

Open Vessel Data Management User's Guide

Version 2.3

Last Updated: October 25, 2019

Table of Contents

1 Overview.....	1
1.1 What is OpenVDMv2?.....	1
1.2 What does OpenVDMv2 do?.....	1
1.3 OpenVDMv2 Core Philosophies.....	2
1.3.1 Remain flexible to Vessel Operators.....	2
1.3.2 Minimal Conformity Requirements.....	2
1.3.3 No changes to data files.....	2
1.3.4 Delete nothing.....	2
2 Quick Start.....	3
2.1 Creating a New Cruise.....	3
2.2 Running the End-of-Cruise Tasks.....	4
2.3 Starting OpenVDM.....	4
2.4 Stopping OpenVDM.....	4
3 OpenVDMv2 System Components.....	6
3.1 The Shipboard Data Warehouse.....	6
3.1.1 The Cruise Data Directory.....	6
3.1.2 The Public Data Directory.....	6
3.1.3 The Visitor Information Directory.....	6
3.1.4 Samba and HTTP Access to Data Directories.....	6
3.1.5 The OpenVDMv2 Web-Application.....	6
3.1.6 The Job Broker: Gearman/Supervisor.....	6
3.2 The Shore-side Data Warehouse.....	7
3.2.1 OpenVDMv2 – Port Office.....	7
4 The OpenVDMv2 Web-Application.....	7
4.1 Basic Concepts.....	7
4.1.1 The CruiseID and Start/End Date & Time.....	7
4.1.2 The LoweringID and Start/End Date & Time.....	7
4.1.3 Collection System Transfers.....	8
4.1.4 Extra Directories.....	8
4.1.5 Cruise Data Transfers.....	8
4.1.6 Ship-to-Shore Transfers.....	9
4.2 The Data Dashboard and DashboardData files.....	9
4.3 The Web-Application Layout.....	10
4.3.1 Common Elements.....	10
4.3.2 Home.....	12
4.3.3 Data Dashboard.....	14
4.3.4 Configuration.....	18
5 Data Dashboard Visualization Plugin Development Guide.....	24
5.1 Overview.....	24

OpenVDMv2 User's Guide

5.2 The Processing Workflow.....	24
5.3 The Dashboard Data JSON Schema.....	26
5.3.1 visualizerData.....	26
5.3.2 stats.....	27
5.3.3 qualityTests:.....	27
5.4 A working example of a visualization plugin.....	28
6 Customizing the Data Dashboard.....	31
6.1 datadashboard.yaml.....	31

Table of Screen Captures

Illustration 1: Cruise Start/End Panel.....3

Illustration 2: OpenVDMv2 Header.....10

Illustration 3: OpenVDMv2 Navigation Elements.....10

Illustration 4: OpenVDMv2 Messages Screen.....11

Illustration 5: OpenVDMv2 Home Section.....13

Illustration 6: OpenVDMv2 Data Dashboard Section.....14

Illustration 7: Sample of GIS-based data displayed within the Data Dashboard.....15

Illustration 8: Sample of time series-based data displayed within the Data Dashboard.....16

Illustration 9: Data Quality tab within the Data Dashboard.....17

1 Overview

This document is intended to provide an overview of Open Vessel Data Management version 2.3 (OpenVDMv2) and its use under normal circumstances and installation profile. While this guide is intended as a universal reference for all OpenVDMv2 installations it may not account for every all customizations.

The document is organized as follows: Section 1 is an overview of OpenVDMv2. Section 2 is a quick start guide for the most common tasks. Section 3 describes the many components of OpenVDMv2. Section 4 describes the web-application for OpenVDMv2, Section 5 describes the optional shore-side part of OpenVDMv2, OpenVDMv2 – Port Office.

1.1 What is OpenVDMv2?

The most important thing to understand about OpenVDMv2 is the exact set of tasks it is built to accomplish. OpenVDMv2 is NOT a data acquisition system (DAS). OpenVDMv2 does NOT write data from a sensor to a file. OpenVDM is a data management tool and thus it manages files created by data acquisition systems. OpenVDMv2 is NOT designed to fix data problems within data files or correct misnamed data files, instead it is a tool to help vessel operators find these types of errors and address these types of problems at the source.

The goal of OpenVDM is to take data files from multiple data acquisition systems running throughout the vessel and create a single well-organized cruise data package that is consistent from cruise-to-cruise. How files/directories are named and how data files are organized are entirely determined by the vessel operator. OpenVDM is simply a tool for enforcing a vessel's data management plan and alerting the vessel operator when there are problems.

1.2 What does OpenVDMv2 do?

In simple terms, OpenVDMv2 strives to help vessel operators organize datasets created aboard their vessels and make those datasets safely available to shipboard personnel immediately.

To accomplish this goal, OpenVDM copies data from the various DAS workstations, where data files are created, to a single cruise data directory on a central shipboard server. As soon as new files arrive on the central shipboard server they are made available as read-only files to science and crew via multiple protocols such as SMB and http.

OpenVDMv2 also strives to make configuration, control and operation as quick and painless as possible, allowing vessel operators and technicians to focus on collecting more/better data.

1.3 OpenVDMv2 Core Philosophies

The design and continuing evolution of OpenVDMv2 is guided by several core philosophies. These core philosophies keep development efforts focused and the project moving forward in a well-defined direction.

1.3.1 Remain flexible to Vessel Operators

OpenVDMv2 is designed to be as flexible as possible so that it can be leveraged by the largest cross-section of oceanographic vessel operating modes. The user-interface and underlying code strives to provide the vessel operator with the greatest number of options for how to define source locations, filename filters, destination locations and transport protocols. Every vessel operator has their preferred way to move data. It is the goal of the OpenVDMv2 development team to provide vessel operators the option to use the methods they are most comfortable with.

1.3.2 Minimal Conformity Requirements

OpenVDMv2 strives to help vessel operators organize their collected datasets to the operator's standards, not to one defined by OpenVDMv2.

OpenVDMv2 does not mandate any particular standardization for the source/destination locations of data files or transport protocol used to move the files. OpenVDM strives to let the vessel operator make those decisions based on what works best for their vessel. The only true requirement OpenVDMv2 places on the vessel operator is that the vessel operator must have a clearly defined plan.

1.3.3 No changes to data files

Under no circumstances will OpenVDMv2 ever alter the files it manages. This includes any alterations to either the name or contents of a file. If there is anything about the file that does not conform with the vessel's defined data management plan, OpenVDM will alert the vessel operator of the problem but resolving the problem must be handled by the operator or an external mechanism.

1.3.4 Delete nothing

OpenVDMv2 does not delete files. If a file is being managed by OpenVDMv2 by mistake, the vessel operator or designate must manually remove the file from the cruise data directory.

2 Quick Start

For those who hate reading an entire manual or just simply need a quick refresher on how things work here an abbreviated guide for performing the most common tasks.

2.1 Creating a New Cruise

1. Define the new cruise ID and start date.
 - a) Goto: Configuration → Main Tab. This may require a logging into the system if not done previously.
 - b) Once at the Main Tab click the “Setup New Cruise” button located within the “Cruise Control” panel. This will open the Cruise Creation Form.
 - c) Complete the Cruise Creation Form by specifying the cruise name, start date/time and enabling the desired Collection System Transfers and optional components.

*** Double-clicking the date fields will open a date picker widget.
 - d) When finished, click “Create”.

*** If you have entered a cruise name that already exists you will be asked to pick another.
 - e) Completing the form will initiate the cruise data directory creation process. Verify all the parts of the cruise creation task were successful (all green check marks in the modal window)
2. Verify the Collection Systems are available for transfers. To do this goto Configuration → Collection System Transfers tab. Click the “Test” link next to the systems that have been enabled for this cruise. Resolve any error's encountered during these tests.
3. Turn on the automated transfers. Click the red “System Status” Panel to switch the system status from “Off” (in red) to “On” (in green).
4. Move on to something else... OpenVDMv2 will handle things from here.

