncdump to get values of the data's sigma coordinate.

B4.3.2. Format the dataset's root element using the dataset file name.

B4.3.3. Format the dataset's <properties> element.

B4.3.4. Based on the dataset's category, format the appropriate <variable> elements.

B4.3.5. Format the <grid> element.

B4.3.6. Format the <axis> element using input times and sigma/depth values for the time and z axes respectively.

B4.3.7. For each aggregate, write the configuration file to the server directory /usr/local/las/server/.

B4.4. Output.

B4.4.1. Configuration files for each aggregate dataset in the directory /usr/local/las/server/. File names are of the form ngAggregateU1D.xml.

B5. createCatalogConfig.php

B5.1. Purpose. Creates the file catalogConfig.xml. The file contains configuration information that tells the Aggregation Server which files to associate with which aggregate.

B5.2. Input.

B5.2.1. The names of the aggregates that the server needs to create a catalog entry for.

B5.2.2. The names of the files that go into a particular aggregate.

B5.3. Processing.

B5.3.1. Format the catalog elements < catalog>, < dataset >, and top-level < service > elements.

B5.3.2. Format the NGOM <dataset> element and indicate the name of the aggregate. The aggregate name will look something like ngAggregateU1D.nc.

B5.3.3. Format the <metadata> and <aggregation> elements indicating that the aggregated files will be joined across the time axis.

B5.3.4. Format the <fileAccess> elements that hold the file names that make up the aggregate.

B5.3.5. Write the XML data to the file catalogConfig.xml in the directory /usr/local/las/las_servlet/jakarta/content/thredds/dodsC.

B5.4. Output.

B5.4.1. The file catalogConfig.xml is output. It is the Aggregation Server's configuration file and is found in the directory /usr/local/las/las_servlet/jakarta/content/thredds/dodsC.

B6. createCATxml.php

B6.1. Purpose. Creates a configuration file that establishes a hierarchical set of categories for grouping LAS datasets.

B6.2. Input.

B6.2.1. The titles of the category.

B6.2.2. The datasets names that fall under each category.

B6.3. Processing.

B6.3.1. Format the < las_categories> element.

B6.3.2. For each category, format a <category> element to define the category title.

B6.3.3. For each category, format a sequence of <category> and < filter > elements to hold the names of the datasets that are contained in a particular higher level category.

B6.3.4. If this group of datasets has a vertical dimension, include <category> and < filter > elements for a "depth" version of the dataset.

B6.3.5. Write the file hazmat_cat.xml to the directory /usr/local/las/server.

B6.4. Output.

B6.4.1. The LAS configuration file hazmat_cat.xml is output. It contains information that helps organize datasets into groups.

B7. createDGOMxml.php

B7.1. Purpose. Creates the LAS configuration file that defines a particular dataset. It contains a list of the variables found in the dataset as well as a description of the horizontal, vertical, and temporal axes.

B7.2. Input.

B7.2.1. The file name of the dataset for which the configuration information is to apply.

B7.2.2. The directory where the dataset is located.

B7.3. Processing.

B7.3.1. Use /usr/local/bin/ncdump to obtain information about the temporal extent of the data in the dataset and the sigma values used for the z axis.

B7.3.2. Using the time information and the type of dataset (i.e. forecast surface, forecast 3-D, nowcast surface, or nowcast 3-D), create a new, more descriptive, file name. Files names will look something like ng288_05Oct04U3D and will be used when copying the files to the /data/dgom directory.

B7.3.3. Write the new file name to a temporary file called fileNameFile.txt located in the directory /usr/local/las/server/.

B7.3.4. Convert the data valid times to a form that looks like DD-MM-YYYY HH:MM. These values will show up in the UI's *Select time* pull down menu.

B7.3.5. Write the time strings to the file timeListFile.txt located in the directory /usr/local/las/server/.

B7.3.6. Create the file names for the configuration files representing the dataset on depth surfaces. The capital "D" following the "ng288" in the example file name ng288D_05Oct04U3D designates the file as containing depth values for the vertical axis.

B7.3.7. Invoke script createSigmaxml.php to format the dataset's configuration file.

B7.3.8. If this is a dataset with data on the vertical axis (i.e. forecast 3-D or update 3-D), then invoke script createDepthxml.php to format a configuration file with depth values for the z axis.

