

# Open Vessel Data Management User's Guide

Version 2.9

Last Updated: October 31, 2022

## Table of Contents

1	Overview.....	1
1.1	What is OpenVDM?.....	1
1.2	What does OpenVDM do?.....	1
1.3	OpenVDM Core Philosophies.....	2
2	Quick Start.....	4
2.1	Creating a New Cruise.....	4
2.2	Finalizing the Current Cruise.....	4
2.3	Enabling Automatic Data Transfers.....	5
2.4	Disabling Automatic Data Transfers.....	5
3	OpenVDM System Components.....	6
3.1	The Shipboard Data Warehouse.....	6
3.2	The Shore-side Data Warehouse.....	7
4	The OpenVDM WebUI.....	7
4.1	Basic Concepts.....	7
4.2	The Data Dashboard and DashboardData files.....	9
4.3	The WebUI Layout.....	10
5	Data Dashboard Visualization Plugin Development Guide.....	24
5.1	Overview.....	24
5.2	The Processing Workflow.....	24
5.3	The Dashboard Data JSON Schema.....	26
5.4	A working example of a visualization plugin.....	28
6	Customizing the Data Dashboard.....	30
6.1	datadashboard.yaml.....	30

## Table of Screen Captures

Illustration 1:	OpenVDM Header.....	10
Illustration 2:	OpenVDM Navigation Elements.....	11
Illustration 3:	OpenVDM Messages Screen.....	12
Illustration 4:	OpenVDM Home Section.....	13
Illustration 5:	OpenVDM Data Dashboard Section.....	14
Illustration 6:	Sample of GIS-based data displayed within the Data Dashboard.....	15
Illustration 7:	Sample of time series-based data displayed within the Data Dashboard.....	16
Illustration 8:	Data Quality tab within the Data Dashboard.....	17
Illustration 9:	OpenVDM Main Tab with the Configuration section.....	19

# 1 Overview

This document is intended to provide an overview of Open Vessel Data Management v2.9 (OpenVDM) and its use under normal circumstances and installation profile. While this guide is intended as a universal reference for all OpenVDM installations it may not account for every all customizations.

The document is organized as follows: Section 1 is an overview of OpenVDM. Section 2 is a quick start guide for the most common tasks. Section 3 describes the many components of OpenVDM. Section 4 describes the web-application for OpenVDM

## 1.1 What is OpenVDM?

The most important thing to understand about OpenVDM is the exact set of tasks it is built to accomplish. OpenVDM is NOT a data acquisition system (DAS). OpenVDM does NOT write data from a sensor or data stream to a file. OpenVDM is a data management tool and thus it manages files created by data acquisition systems and provides mechanisms for vessel operators to build custom data management workflows. OpenVDM is NOT designed to fix problems within data files or correct misnamed files. It IS a tool to help vessel operators find these types of errors and so that they can be addressed at the source.

The goal of OpenVDM is to take data files from multiple data acquisition systems running throughout a 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 OpenVDM do?

OpenVDM strives to help vessel operators organize files created during a cruise into a logical directory structure and make that directory structure easily and safely available to shipboard personnel.

To accomplish this goal, OpenVDM copies data from the various remote workstations (where the data files are created) to a cruise-specific 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.

OpenVDM also strives to make configuration, control and operation as quick and as painless as

possible, allowing vessel operators and technicians to focus on other tasks.

## **1.3 OpenVDM Core Philosophies**

The design and continuing evolution of OpenVDM 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**

OpenVDM 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. The goal of the OpenVDM development team is to support the transport methods that the vessel operator are most comfortable with.

### **1.3.2 Minimal Conformity Requirements**

OpenVDM strives to help vessel operators organize their collected datasets to the operator's requirements and not to one defined by OpenVDM.

OpenVDM 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 OpenVDM 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 OpenVDM 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... with one exception**

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

## OpenVDM User's Guide

exception to this rule is that the vessel operator can specify that a transfer perform a “hard sync”. This means that if the files on the source are renamed or deleted those changes will propagate through the transfer so long as the updated list of files still meet the directory and file naming requirement specified in the OpenVDM configuration.

## 2 Quick Start

This section is an abbreviated guide for performing the most common tasks.