2.2 Running the End-of-Cruise Tasks

1. Goto: Configuration → Main Tab. This may require a logging into the system if not done previously. Click the “System Status” Panel to set the system status to “Off” (in red).
2. Configuration → Main Tab. Click “Run End-of-Cruise Tasks” button located within the “Cruise Control” panel.
3. Wait for the “Finalizing Cruise” task and any subsequent tasks to complete in the task status dropdown. Once the task has completed for the first time, a “check” icon will appear to the left of the text informing the user that the task has finished successfully. If there was a problem with the task a “warning” icon will appear to the left of the text. Please note that this task complete notification does not take into account any linked tasks such as updating the data dashboard or any custom end-of-cruise tasks the vessel operator may have configured.
4. Optionally you may need to manually run any enabled Cruise Data Transfers to copy any new data pulled to the Data Warehouse as part of the Cruise Finalization to the external storage device/location.
5. Optionally you may need to manually run the Ship-to-Shore Transfers to push any new data pulled to the Data Warehouse as part of the Cruise Finalization to the shore-based data warehouse.

2.3 Starting OpenVDM

Configuration → Main Tab. Click the “System Status” Panel to set the system status to “On” (in green). This requires that the user is logged into OpenVDMv2. If the user is not logged into OpenVDM, clicking the System Status Panel will do nothing.

2.4 Stopping OpenVDM

Configuration → Main Tab. Click the “System Status” Panel to set the system status to “Off” (in red). This requires that the user is logged into OpenVDMv2. If the user is not logged into OpenVDM, clicking the System Status Panel will do nothing.

3 OpenVDMv2 System Components

3.1 The Shipboard Data Warehouse

The Shipboard Data Warehouse is the core physical component of OpenVDMv2. This is the server that hosts the OpenVDMv2 web-application, background processes and stores cruise data.

3.1.1 The Cruise Data Directory

This is the directory on the Shipboard Data Warehouse where all the cruise data directories reside.

3.1.2 The Public Data Directory

This is the directory on the Shipboard Data Warehouse where science personnel and crew can share files.

3.1.3 The Visitor Information Directory

This is the directory on the Shipboard Data Warehouse where technicians can post files for crew and science personnel to access.

3.1.4 Samba and HTTP Access to Data Directories

A typical OpenVDMv2 installation provides multiple protocols for accessing the Cruise Data Directory, Public Data Directory and Visitor Information Directory. This helps ensure anyone that requires access to data and information can get to it using the method they are most comfortable with.

3.1.5 The OpenVDMv2 Web-Application

The OpenVDMv2 Web-Application provides the user-interface for accessing information about the current cruise and controlling the various OpenVDMv2 behaviors.

3.1.6 The Job Broker: Gearman/Supervisor

Behind the OpenVDMv2 Web-Application is a large array of scripts and programs that perform all the various background tasks and data transfer functions of OpenVDMv2. Managing all of these background processes is the responsibility of the Job Broker. The

Job Broker is comprised of two powerful open-source programs, Gearman and Supervisor. Gearman connects job requests to the appropriate script/program to do the work. Supervisor handles starting/stopping the various programs and scripts.

3.2 The Shore-side Data Warehouse

The Shore-side Data Warehouse is the optional shore-based counterpart for the Shipboard Data Warehouse. It's where data is sent during ship-to-shore transfers.

3.2.1 OpenVDMv2 – Port Office

Port Office is an optional web-application that can be installed on the shore-side data warehouse for displaying dashboard data created on the ship for both current and past cruises. Documentation for Port Office is still in development and will not be discussed much within this document.

4 The OpenVDMv2 Web-Application

What separates OpenVDMv2 from data management methods used on other vessels is OpenVDMv2's web-application. The web-application strives to provide an intuitive interface for both configuring and managing all of OpenVDMv2's various components and behaviors.

4.1 Basic Concepts

When describing how to use the OpenVDMv2 web-application it is important to understand some of the web-application's core concepts and nomenclature.

4.1.1 The CruiseID and Start/End Date & Time

At the most basic level, OpenVDMv2 organizes collected data by cruise. The cruiseID is the official designation for a cruise. This is usually very short, and often includes an incremented value (i.e. CS1701, CS1702, etc). This is different than a cruise or project title. When creating a new cruise within OpenVDMv2 the cruise is given a start date/time. This is the date/time the cruise begins and does not necessarily need to be the current date/time. Optionally an end date/time can be defined. This is the date/time the cruise ends.

4.1.2 The LoweringID and Start/End Date & Time

Starting with version 2.3, OpenVDMv2 can manage data from ROVs where data needs to be ordered by lowering within a cruise. The loweringID is the official designation for a lowering. This is usually very short, and often includes an incremented value (i.e. CS-001, CS-002, etc). This is different than the cruiseID. When creating a new lowering within OpenVDMv2 the lowering is given a start date/time. This is the date/time the lowering begins and does not necessarily need to be the current date/time. Optionally an end date/time can be defined. This is the date/time the lowering ends. The loweringID must be unique with a single cruise however the same loweringID can be reused on multiple cruises (i.e., the first lowering of every cruise is “deck_test”).

4.1.3 Lowering Components

Starting in version 2.3, OpenVDM can optionally manage data from a single deployable platform such as an ROV. Not all vessels will require this functionality and therefore the operator can choose to hide all lowering-related components from the OpenVDM user-interface. This is done from the Cruise Control panel on the main tab of the Configuration section of the web-application. To show/remove all lowering components from the user-interface, click “Edit Current Cruise” and enable/disable “Show Lowering Components”. Enabling/disabling lowering components can also be done when setting up a new cruise.

The easiest way to know whether lowering components have been enable is by looking at the web-application header. When lowering components are disabled there will be 4 panels shown. When the components are enabled there will be 6. The 2 additional panels contain the loweringID and directory size of the current lowering.

4.1.4 Collection System Transfers

Within OpenVDMv2 a Collection System refers to any data acquisition system whose data is managed by OpenVDMv2. A Collection System Transfers is the unit of information used by OpenVDMv2 to locate the appropriate files on a Collection System workstation and move those files to the correct destination directory on the shipboard data warehouse. This unit of information includes: where the collection system is on the network, how to connect to the collection system, where the data files exist on the collection system, any filename/directory filters and the destination directory for those data files within the cruise data directory. Starting with OpenVDM version 2.3 Collections System Transfers are specified as being cruise or lowering. This defines whether

OpenVDM should manage the data as a cruise-level dataset (all files collected during the cruise are stored in one location) or as a lowering-level dataset (data is organized by lowering within the cruise data directory).

4.1.5 Extra Directories

Extra Directories are directories managed by OpenVDMv2 within the cruise data directory that are not associated with a collection system transfer. These are typically created at the request of the vessel operator for vessel-specific reasons.

There are a few extra directories that are required by OpenVDM for internal purposes. Although these OpenVDM-specific extra directories are required, their names/locations can be edited by the vessel operator.

4.1.6 Cruise Data Transfers

Cruise Data Transfers are units of information used by OpenVDM to setup an automated copy/backup of the entire cruise data directory, or just a subset of the collected cruise data to a remote server, NAS or external hard drive. These can be used for backing up cruise data to the vessel's backup server and/or for creating backups for the scientist party and archival facilities.

4.1.7 Ship-to-Shore Transfers

Ship-to-shore transfers are units of information used to identify subsets of the cruise data directory for transfer to the shore-side data warehouse. Ship-to-shore transfers include what types of files to transfer, where to look for those files and a transfer priority for defining the order in which to transfer the files to the shoreside data warehouse.

4.2 The Data Dashboard and DashboardData files

The Data Dashboard section of the web-application is for vessel operators to create custom data visualizations of individual data files, report out on quality assurance test results for individual files and report out on statistical information collected from individual data files.

The main data dashboard tab and data quality tab are part of the default OpenVDMv2 installation. Additional tabs can be added by the vessel operator.

There are built-in functions for displaying time-series and geographic data sets. There are

also mechanisms for allowing the vessel operator to build custom visualizations.

The Data Dashboard does not attempt to visualize the raw data directly. Instead it uses JSON-formatted DashboardData files which contain an averaged representation of the raw data, QA test names/results and data statistics. The extent of data files displayed within the Data Dashboard section is dependent on the vessel operator defining what files to visualize, how to visualize those files, what quality tests to run against those files and what statistics to collect from those files. After this information is known the vessel operator must develop OpenVDMv2 plugins that create DashboardData files from the raw files.