B7.4. Output.

B7.4.1. A new, more descriptive, file name is written to a temporary file.

B7.4.2. Data valid times are written to a temporary file.

B7.4.3. Configuration files for the dataset (see createSigmaxml.php and createDepthxml.php for more details).

B8. createDepthxml.php

B8.1. Purpose. Creates an LAS dataset configuration file based on data with a vertical axis on depth surfaces. The configuration file defines the variables in the dataset and the horizontal, vertical, and temporal axes.

B8.2. Input.

B8.2.1. The code needs the file name for the depth version of the configuration file.

B8.2.2. The routine requires the file name of the original model output file that resides in the directory /disks/NASWORK/dgom/model-transfer.

B8.3. Processing.

B8.3.1. Format the <dataset> and dataset specific elements (looks something like <ng288D_05Oct04U3D>). The dataset specific element uses the dataset's file name to keep it unique so the XML parser works correctly.

B8.3.2. Format the <properties> element.

B8.3.3. Format the <variables> element which will vary between datasets.

B8.3.4. Format a <composite> element to hold vector value definitions.

B8.3.5. Format the <grids> element.

B8.3.6. Format an <axes> element. User friendly date/time strings, of the form DD-MM-YYYY HH:MM, are used for the temporal axis. Standard depths (Table 5) are used for the z axis.

Table 5. Standard Depths in Meters

Depth (m)	Depth (m)	Depth (m)
0	300	1400
10	400	1500
20	500	1750
30	600	2000
50	700	2500
75	800	3000
100	900	3500
125	1000	4000
150	1100	4500
200	1200	5000
250	1300	5500

B8.3.7. Write the configuration file to the directory /usr/local/las/server/. The file name will look something like ng288D_05Oct04U3D.xml.

B8.4. Output.

B8.4.1. The code outputs a depth-based configuration file for the dataset.

B9. createSigmaxml.php

B9.1. Purpose. This routine creates an LAS dataset configuration file based on data with a vertical axis on sigma surfaces. The configuration file defines the variables in the dataset and the horizontal, vertical, and temporal axes.

B9.2. Input.

B9.2.1. The code needs the file name for the sigma version of the configuration file. This name looks something like ng288_05Oct04U3D.xml.

B9.2.2. The routine requires the file name of the original model output file that resides in the directory /disks/NASWORK/dgom/model-transfer.

B9.3. Processing.

B9.3.1. Format the <dataset> and dataset specific (looks something like <ng288_05Oct04U3D>) elements. The dataset specific element uses the dataset's file name to keep it unique so the XML

parser works correctly.

B9.3.2. Format the <properties> element.

B9.3.3. Format the <variables> element which will vary between datasets.

B9.3.4. Format a <composite> element to hold the vector value definitions.

B9.3.5. Format the <grids> element.

B9.3.6. Format an <axes> element. User friendly date/time strings, of the form DD-MM-YYYY HH:MM, are used for the temporal axis. Model sigma values are used for the z axis.

B9.3.7. Write the configuration file to the directory /usr/local/las/server/. The file name will look something like ng288_05Oct04U3D.xml.

B9.4. Output.

B9.4.1. The code outputs a sigma-level-based configuration file for the dataset.

B10. createLASxml.php

B10.1. Purpose. This program creates the las.xml file used as the top-level source of configuration information. It points the LAS to where individual datasets configuration files are located. The file also contains the location of any category configuration files. The program createLASxml.php also tells the LAS where to find information on the operations it can perform.

B10.2. Input.

B10.2.1. The script needs the names of all the datasets that the LAS can serve.

B10.3. Processing.

B10.3.1. Format the <!ENTITY> elements for any categories, operations, and datasets the server will need configuration information about. This element contains the file name of each configuration file.

B10.3.2. Format the <lasdata> element.

B10.3.3. Format a reference to the category entity.

B10.3.4. Format the <institution> element to indicate who is operating the LAS.

B10.3.5. Format the <properties> element.

B10.3.6. Format the <operations> element. This element holds references to dataset specific operation entities like &HAZMAT_operations and &HAZMATstd_operations.

B10.3.7. Format references to all the dataset name entities.

B10.3.8. Write the configuration file las.xml to the directory /usr/local/las/server.

B10.4. Output.

B10.4.1. Output the configuration file las.xml.