### 2.1 Creating a New Cruise

1. Define the new cruise ID and start date.<sup>1</sup>
  - a) Goto: Configuration → Main Tab. This requires authenticating to the system.
  - 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.

\*\*\* Clicking the calendar icon will open a date picker widget.
  - d) When finished, click “Create”.

\*\*\* If you enter a cruise ID that already exists you will be asked to enter another.
  - e) 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 enabled systems. Resolve any error's encountered during these tests.
3. Enable the automated transfers. Click the red “System Status” Panel. This will cause the system status to change from “Off” (in red) to “On” (in green).
4. Move on to something else... OpenVDM will handle things from here.

### 2.2 Finalizing the Current Cruise

1. Goto: Configuration → Main Tab. This requires authenticating to the system.
2. Click the “System Status” Panel to disable the automatic transfers. This will cause the system status to change from “On” (in green) to “Off” (in red).
3. Click the blue “Run End-of-Cruise Tasks” button located within the “Cruise Control” panel.

---

<sup>1</sup> Specifying the end date and start/end ports is recommended but optional

4. Wait for the “Finalizing Cruise” task and any subsequent tasks to complete. Monitor the status of the task in the task status dropdown. A “check” icon will appear to the left of the button text when the task has completed 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.
5. Manually run all enabled Cruise Data Transfers. This will copy any new data pulled to the Data Warehouse as part of the Cruise Finalization to the external storage device(s) / location(s).
6. 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. This is only needed for vessels that leverage OpenVDM’s ship-to-shore mechanisms.

## **2.3 Enabling Automatic Data Transfers**

1. Verify the automatic transfers are currently disabled. If the System Status panel is green and says “On” then the transfers are already enabled... you’re done.
2. Authenticate to the system if not currently.
3. Enable the automated transfers. Click the red “System Status” Panel. This will cause the system status to change from “Off” (in red) to “On” (in green).

## **2.4 Disabling Automatic Data Transfers**

1. Verify the automatic transfers are currently enabled. If the System Status panel is red and says “Off” then the transfers are already disabled... you’re done.
2. Authenticate to the system if not currently.
3. Disable the automated transfers. Click the green “System Status” Panel. This will cause the system status to change from “On” (in green) to “Off” (in red).

## **3 OpenVDM System Components**

### **3.1 The Shipboard Data Warehouse**

The Shipboard Data Warehouse (SBDW) is the core component of OpenVDM. This is the server that hosts the OpenVDM web-application, runs the background processes and stores cruise data.

#### **3.1.1 The Cruise Data Directory**

This is the directory structure on the SBDW containing all the files OpenVDM is managing for the current cruise.

#### **3.1.2 The Public Data Directory**

This is the directory on the SBDW where science personnel and crew can share files.

#### **3.1.3 The Visitor Information Directory**

This is the directory on the SBDW where marine technicians can post files for crew and science personnel to access.

#### **3.1.4 Samba and HTTP Access to Data Directories**

The default OpenVDM installation provides SMB (Windows) and http protocols for accessing the Cruise Data Directory, Public Data Directory and Visitor Information Directory. Access to the Cruise Data Directory is read-only for anonymous access. Read/Write access is provided to an account of the vessel operator's choosing. This is done to allow authorized personnel to quickly/easily address issues

#### **3.1.5 The OpenVDM Web User Interface (WebUI)**

The OpenVDM WebUI allows non-authenticated users to see the status of transfers. For authenticated users the WebUI provides full control of OpenVDM's behavior and configuration.

#### **3.1.6 The Job Broker (Gearman) and Process Manager (Supervisor)**

Behind the OpenVDM WebUI is an array of scripts and programs that perform OpenVDM's various background tasks and file transfer functions. Managing all of these



background processes is the responsibility of the Gearman and Supervisor. Gearman is a job broker that connects job requests to the appropriate script/program capable of performing the work (workers). Supervisor makes sure the correct number of workers are started at boot and automatically restarted if a process unexpectedly crashes.

## **3.2 The Shore-side Data Warehouse**

The Shore-side Data Warehouse (SSDW) is the optional shore-based counterpart to the SBDW. It is where data is sent during ship-to-shore transfers.

### **3.2.1 OpenVDM – Port Office**

Port Office is an optional web-application that can be installed on the SSDW for replicating data dashboard from current and past cruises shoreside. Port Office is still in development and will not be discussed much within this document.