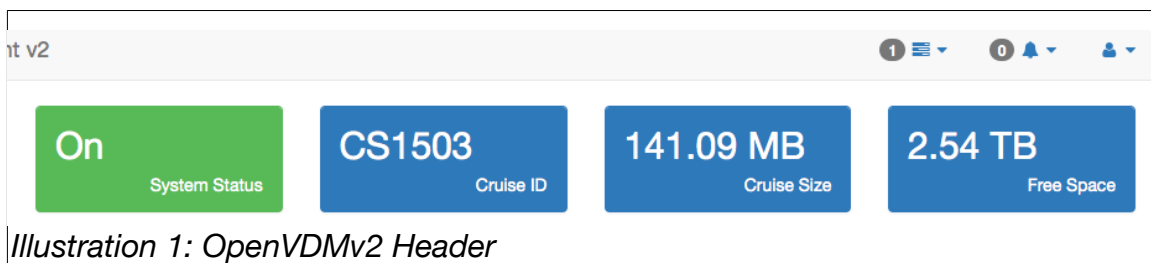
Developing OpenVDMv2 plugins is a topic unto itself and will not be discussed much further within this document.

4.3 The Web-Application Layout

4.3.1 Common Elements

Header

Every page in the web-application contains a common array of elements within the page header. The header includes a task process dropdown, messages dropdown, user management/logout dropdown, the system status, cruiseID, the size of the cruise data directory and the remaining free space on the shipboard data warehouse.



Navigation

There are two main navigation elements to OpenVDM. The primary navigation element is the vertical bar on the left of the interface. This navigation element is present on all pages. The secondary navigation element is a horizontal row of tabs below the header elements. The content of the secondary navigation changes based on the current selected section of the web-application.

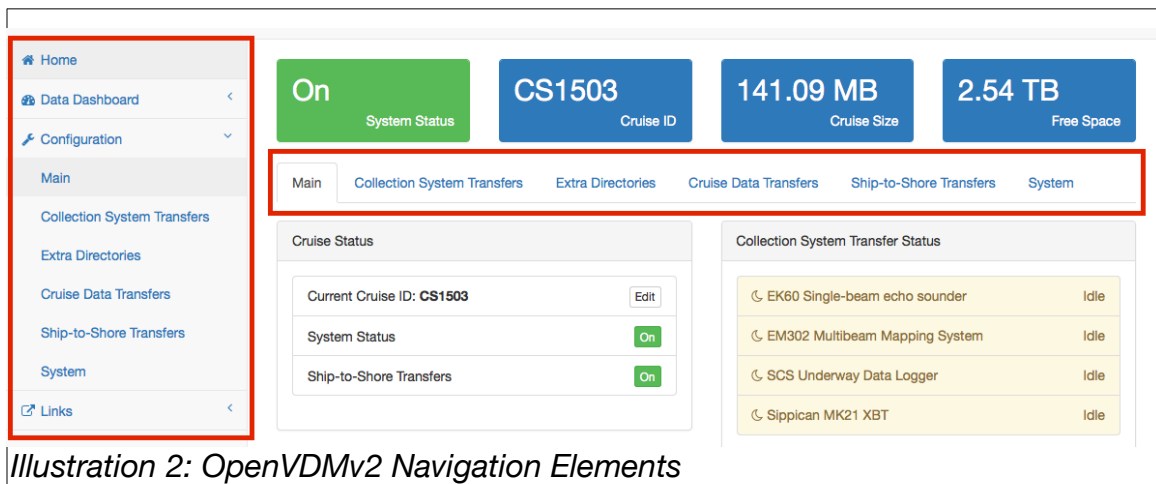


Illustration 2: OpenVDMv2 Navigation Elements

Authentication

The Home and Data Dashboard sections of the web-application are open and do not require authentication.

The Configuration section does require user authentication. Whenever the users goes to any part of the configuration section without prior authentication, they will be presented with the login screen.

The contents of the links section changes depending on whether the user has authenticated with OpenVDMv2. The vessel operator can define additional links and specify whether those links should be public or require authentication.

User Management

The username and password for the management account may be changed at any time.

Messages

Whenever OpenVDM needs to inform the user that something unexpected has occurred, it creates a message. Messages can be quickly acknowledges by clicking on them in the Message dropdown or by going to the messages section and acknowledging/deleting the message(s) there. The messages section does require authentication.

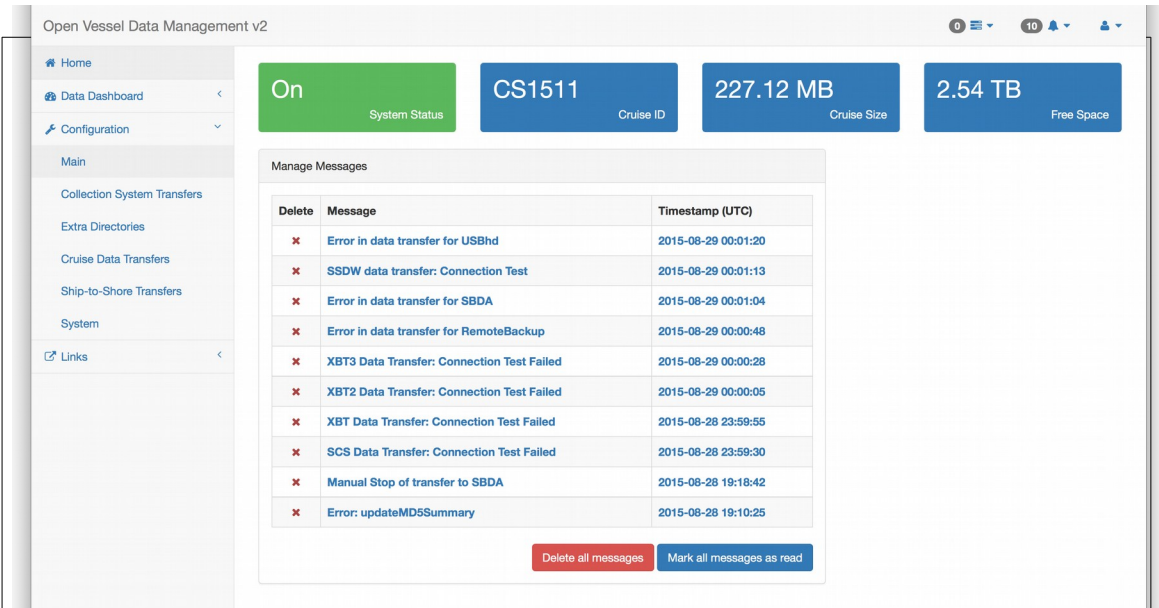


Illustration 3: OpenVDMv2 Messages Screen

4.3.2 Home

This is the top-most section in the sidebar navigation element. This is also where users will be first directed to when they access the OpenVDM web-application.

Incorrect Filenames Detected

This panel in the upper left displays the files OpenVDM believes do not match the file naming conventions of the vessel. The contents of the panel lists the files by Collection System

Recent Shipboard Data Transfers

This panel in the middle left lists the date/time, collection system and filenames of the last 5 shipboard transfers from Collection Systems to the shipboard data warehouse.

Recent Ship-to-Shore Data Transfers

This panel in the lower left lists the date/time and filenames of the last 5 ship-to-shore transfers from the shipboard data warehouse to the shoreside data warehouse.

Collection System Transfer Status

This panel shows the current status of the currently enabled Collection System Transfers. If a Collection System Transfer is disabled it will not appear.

Cruise Data Transfer Status

This panel shows the current status of the currently enabled Cruise Data Transfers. If a Cruise Data Transfer is disabled it will not appear. The only exception is the Ship-to-Shore Transfer which will appear even if disabled.