B11. makeCurve2RecMap.jnl

B11.1. Purpose. This Ferret script, located in the directory /usr/local/ferret/go, is used to create the curvilinear to rectilinear mapping[8] necessary for the LAS to properly interpret NGOM data. This script needs to only run once for each data source or when a data source's grid structure changes.

B11.2. Input.

B11.2.1. The script needs the path and name of the file that contains the curvilinear latitude and longitude values. This information is coded directly into the script.

B11.2.2. The script user must also determine the range of latitude and longitude values in the data grid. This information is entered in the following two lines of code within the script file:

```
def axis/x=-98.162:-78.042:.050/modulo/units=degrees xax
def axis/y=17.991:30.734:.05/units=degrees yax
```

B11.2.3. The name of the file to contain the curvilinear to rectilinear map. This file is used by custom Ferret scripts and needs to be stored in the directory /usr/local/las.

B11.3. Processing.

B11.3.1. Access the input dataset for which a curvilinear to rectilinear map is desired.

B11.3.2. Capture the latitude and longitude information from the input data file.

B11.3.3. Define axes for the map using the data's grid to determine the minimum and maximum latitude and longitude values.

B11.3.4. Compute weights for the mapping.

B11.3.5. Save the new map for use by LAS scripts.

[8] See http://ferret.pmel.noaa.gov/LASdoc/serve/cache/82.html for information on mapping curvilinear coordinates.

B11.4. Output.

B11.4.1. The file containing the curvilinear to rectilinear map (i.e. NOS_GOM_Model_C2Rmap.nc).

APPENDIX C: LAS (VERSION 6.5.1.2) PREREQUISITE AND SUPPORT SOFTWARE

C1. Prerequisite Software.

C1.1. Ferret (Version: 5.8.1)

Source: http://ferret.pmel.noaa.gov/Ferret/

C1.2. Perl (Version: v5.8.0 built for i386-linux-thread-multi)

Source: http://www.perl.org

C1.2.1. Module Bit-Vector (Version: 6.4)

Source: http://search.cpan.org/~rse/lcwa-1.0.0/lib/bitvector/Vector.pm

C1.2.2. Module Carp-Clan (Version: 5.3)

Source: http://search.cpan.org/~stbey/Carp-Clan-5.3/Clan.pod

C1.2.3. Module Date-Calc (Version: 5.4)

Source: http://search.cpan.org/~stbey/Date-Calc-5.4/

C1.2.4. Module DBD-mysql (Version: 3.0002)

Source: http://search.cpan.org/~capttofu/DBD-mysql-3.0002/lib/DBD/mysql.pm

C1.2.5. Module File-PathConvert (Version: 0.9)

Source: http://search.cpan.org/~rbs/File-PathConvert-0.9/PathConvert.pm

C1.2.6. Module Log-Agent (Version: 0.307)

Source: http://search.cpan.org/~mrogaski/Log-Agent-0.307/Agent.pm

C1.2.7. Module Netcdf-perl (Version: 1.2.3)

Source: http://home.badc.rl.ac.uk/iwi/netcdf-perl/

C1.3. MySQL (Version: 4.0.25)

Source: http://www.mysql.com/

C1.4. Expat (Version: 1.95.8)

Source: http://expat.sourceforge.net/

C1.5. Java (Version: 1.4.2.09)

Source: http://java.sun.com/

C1.6. NetCDF Libraries (Version: 3.6.0-pl)

Source: http://www.unidata.ucar.edu/software/netcdf/

C1.7. Ant (Version: 1.6.5)

Source: http://ant.apache.org/

C2. Support Software

C2.1. Apache Web Server (Version: 2.0.46)

Source: http://www.apache.org/

C2.2. Apache Tomcat (Version: 5.5.9)

Source: http://tomcat.apache.org/

C2.3. RedHat Linux (Version: Red Hat Enterprise Linux AS release 3 [Taroon Update 6])

Source: http://www.redhat.com/

APPENDIX D: RUNNING THE LAS (VERSION 6.5.1.2) CONFIGURE SCRIPT

D1. Prerequisite Steps

D1.1. Prior to running the configuration script, the LAS software must be downloaded from the LAS website[9] and placed on the server according to the installation instructions[10].