## **4 The OpenVDM WebUI**

The OpenVDM WebUI provides an intuitive interface for both configuring and managing OpenVDM's various components and behaviors.

### **4.1 Basic Concepts**

When describing how to use the OpenVDM 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**

OpenVDM organizes data files within a directory structure by cruise. The cruiseID is the official designation for a cruise and is used for the top-level directory name. The cruiseID is usually very short, and often includes an incremented value (i.e. CS2001, CS2002, etc). The cruiseID is different than the cruise/project title.

Cruises created within OpenVDM must have a start date/time. This is the date/time the cruise begins. 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**

OpenVDM can manage data from a single deployable platform (ROV, AUV, HOV, etc).

When this functionality is used the data from that platform will be organized by lowering (deployment). The loweringID is the official designation for the deployment. This is usually very short, and often includes a prefix followed by an incrementing number (i.e. CS-001, CS-002, etc). This is different than the cruiseID.

When creating a new lowering within OpenVDM the lowering is given a start date/time. This is the date/time the lowering begins. Optionally an end date/time can be defined for when the lowering ended. The loweringID must be unique within 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

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

A Collection System is a source location files to be managed by OpenVDM. A Collection System Transfers is the unit of information used by OpenVDM to connect to a Collection System, find the appropriate files, copy the files to the SBDW and save the files in the appropriate sub-directory within the cruise data directory. This unit of information includes:

- How to connect to the collection system
- Where the data files exist on the collection system
- Filename filters
- Destination directory within the cruise data directory.

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 OpenVDM within the cruise data directory that are not associated with a collection system. These are typically created at the request of the vessel operator for vessel-specific reasons.

OpenVDM has several extra directory definitions for internal purposes. The location of these required directories can be customized by the vessel operator.

#### **4.1.6 Cruise Data Transfers**

Cruise Data Transfers are units of information within OpenVDM for configuring an automated copy/backup of all or a subset of the cruise data directory 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 by OpenVDM to identify and transfer subsets of the cruise data directory to the SSDW. 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 WebUI is for displaying visualizations of data files, reporting quality assurance test results and reporting statistical information. The visualizations, QA test results and stats are associated with individual files.

The Main and Data Quality tabs within the Data Dashboard section are part of the default OpenVDM installation. Additional tabs can be added to 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 interpret the raw data directly. It uses a JSON-formatted DashboardData files which contain a simplified and sub-sampled representation of the raw data, QA test results and data statistics. The extent of data displayed within the Data Dashboard section is dependent which files the vessel operator wants to visualize, how the vessel operator wants to visualize those files, what QA tests the vessel operator wants to run and what statistics the vessel operator wants to collect.

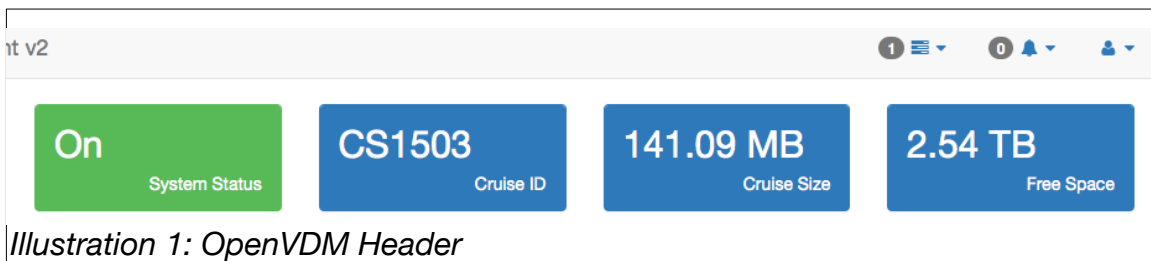
Getting from the raw data file to a DashboardData file is done via an OpenVDM plugin. Developing OpenVDM plugins is a topic unto itself and will not be discussed much further within this document.