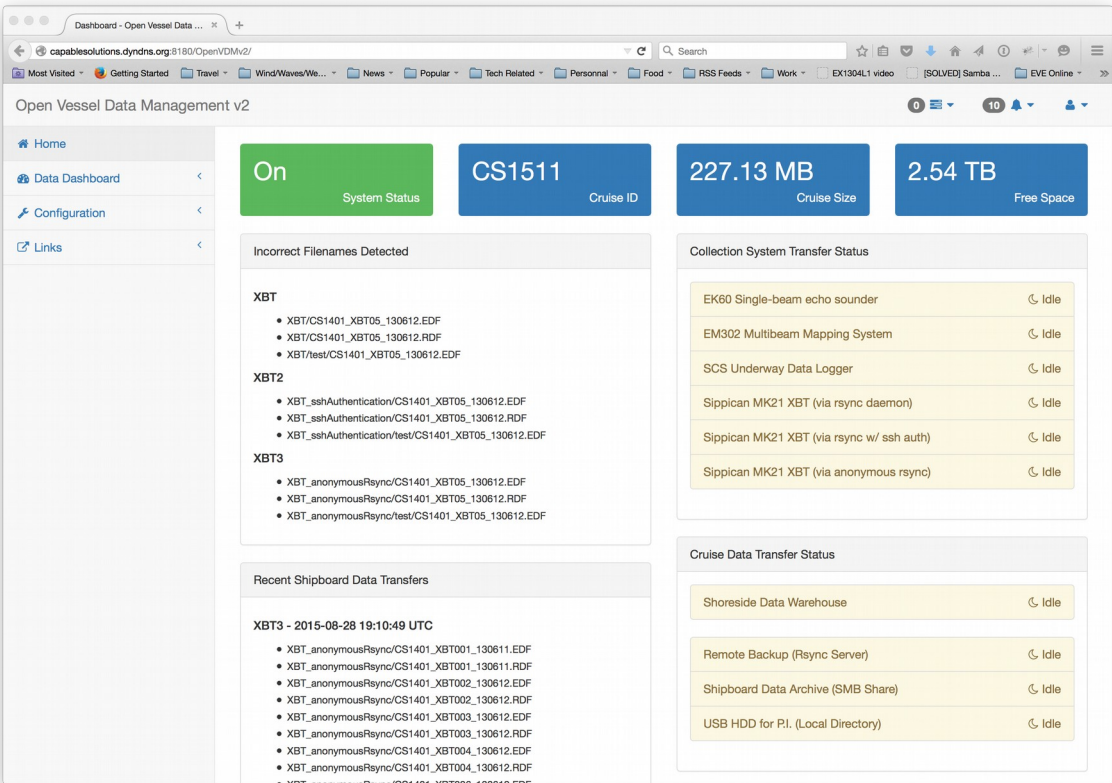


Illustration 4: OpenVDMv2 Home Section

4.3.3 Data Dashboard

The second section in the sidebar navigation is the Data Dashboard. This section is separated into several sub-sections which are displayed in the secondary navigational element.

Main

The main tab shows the most recent visualizer data for each of the dashboardData datatypes. Clicking on a widget redirects the user to the appropriate tab that displays the data in a larger panel.

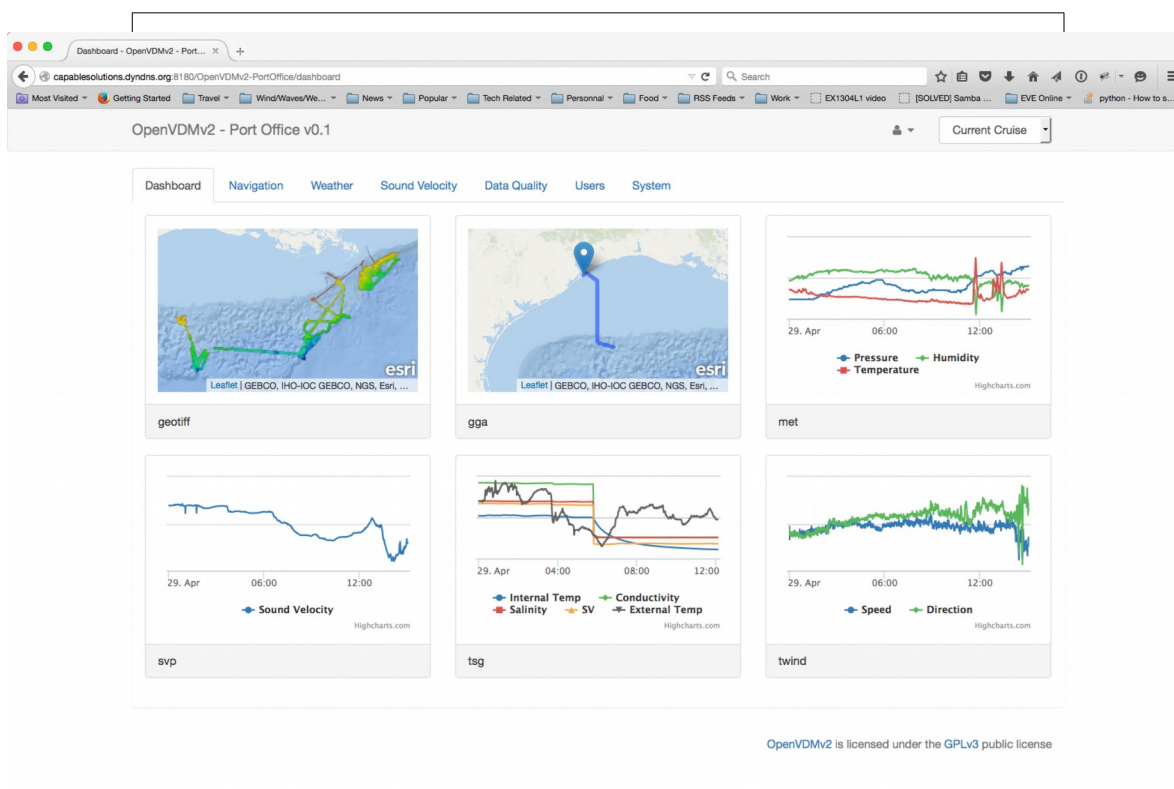


Illustration 5: OpenVDMv2 Data Dashboard Section

GIS Display

OpenVDM includes functions for visualizing geoJSON-formatted data and TMS tiles onto a Leaflet-based map. Checkboxes below the map can be selected to display additional data/tile sets.

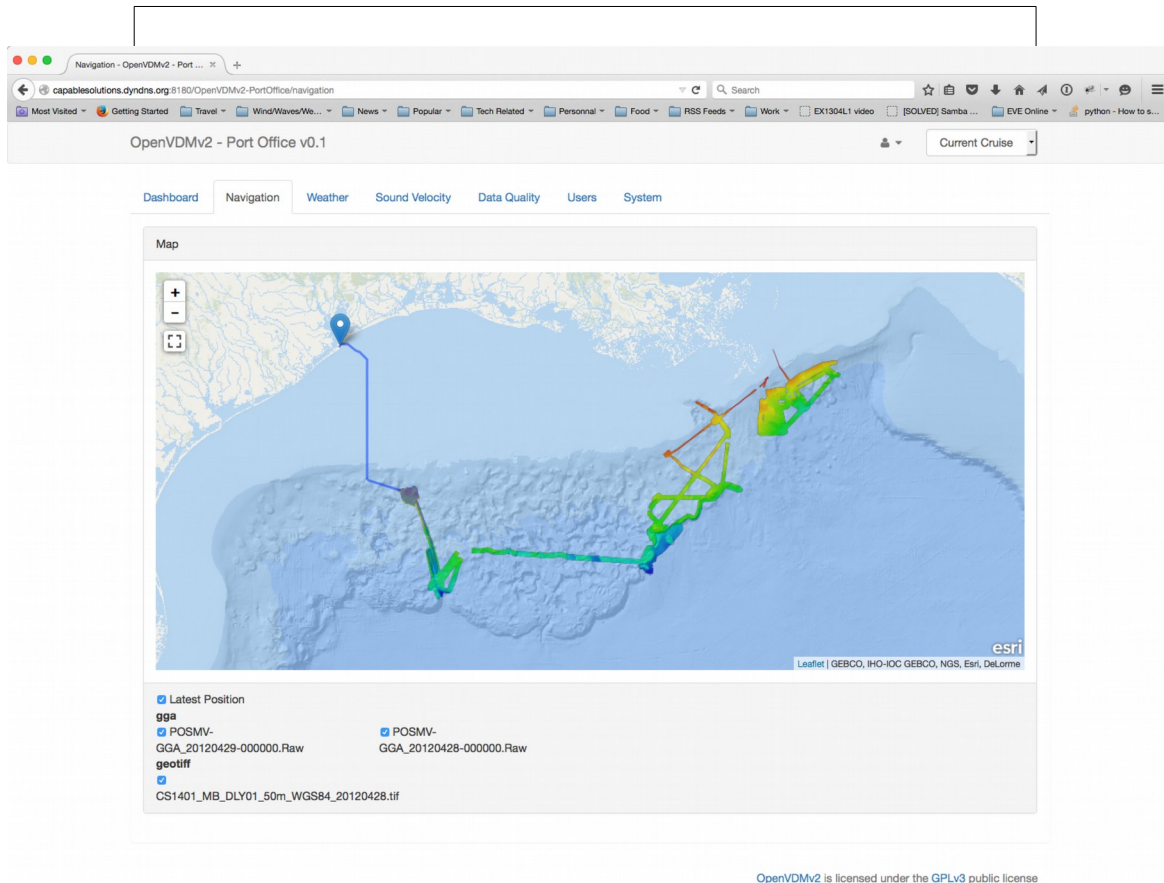


Illustration 6: Sample of GIS-based data displayed within the Data Dashboard

Time-series Display

OpenVDM can visualize time-series data within a chart. Radio buttons beneath the chart allow the user to select a different dataset to visualize. Only one file's worth of data can be displayed at a time using the default OpenVDM charting functions.