D1.2. Additionally, all prerequisite software must be installed (see Appendix A1). It is worth noting here that PMEL tests LAS software using specific versions of the prerequisite software. It is strongly advised that local implementations of the LAS use the same versions. Version information can be found in the installation instructions discussed in Section D1.1.

D2. Development LAS Configuration

D2.1. The development LAS is configured from its home directory (/usr/local/las_dev). The configure script queries/responses are provided in Section D2.2. In most cases, the appropriate response is to accept the default value shown between the brackets. This is indicated by a lone "CR" (carriage return) at the end of the script query line. For queries where the default is not appropriate, the recommended text is provided followed by a "CR".

D2.2. Running the Development LAS Configuration Script.

cd /usr/local/las_dev
source ../ferret_paths
sudo ./configure
Location of perl executable [/usr/bin/perl]: *CR*
Location of ferret executable: [/usr/local/ferret/bin/ferret] *CR*
Location of java executable: [/usr/local/j2sdk1.4.2_09/bin/java] *CR*
Would you like to use a different Tomcat installation for LAS? [yes] *CR*
… This LAS uses same Tomcat as production LAS: [/usr/local/las/las_servlet/jakarta] *CR*
Which HTTP port does the Tomcat server use: [] ***8080*** *CR*
Enter the domain name of the Tomcat Server (do not include the port number): [205.156.4.56] *CR*
Enter name of mysql host : [localhost] *CR*
MySQL account name: [root] *CR*
Enter password: [] *CR*
Enter the fully qualified domain name of the mysql host : [DODS.ncd.noaa.gov] ***205.156.4.56*** *CR*
… domain name of the Web Server (including port number if non standard): [205.156.4.56] *CR*
… from the HTTP server to the tomcat server (recommended; instructions below): [yes] *CR*
Is the server running? [yes] *CR*
Enter path name for LAS: [/las_dev] *CR*
Enter path name for the LAS server: [/las_dev-bin] *CR*

[9] LAS home page: http://ferret.pmel.noaa.gov/Ferret/LAS/
[10] LAS Installation Instructions: http://ferret.pmel.noaa.gov/LASdoc/serve/cache/11.html

Enter path name for the LAS server output: [/las_dev-output] *CR*
Enter a title for the LAS server: [CSDL Development LAS] *CR*
… separated list of email address(es): [John.Schattel@noaa.gov Jacobo.Freireich@noaa.gov] *CR*
Do you want to set up the server to use this file? [yes] *CR*
Group name? [apache] *CR*

D3. Public LAS Configuration

D3.1. The public LAS is configured from its home directory (/usr/local/las). The configure script queries/responses are provided in Section D3.2. In most cases, the appropriate response is to accept the default value shown between the brackets. This is indicated by a lone "CR" (carriage return) at the end of the script query line. For queries where the default is not appropriate, the required text is provided followed by a "CR".

D3.2. Running the Public LAS Configuration Script

cd /usr/local/las_dev
source ../ferret_paths
sudo ./configure
Location of perl executable [/usr/bin/perl]: *CR*
Location of ferret executable: [/usr/local/ferret/bin/ferret] *CR*
Location of java executable: [/usr/local/j2sdk1.4.2_09/bin/java] *CR*
Would you like to use a different Tomcat installation for LAS? [no] *CR*
Use the following Tomcat ports [8080, 8005, 8008]? [yes] *CR*
… full domain name of the Tomcat Server (do not include the port number): [205.156.4.56] *CR*
Enter name of mysql host : [localhost] *CR*
MySQL account name: [root] *CR*
Enter password: [] *CR*
Enter the fully qualified domain name of the mysql host : [DODS.ncd.noaa.gov] ***205.156.4.56*** *CR*
… domain name of the Web Server (including port number if non standard): [205.156.4.56] *CR*
Do you plan to use a proxy pass or connector from the HTTP server to the tomcat server (recommended; instructions below): [yes] *CR*
Is the server running? [yes] *CR*
Enter path name for LAS: [/las] *CR*
Enter path name for the LAS server: [/las-bin] *CR*
Enter path name for the LAS server output: [/las-output] *CR*
Enter a title for the LAS server: [Coast Survey Development Laboratory LAS] *CR*
… separated list of email address(es): [John.Schattel@noaa.gov Jacobo.Freireich@noaa.gov] *CR*
Do you want to set up the server to use this file? [yes] *CR*
Group name? [apache] *CR*