## 4.3 The WebUI Layout

### 4.3.1 Common Elements

#### Header

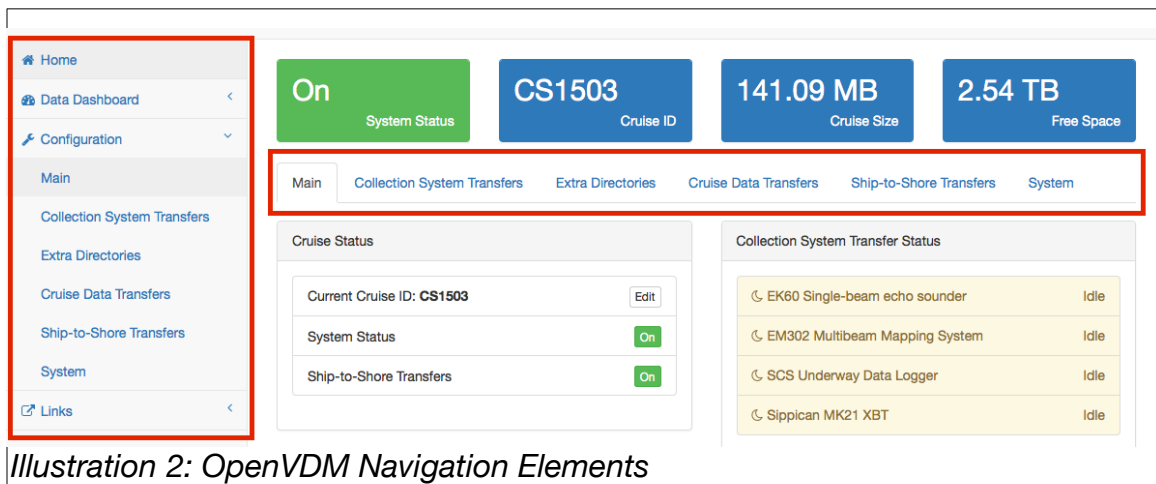
Every page in the web-application contains a common array of elements within the page header. The header includes a task process dropdown, messages dropdown, user management/logout dropdown, the system status, cruiseID, the size of the cruise data directory and the remaining free space on the shipboard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



*Illustration 1: OpenVDM Header*

#### 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: OpenVDM Navigation Elements*

### Authentication

The Home and Data Dashboard sections of the WebUI 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 OpenVDM. 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.