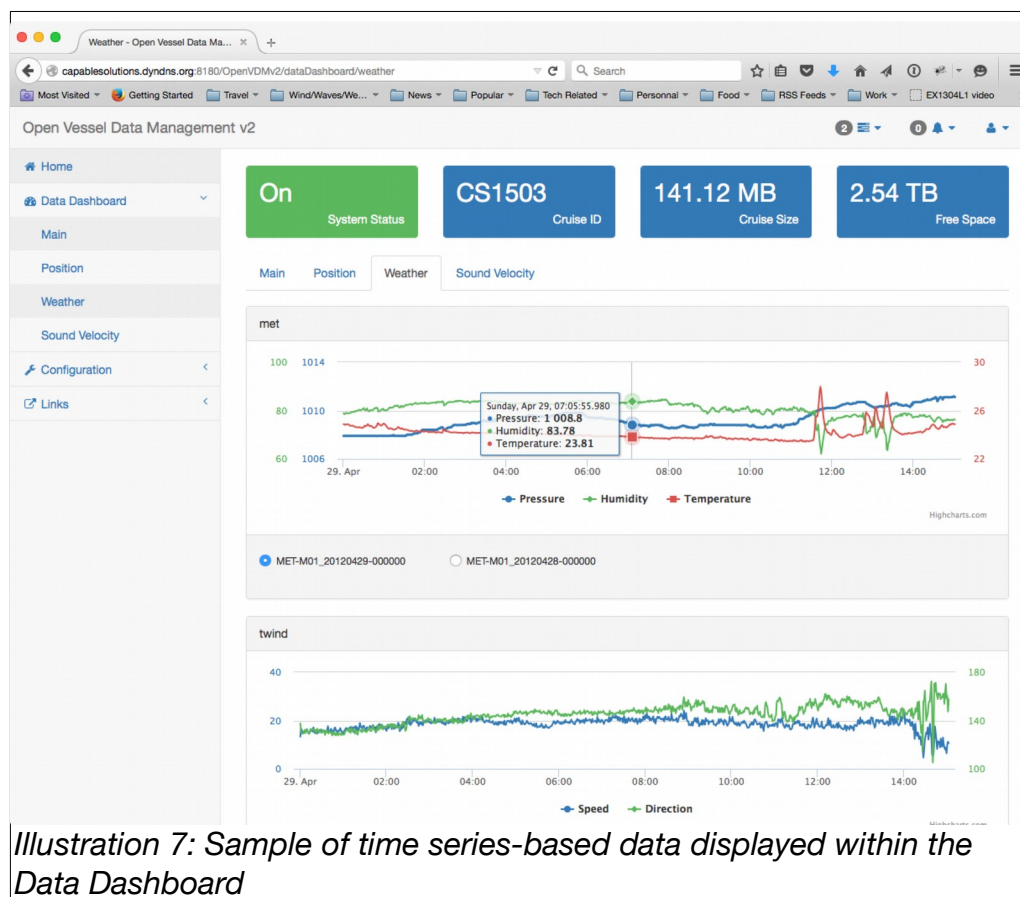


Illustration 7: Sample of time series-based data displayed within the Data Dashboard

QA/QC

The QA/QC section shows the quality test results and links to view the data statistics for each file. QA Tests and data statistics are optional components of dashboardData files so it is possible that a file may not show any QA test results and/or an active link to view the data statistics.

The screenshot displays the 'QA/QC' tab within the 'Open Vessel Data Management v2' dashboard. The interface includes a sidebar with navigation options: Home, Data Dashboard, Main, Position, Weather, Sound Velocity, QA/QC (selected), Configuration, and Links. The main content area shows four status cards at the top: 'On' (System Status), 'CS1502' (Cruise ID), '233.05 MB' (Cruise Size), and '2.54 TB' (Free Space). Below these are tabs for 'Main', 'Position', 'Weather', 'Sound Velocity', and 'QA/QC'. The 'QA/QC' tab is active, showing four data quality sections: 'geotiff', 'gga', 'met', and 'svp'. Each section contains a table of files with columns for 'Filename', 'Row Integrity', 'Delta-T', 'Velocity', 'Bounds', and 'Stats'. The 'geotiff' section has one file. The 'gga' section has two files. The 'met' section has two files. The 'svp' section has two files. Each file row includes a 'Show' button to view data statistics.

Open Vessel Data Management v2

Home Data Dashboard Main Position Weather Sound Velocity QA/QC Configuration Links

On System Status CS1502 Cruise ID 233.05 MB Cruise Size 2.54 TB Free Space

Main Position Weather Sound Velocity QA/QC

geotiff

Filename	Stats
CS1502/EM302/proc/CS1401_MB_DLY01_50m_WGS84_20120428.tif	Show

gga

Filename	Row Integrity	Delta-T	Velocity	Bounds	Stats
CS1502/NAV/POSMV-GGA_20120428-000000.Raw	✓	✓	⚠	✓	Show
CS1502/NAV/POSMV-GGA_20120429-000000.Raw	✓	✓	⚠	✓	Show

met

Filename	Row Integrity	Delta-T	Bounds	Stats
CS1502/METOC/MET-M01_20120428-000000.Raw	✓	✗	✓	Show
CS1502/METOC/MET-M01_20120429-000000.Raw	✓	✗	✓	Show

svp

Filename	Row Integrity	Delta-T	Bounds	Stats
CS1502/METOC/Sound-Velocity-Probe_20120428-000000.Raw	⚠	✓	⚠	Show
CS1502/METOC/Sound-Velocity-Probe_20120429-000000.Raw	⚠	✓	✓	Show

Illustration 8: Data Quality tab within the Data Dashboard

4.3.4 Configuration

The Configuration section of the OpenVDM web-application controls the various OpenVDMv2 configuration options. Like the Data Dashboard sections, this section is divided in to several sub-sections.

While the heading the Configuration Section appears identical to the rest of the OpenVDMv2 web-application, the System Status Panel is now a button for turning on/off OpenVDMv2 transfers.

Main Tab

The Main Tab displays the current status of all transfers and provided links to the most commonly used OpenVDMv2 functions.

The Cruise Status Panel is to create a new cruise, run the end-of-cruise script and edit the current cruiseID/Start Date.

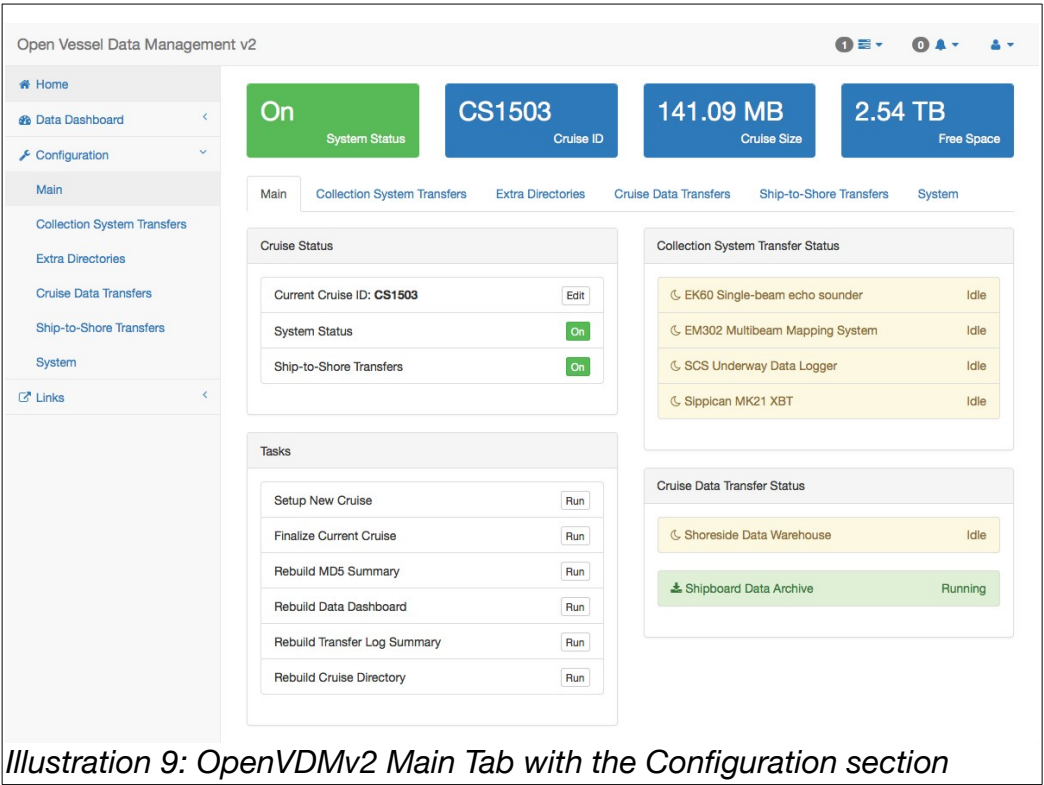
In the lower-left is the Tasks panel. Here the user can initial the various tasks of OpenVDM not related to transfers.