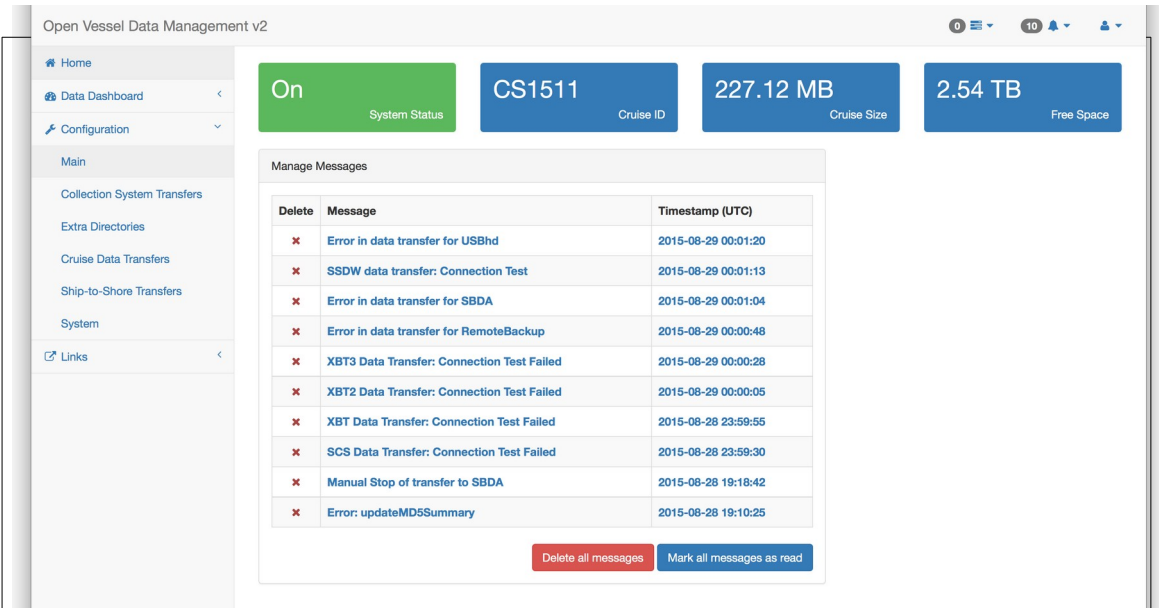


Illustration 3: OpenVDM Messages Screen

### 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.

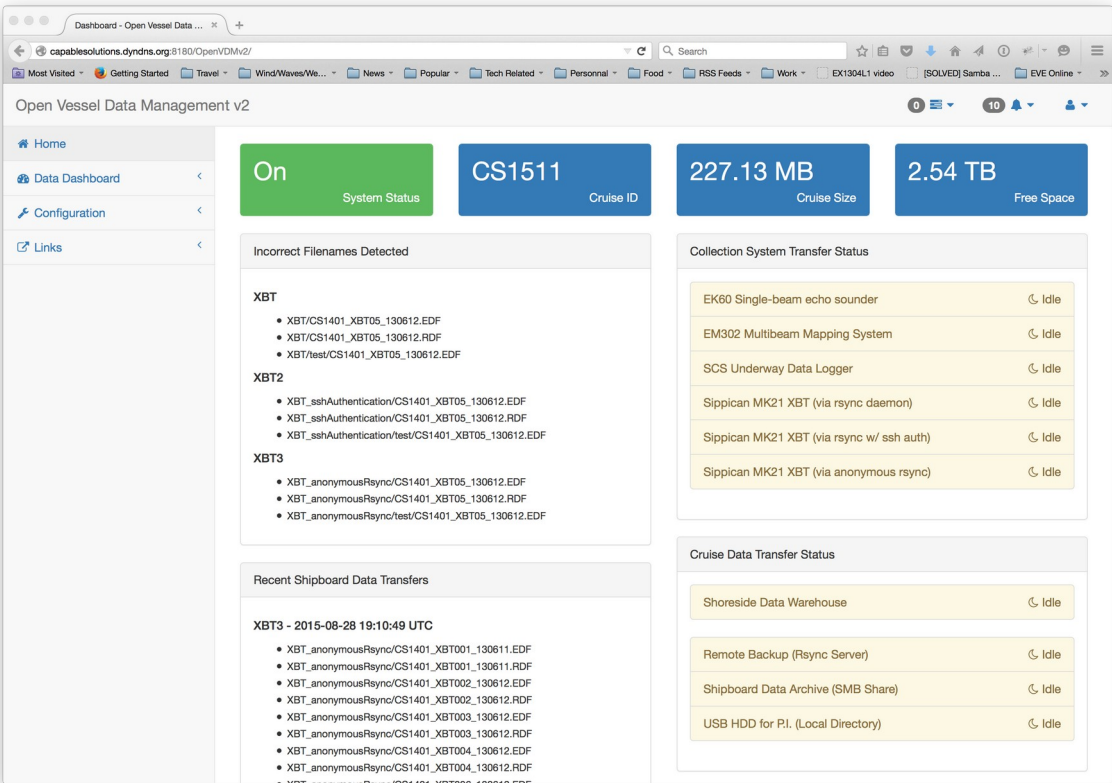


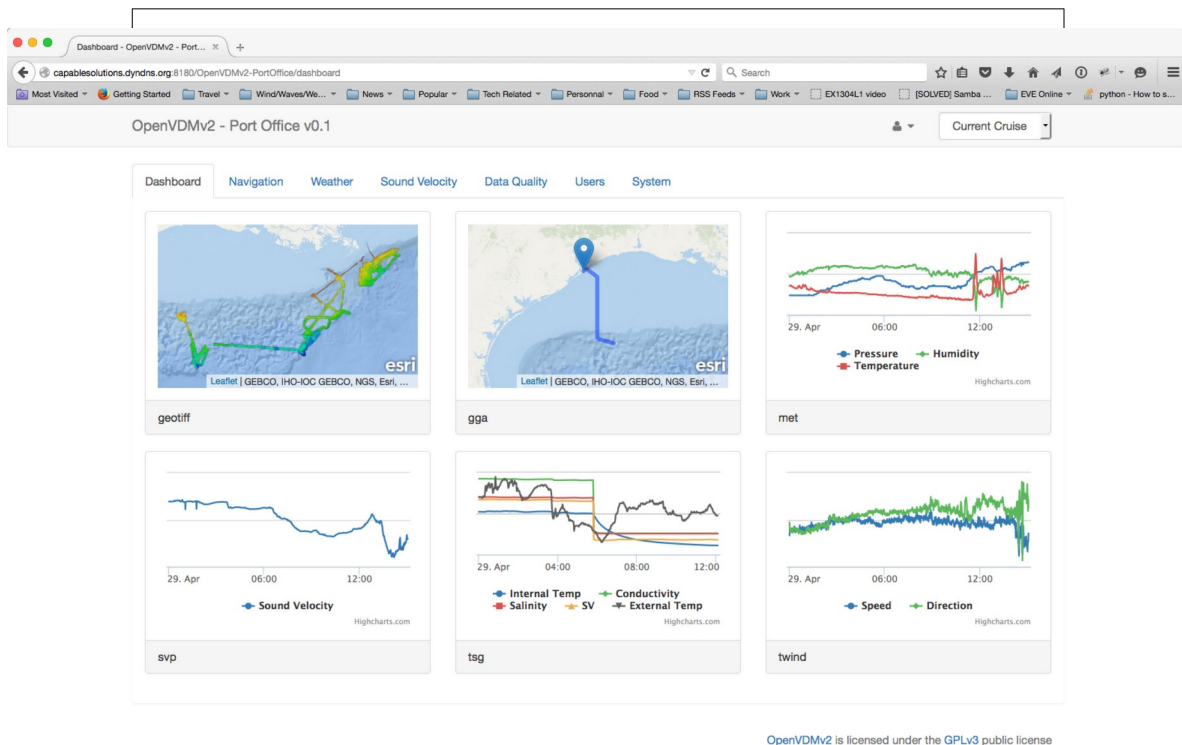
Illustration 4: OpenVDM Home Section