- **Setup New Cruise** – Define a new cruiseID and start date, create a new cruise directory structure, initialize the data dashboard, MD5_Summary files and transfer log summary.
- **Finalizing Current Cruise** – Run all Collection System Transfers one last time, update the Data Dashboard, Copy the contents on PublicData to the cruise data directory, updated the MD5 Summary file
- **Rebuilding MD5 Summary** – Rebuild the contents of the MD5_Summary files based on the files currently in the cruise data directory
- **Rebuilding Data Dashboard** – Rebuild the dataDashboard JSON files based on the raw data files in the cruise data directory.
- **Rebuilding Transfer Log Summary** – Rebuild the transfer log summary using the log files in the transfer log directory.
- **Rebuilding Cruise Directory** – Create any missing subdirectories in the cruise data directory, fix incorrect file permissions.

In the upper-right is the Collection System Transfer Status panel. This panel shows the current status of the currently enabled Collection System Transfers.

OpenVDMv2 User's Guide

In the lower-right is the Cruise Data Transfer Status panel. This panel shows the current status of the currently enabled Cruise Data Transfers.



Collection System Transfers Tab

This tab manages the Collection System Transfer profiles. The default view is a listing containing all the configured Collection System Transfers.

To Edit/Delete/Test/Run any of the transfers, click the corresponding link. To toggle the enable/disable status of a collection system transfer, click the corresponding button in the **Enable** column.

At the bottom of the list is a button to add additional Collection System Transfers.

The screenshot displays the 'Collection System Transfers' tab in the Open Vessel Data Management v2 application. The interface includes a sidebar with navigation links (Home, Data Dashboard, Configuration, Main, Collection System Transfers, Extra Directories, Cruise Data Transfers, Ship-to-Shore Transfers, System, Links) and a main content area. At the top of the main area, there are four summary cards: 'On' for System Status, 'CS1503' for Cruise ID, '141.12 MB' for Cruise Size, and '2.54 TB' for Free Space. Below these cards is a tabbed interface with 'Main' selected. The main content area features a table of Collection System Transfers and a 'Page Guide' section.

Name	Action	Enabled
SBE 911+ CTD	Edit / Delete / Test / Run	
EK60 Single-beam echo sounder	Edit / Delete / Test / Run	
EM302 Multibeam Mapping System	Edit / Delete / Test / Run	
Ocean Surveyer 75kHz ADCP	Edit / Delete / Test / Run	
SCS Underway Data Logger	Edit / Delete / Test / Run	
Thermo-salinograph	Edit / Delete / Test / Run	
Workhorse 300kHz ADCP	Edit / Delete / Test / Run	
Sippican MK21 XBT	Edit / Delete / Test / Run	

[Add New Collection System Transfer](#)

Page Guide

This page is for managing Collection System Transfers. A Collection System Transfer is an OpenVDM-managed file transfer from a data acquisition system to the Shipboard Data Warehouse.

Clicking an [Edit](#) link will redirect you to the corresponding "Edit Collection System Transfer Form" where you can modify the Collection System Transfer settings.

Clicking a [Delete](#) link will permanently delete the corresponding Collection System Transfer. There is a confirmation window so don't worry about accidental clicks.

Clicking a [Test](#) link will verify the corresponding Collection System Transfer configuration is valid. A window will appear displaying the test results. If there is a in a row, the corresponding Collection System Transfer has encountered an error. Click [Test](#) to diagnose the problem.

Clicking a [Run](#) link will manually trigger the corresponding Collection System Transfer to start. If the Collection System Transfer is currently running, this link is not present.

Clicking a [Stop](#) link will manually trigger the corresponding Collection System Transfer to stop immediately. If the Collection System Transfer is not currently running, this link is not present.

Extra Directories Tab

This tab manages the Extra Directory profiles. The default view is a listing containing all the configured Extra Directories.

To Edit/Delete any of the directories, click the corresponding link. To toggle the enable/disable status of an extra directory, click the corresponding button in the **Enable** column.

At the bottom of the list is a button to add additional Extra Directories.

Cruise Data Transfers Tab

This tab manages the Cruise Data Transfer profiles. The default view is a listing containing all the configured Cruise Data Transfers.

To Edit/Delete/Test/Run any of the transfers, click the corresponding link. To toggle the enable/disable status of a cruise data transfer, click the corresponding button in the **Enable** column.

At the bottom of the list is a button to add additional Cruise Data Transfers.

Ship-to-Shore Transfers Tab

This tab manages the Ship-to-Shore Transfer profiles. The default view is a listing containing all the configured Ship-to-Shore Transfers.

To Edit/Delete any of the transfers, click the corresponding link. To toggle the enable/disable status of a ship-to-shore transfer, click the corresponding button in the **Enable** column.

At the bottom of the list is a button to add additional Ship-to-Shore Transfers.

System Tab

This tab manages other OpenVDM behaviors.

System Behaviors

- Ship-to-Shore Bandwidth Limit – This sets the maximum bandwidth to be used for ship-to-shore transfers to the Shore-side Data Warehouse.
- Data File size Limit for MD5 Checksum – This sets the maximum file size that a MD5 checksum will be calculated for.

OpenVDM Specific Ship-to-Shore Transfers

- Dashboard Data – This enables/disables uploading data dashboard files to the Shore-side Data Warehouse as part of the standard ship-to-shore transfers
- MD5 Summary – This enables/disables uploading the MD5 checksum summary files to the Shore-side Data Warehouse as part of the standard ship-to-shore transfers

- OpenVDM Configuration – This enables/disables uploading OpenVDM configuration file to the Shore-side Data Warehouse as part of the standard ship-to-shore transfers
- Transfer Logs – This enables/disables uploading the transfer log files to the Shore-side Data Warehouse as part of the standard ship-to-shore transfers

OpenVDM Required Directories

- Dashboard Data – This allows the vessel operator to specify the name of directory that will contain the dashboard data files.
- Misc. cruise docs, pictures and data – This allows the vessel operator to specify the name of the directory where files from the PublicData directory will be moved to when the cruise is finalized.
- Transfer Logs – This allows the vessel operator to specify the name of the directory that will contain the transfer log files.

Data Warehouses

- Shipboard Data Warehouse Configuration – Edit/Test the configuration of the shipboard data warehouse. Items that can be edited include the IP address of the data warehouse, directory path to the root of the cruise data directories, directory path the PublicData directory and the user account to use when setting file permissions.
- Shoreside Data Warehouse Configuration – Edit/Test the configuration of the shoreside data warehouse. Items that can be edited include the IP address of the data warehouse, directory path to the root of the cruise data directories and the user account/password to use when connecting to the shoreside data warehouse.

OpenVDMv2 User's Guide

Configuration - Open Vessel Dat...

capablesolutions.dyndns.org:8180/OpenVDMv2/config/system

Search

Most VisitedGetting StartedTravelWind/Waves/We...NewsPopularTech RelatedPersonalFoodRSS FeedsWorkEX1304L1 video

Open Vessel Data Management v2

Home

Data Dashboard

Configuration

Main

Collection System Transfers

Extra Directories

Cruise Data Transfers

Ship-to-Shore Transfers

System

Links

OnSystem Status

CS1503Cruise ID

141.12 MBCruise Size

2.54 TBFree Space

Main

Collection System Transfers

Extra Directories

Cruise Data Transfers

Ship-to-Shore Transfers

System

System Behaviors	Action	Enabled
Ship-to-Shore Transfer Bandwidth Limit: Unlimited	Edit	Off
Data Filesize Limit for MD5 Checksum: 10 MB	Edit	Off

OpenVDM Specific Ship-to-Shore Transfers	Action	Enabled
Dashboard Data	Edit	On
MD5 Summary	Edit	On
Transfer Logs	Edit	Off

OpenVDM Required Directories	Action
Dashboard Data	Edit
Misc. cruise docs, pictures and data.	Edit
Transfer Logs	Edit

Data Warehouses	Action
Shipboard Data Warehouse (SBDW)	Edit / Test
Shoreside Data Warehouse (SSDW)	Edit / Test


Page Guide

This page is for managing OpenVDM specific items. These items cannot be deleted.

The **System Behaviors** table is for setting variables used in various OpenVDM programs. Click the corresponding [Edit](#) link to modify the value. For all of these variables there is a default value. Click the corresponding **Enable** button to either use of the set value or ignore the set value and use the system default.

The **OpenVDM Specific Ship-to-Shore Transfers** table is for managing Ship-to-Shore Transfers of data generated by OpenVDM such as the Dashboard Data and RSS Feeds. Click the corresponding [Edit](#) link to change the behaviors of these transfers. Use the button in the **Enable** column to enable/disable the corresponding transfer.

The **OpenVDM Required Directories** table is for managing locations of directories within the cruise data directory that are required by OpenVDM. Click the corresponding [Edit](#) link to change these locations.

The **Data Warehouses** table is for managing the Shipboard Data Warehouse and the Shoreside Data Warehouse. Click [Edit](#) to view/modify the warehouse's settings. Click [Test](#) to verify the corresponding Data Warehouse configuration is valid. A window will appear displaying the test results. If there is a  in a row, there is an error with the corresponding Data Warehouse

5 Data Dashboard Visualization Plugin Development Guide

5.1 Overview

The time-series and GIS-based visualizations displayed in the OpenVDM Data Dashboard are built from JSON-formatted files dashboard data files. The dashboard data files are created by visualization plugins. The plugins read raw data files and create the JSON-formatted dashboard data files on a one-to-one basis (i.e. one dashboard data file for each raw data file).

Because no two ships are identical the visualization plugins must be developed custom for the data files generated by the vessel.

The dashboard data files must conform to simple JSON object schema (described below). This conformity allows OpenVDM to properly interpret the dashboard data files when visualizing the data and populating the Data Quality dashboard tab. OpenVDM places no constraints on how the vessel develops the plugin so long as the output conforms to the required schema.

To help vessel operators develop visualization plugins, several plugin for common data types (standard NMEA0183) are included in the `/usr/local/bin/dashboardDataScripts` directory.

5.2 The Processing Workflow

Dashboard data files are created/updated by OpenVDM when the associating raw file is pulled into the cruise data directory for the first time or updated. This allows for a faster and more efficient updating of the data dashboard.

Step 1: Pulling in new and/or updated files from the source collection system. At the end of each Collection System Transfer a manifest of all files pulled from the collection system source into the Data Warehouse is compiled and forwarded to all subsequent processes.

Step 2: Confirm the presence of new and/or updated raw data files. If no new or updated files were transfer then there is no reason to commit resources for creating/updating dashboard data files... the process exits. If new/updated files were transferred to the data warehouse, those filenames along with the name of the source

collection system transfer are passed to the data dashboard worker.

Step 3: Confirm a plugin exists for the collection system where the files arrived from. Not all collection systems may have or need a data dashboard plugin. The first step within the data dashboard worker is to verify the presence of a data dashboard plugin for the source collection system.

Step 4: Verify a processor exists for the specific file type. In a given collection system transfer, it may not be necessary to build a data dashboard file for all file types. The plugin is fed a list of files. The next step in the process is to confirm that the file type of the next file being processed actually requires processing into a JSON-formatted data dashboard file.

Step 5: Process the raw file and return the JSON-formatted, schema conforming, Dashboard Data file.

Step 6: Save the Dashboard Data file to the DashboardData directory within the Cruise Data Directory. The Dashboard Data files are stored within a directory structure that mimics the directory structure used for storing the raw data files.

5.3 The Dashboard Data JSON Schema

Below is the schema for the JSON-formatted dashboard data files. There are 3 main sections to the schema.

```
{
  "visualizerData":[
    // for time-series data
    {
      data: [
        <int>(Unix Epoch),
        <int>/<float>
      ],
      unit: <string>,
      label: <string>
    }
    // for GIS-based data
    {
      <GeoJSON Feature Collection>
    }
  ],
  "stats":[
    {
      "statName": <string>,
      "statUnit": <string>,
      "statType": <string> ("bounds"|"timeBounds"|"valueValidity"|"rowValidity"),
      "statData": [
        <object>
      ]
    }
  ],
  "qualityTests":[
    {
      "testName": <string>,
      "results": <string> ("Passed"|"Warning"|"Failed")
    }
  ]
}
```

5.3.1 visualizerData

The “visualizerData” array contains the data to visualize.

For time-series data, the visualizerData array contains the following elements:

- data: an array containing the data points. Each point is a two-element array with the first element being an integer representing number of milliseconds since the Unix epoch. The second element is the data value. This can be an integer or floating point number.
- unit: a string describing the data value’s unit of measure.
- label: a string containing the readable name for the dataset.

For GIS-based data, the visualizerData array contains GeoJSON-formatted featureCollections.

5.3.2 stats

The “stats” array contains objects that fully described any statistical data calculated from the raw data files. Each object contains the following elements:

- statName: a string containing the readable name of the statistic
- statUnit: a string containing the value of the statistic
- statType: the type of the statistic. Types are used to by OpenVDM to understand how to present the statistic in the data quality tab of the data dashboard. The acceptable statTypes are: “bounds”, “timeBounds”, “geoBounds”, “valueValidity” and “rowValidity”
- statData: an object containing the data related to the statistic. The contents of this object is different for each statType.
 - For bounds, the object contains a 2-element array containing a min and max value.
 - For timeBounds, the object contains a 2-element array containing a start and stop time represented as the number of milliseconds since the Unix epoch.
 - For geoBounds, the object contains a 4-element array containing the North-most latitude, East-most longitude, South-most latitude and West-most longitude.
 - For valueValidity, the object contains a 2-element array containing the number of rows in the raw data file where the value was valid and the number of rows where the value was invalid.
 - For rowValidity, the object contains a 2-element array containing the number of rows in the raw data file that were properly formatted and the number of rows where there were formatting errors.

5.3.3 qualityTests:

The “qualityTests” array contains objects that fully described any quality test run against the raw data files. Each object contains the following elements:

- testName: a string containing the readable name of the test performed
- results: a string containing the results of the test. Acceptable values include: “Passed”, “Warning”, “Failed”.

5.4 A working example of a visualization plugin

Plugins are organized is as follows:

- Plugins are associated with Collection System Transfers on a 1-to-1 basis (i.e. there can be no more than only one plugin per collection system transfer)
- The root plugin must be written in python (currently v2.7).
- The name of the plugin is strictly defined. It is the short name of the collection system transfer followed by “_dashboardData.py” (i.e. If a vessel has a collection system transfer named “SBE911” the name of the plugin must be “SBE911_dashboardData.py”)
- Plugins are stored in the /usr/local/bin/OVDM_dashboardDataScripts directory.

In the repository, within the “/usr/local/bin/OVDM_dashboardDataScripts directory” there are two disabled plugins (SCS_dashboardData.py.dist, EM302_dashboardData.py.dist).

These plugins are considered disabled for two reasons:

- The files do not match the require naming convention for plugins (because of the “.dist” suffix).
- The default install does not include collection system transfers named “SCS” and “EM302”

*** Scientific Computing System (SCS) is an underway data acquisition system developed by the US National Oceanic and Atmospheric Administration (NOAA) that is used on all NOAA vessels and many of the vessel in the US Academic Research Fleet. SCS is used for writing underway data (position, heading, wind speed/direction, sea surface temperature, etc) to file. SCS typically creates a new data file for each data type and prepends a timestamp to each incoming row of data before writing it to file. OpenVDM is NOT in any way coupled to SCS but understanding what SCS is and what it does should help explain how the SCS plugin works.