### 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.

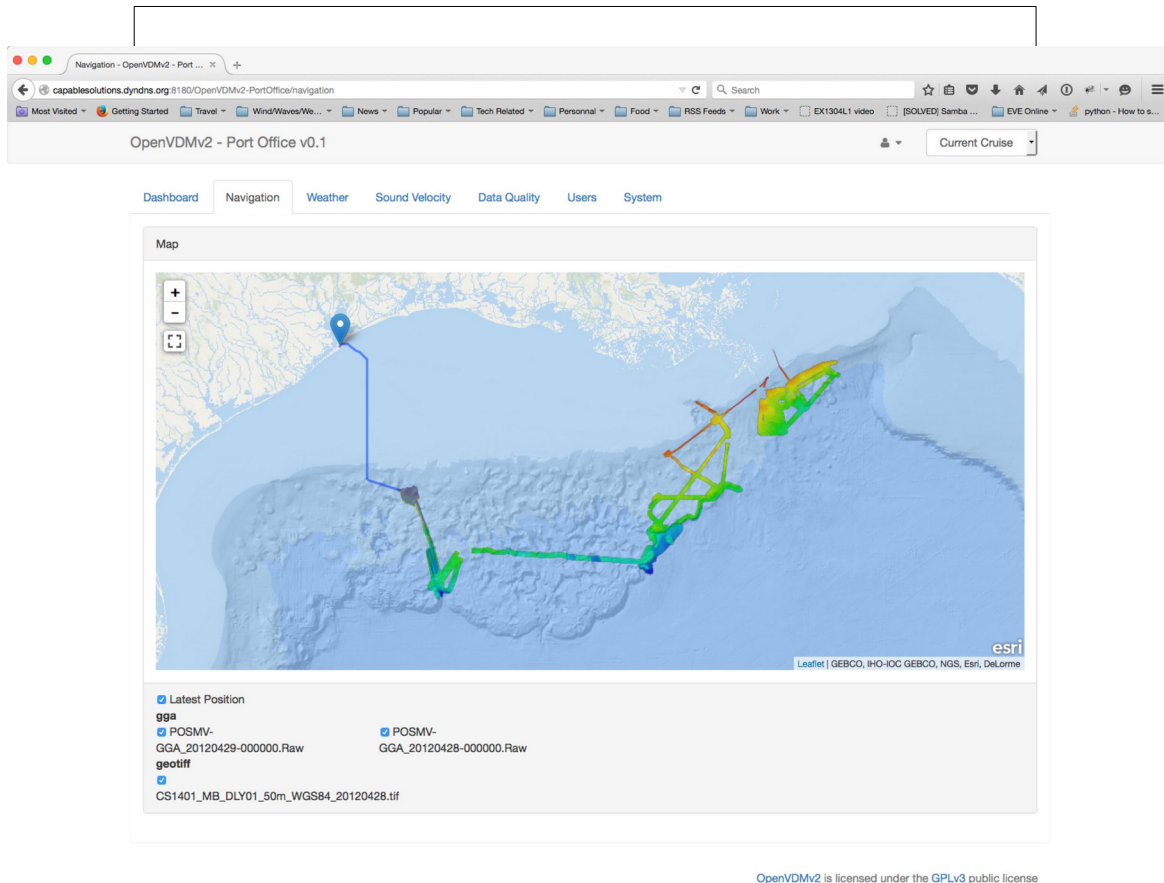


*Illustration 5: OpenVDM Data Dashboard Section*



## GIS Display

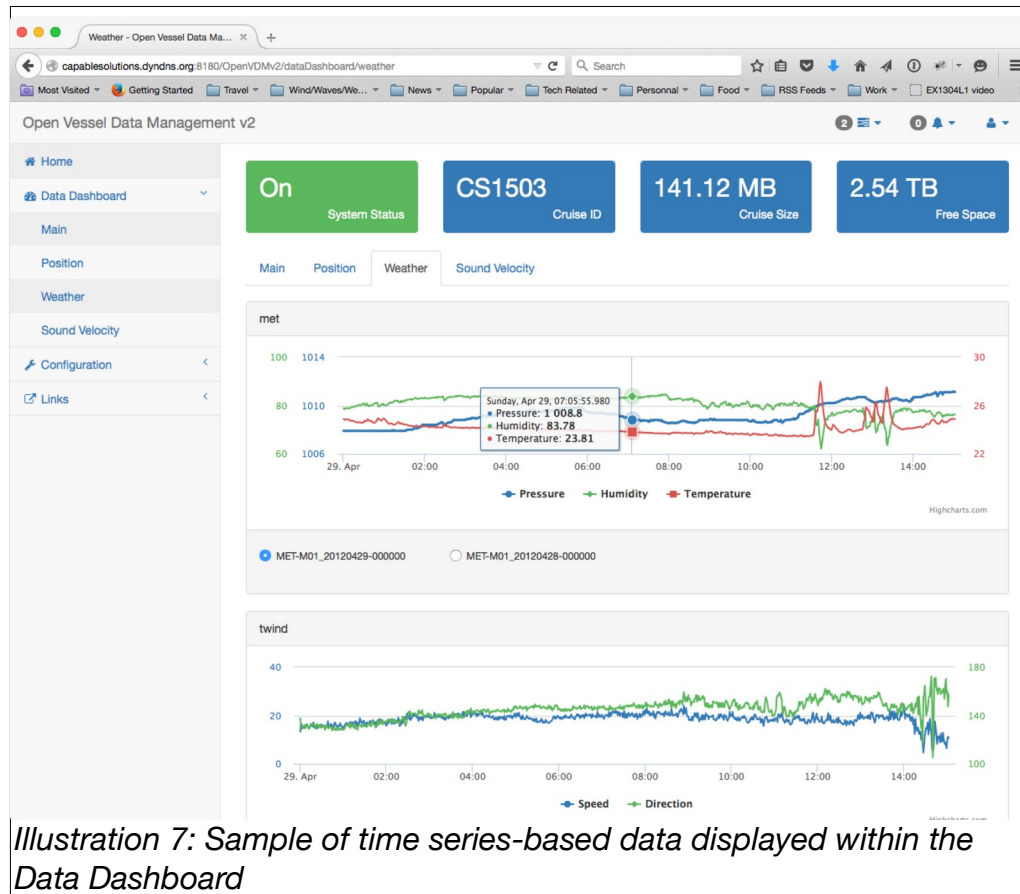
OpenVDM includes functions for visualizing geoJSON-formatted data and TMS tiles onto a Leaflet-based map. Checkboxes below the map can be selected to display additional data/tile sets.



*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.

QA/QC - Open Vessel Data Man...

capablesolutions.dyndns.org @160/OpenVDMv2/dataDashboard/qualityControl

Search

Most Visited

Getting Started

Travel

Wind/Waves/We...