Plugins must adhere to a minimal command-line argument schema:

```
<CollectionSystemName>_dashboardData.py [--datatype] [filename]
```

Calling the plugin with the --dataTyPe flag returns the dashboard data file's data type. Returning a data type (i.e. ‘gga’, ‘svp’, ‘hdt’, etc) let’s OpenVDM know that there is a processor for this raw file. If this command returns nothing then OpenVDM assumed there is no processor for this

OpenVDMv2 User's Guide

specific raw file and moves on to the next raw file.

The dashboard data files's data type is used by the OpenVDM Data Dashboard to properly place and visualize the dashboard data files. (i.e. draw the data as a time-series or as a trackline on a map, also which tabs in the data dashboard should display which datasets).

Calling the plugin without the --dataType flag returns the JSON-formatted dashboard data file contents as a string. OpenVDM will handle writing this string to file, naming the file, and saving the file to the correct directory.

Let's take a closer look at SCS_dataDashboard.py...

The SCS plugin creates the dashboard data files for a lot of data types ranging from GPS positions to hull-mounted sound speed profilers. In the plugin provided as part of the repository only 5 types are handled but on some ships the SCS plugin handles over 20 different data types. In some cases the same processing steps need to run on files representing different sensors (i.e. a vessel with 4 GPS sensors sending NMEA0183 GGA formatted data to SCS). To make this manageable the SCS plugin pushes the processing of raw data files to other smaller scripts depending on the raw file's data type. Take a look at the "fileTypeFilters" array forward the beginning of the SCS_dataDashboard.py file:

```
fileTypeFilters = [
    {
        "dataType": "twind",
        "regex": "*METOC/TrueWind-RAW_*.Raw",
        "command": ['python', SCRIPT_DIR + 'twind_parser.py']
    },
    {
        "dataType": "tsg",
        "regex": "*METOC/TSG-RAW_*.Raw",
        "command": ['python', SCRIPT_DIR + 'tsg_parser.py']
    },
    {
        "dataType": "met",
        "regex": "*METOC/MET-M01_*.Raw",
        "command": ['python', SCRIPT_DIR + 'met_parser.py']
    },
    {
        "dataType": "svp",
        "regex": "*METOC/Sound-Velocity-Probe_*.Raw",
        "command": ['python', SCRIPT_DIR + 'svp_parser.py']
    },
    {
        "dataType": "gga-posmv",
        "regex": "*NAV/POSMV-GGA_*.Raw",
        "command": ['python', SCRIPT_DIR + 'gga_parser.py']
    }
]
```

This array defines the dataType, the regex expression for matching files of a specific type by their filename and the script to run that processes the raw file and returns the JSON-formatted

dashboard data file. When the `--dataType` flag is called, the plugin simply compares the raw file name to the regex expressions and returns the `dataType` field in the matching row. When the `--dataType` flag is not called, the plugin forwards the file to the processor for that data type. New data types can be easily added to the plugin by simply adding a row to this array. This approach promotes code reuse. On the vessels with multiple GPS sensors, the `'gga_parser.py'` script is used to process all of the vessel's GGA-formatted data files. This approach also simplifies debugging issues because the subprocess can be called independent of the plugin.

Now let's take a look at one of the processors: `gga_parser.py`...

This file is in the `/usr/local/bin/OVDM_dashboardDataScripts` directory but is named `"gga_parser.py.dist"`. You will see from the import statements that this script uses Pandas, numpy and geopy to do some of the processing and for calculating some of the statistics. You will need to install Pandas and geopy prior to running this plugin for the first time. It is recommended to use PIP for these installs instead of apt-get because the apt-get versions are several versions behind the current releases.

The next dozen lines after the import statements define the constants used through out the script.

- `RAW_COLUMNS` defines the columns in the raw data file
- `PROC_COLUMNS` defined the columns from the `RAW_COLUMNS` that the file is actually interested in
- `CROP_COLUMNS` defines the columns used in the output.
- `MAX_VELOCITY` defines the max velocity of the vessel. This number is used to detect GPS "jumps"
- `MIN_LATITUDE`, `MAX_LATITUDE`, `MIN_LONGITUDE` and `MAX_LONGITUDE` define the min/max bounds for latitude and longitude. Used to detect bad data.
- `MAX_DELTA_T` defines the maximum interval that can exists between GPS fixes. Also used for error detection.
- `RESAMPLE_INTERVAL` define the resolution of the output data (1min in this case)

The rest of the file is the parsing of the raw file, calculating the stats, running the QA tests, converting the output to GeoJSON and building the dashboard data schema to be returned to stdout.

This exact layout is used for all the data collected by SCS. Creating a new processor is as

simple as copy/pasting once of these existing processors, tweaking the RAW_COLUMNS, PROC_COLUMNS and CROP_COLUMNS variables, defining QA tests, and updating the output labels and units.

6 Customizing the Data Dashboard

The Data Dashboard layout is controlled via the `datadashboard.yaml` file located in the `/usr/local/etc/openvdm` folder. This yaml-formatted file provides the OpenVDM web-application with all the information it needs to setup the various tabs in the dashboard, what data types to include on each tab, and how to visualize each data type.

By default OpenVDM includes the javascript files for visualizing dashboard data files as time-series graphs and geographic maps. There is also a way for new visualization types and custom styles to be applied to individual datatypes.

6.1 `datadashboard.yaml`

Below is the header from the `datadashboard.yaml` file. The header provides a sample configuration block and an explanation of each line's significance.

```
# SAMPLE BLOCK --- INDENTS ARE IMPORTANT TO THE YAML-FORMAT ---
#- title: Position
#  page: position
#  view: default
#  cssArray:
#    - leaflet
#  jsArray:
#    - dataDashboardDefault
#    - highcharts
#    - leaflet
#  placeholderArray:
#    - plotType: map
#      id: map
#      heading: Position
#      dataArray:
#        - dataType: gga
#          visType: geoJSON
#        - dataType: geotiff
#          visType: tms

# Breakdown of SAMPLE BLOCK
# Title of the Tab
#- title: Position

# Internal-use title, this should be all lower-case, no spaces, and unique from the
# "page" values used for other tabs
#  page: position

# The MVC view used to format the tab. OpenVDM includes a "default" view that can be
# used for displaying maps, and time-series data but additional custom view can be
# developed and used. For custom views, specify the filename of the view (which must
# be located in the ./app/views/DataDashboard directory of the OpenVDM web-application)
# minus the php suffix (i.e use "customView" for "customView.php")
#  view: default
```

OpenVDMv2 User's Guide

```
# Additional css files to include when rendering the tab. Only specify the filename of
# the css file (which must be located in the ./app/templates/default/css directory of
# the OpenVDM web-application) minus the .css suffix (i.e use "customCSS" for
# "customCSS.css").
#
# SPECIAL CASES:
# If the view needs to leverage leaflet simple add "leaflet".
#   cssArray:
#     - leaflet

# Additional js files to include when rendering the tab. Only specify the filename of
# the js file (which must be located in the ./app/templates/default/js directory of
# the OpenVDM web-application) minus the .js suffix (i.e use "customJS" for
# "customJS.js").
#
# SPECIAL CASES:
# If the view needs to leverage leaflet simple add "leaflet".
# If the view needs to leverage highcharts, simple add "highcharts"
# If the view needs to leverage the highcharts export plugin use "highcharts-exporting"
#   jsArray:
#     - dataDashboardDefault
#     - highcharts
#     - leaflet

# Placeholders are the bootstrap-styled panels that contain the <div> blocks where data
# will be visualized that are rendered by the view.
#   placeholderArray:

# The "plotType" defines how the view should render the panel and <div> block
#   - plotType: map

# The "id" specifies the id attribute of the <div> block. This is useful when needing
# to find/modify the contents of the <div> block using javascript. The "id" should be
# unique for each placeholder on an individual tab
#   id: map

# The "heading" is the text to display in the panel-header
#   heading: Position

# The "dataArray" is the dashboard data to display within a single <div> block. Each
# element of this array includes a dashboardData 'dataType' (i.e. 'gga') and a
# 'visType'. The 'visType' specifies how that data should be rendered (geoJSON vs tms)
#   dataArray:
#     - dataType: gga
#       visType: geoJSON
#     - dataType: geotiff
#       visType: tms
#
```