News

Popular

Tech Related

Personal

Food

RSS Feeds

Work

EX1304L1 video

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 OpenVDM 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 OpenVDM web-application, the System Status Panel is now a button for turning on/off OpenVDM transfers.

#### Main Tab

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

OpenVDM 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.

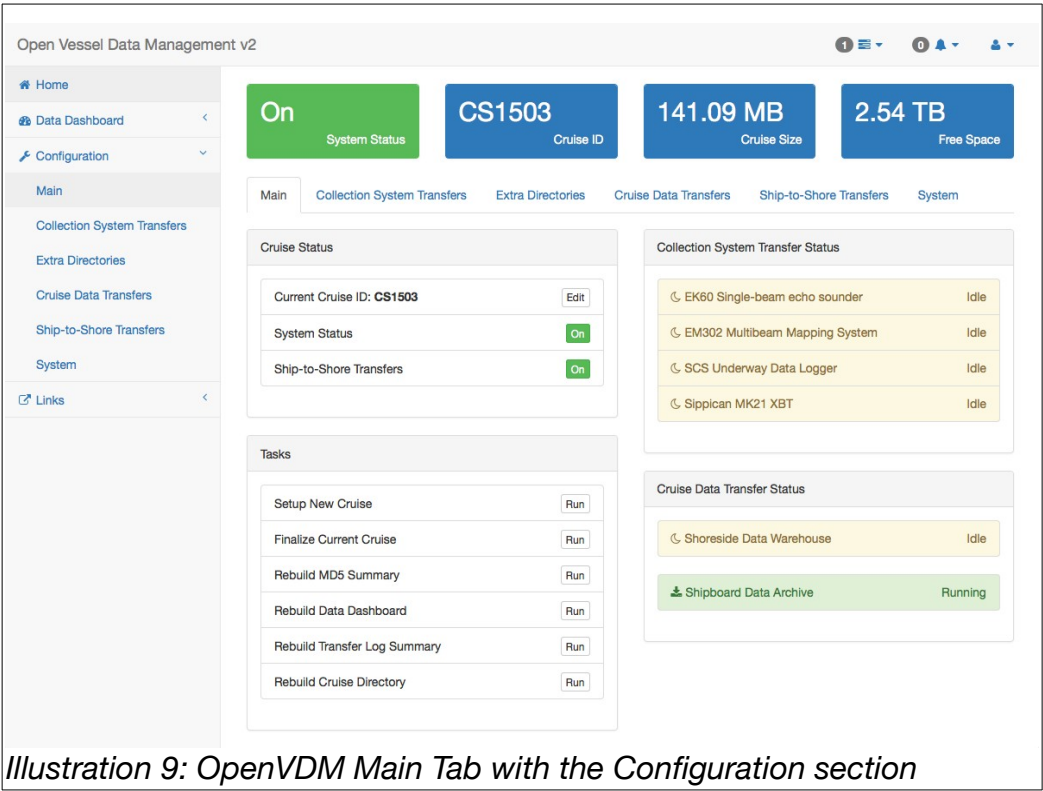


Illustration 9: OpenVDM Main Tab with the Configuration section

## 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 'Open Vessel Data Management v2' web application. The top navigation bar includes links for Home, Data Dashboard, Configuration, Main, Collection System Transfers, Extra Directories, Cruise Data Transfers, Ship-to-Shore Transfers, and System. The 'Collection System Transfers' tab is active, showing a table of configured transfers. Above the table, there are four summary cards: 'On' (System Status), 'CS1503' (Cruise ID), '141.12 MB' (Cruise Size), and '2.54 TB' (Free Space). The table lists various data acquisition systems with their respective actions and enabled status. A 'Page Guide' section on the right provides instructions for using the 'Edit', 'Delete', 'Test', and 'Run' links. At the bottom of the table, there is a button to 'Add New Collection System Transfer'.

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

[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.



# OpenVDM 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: <b>Unlimited</b>	<a href="#">Edit</a>	Off
Data Filesize Limit for MD5 Checksum: <b>10 MB</b>	<a href="#">Edit</a>	Off

OpenVDM Specific Ship-to-Shore Transfers	Action	Enabled
Dashboard Data	<a href="#">Edit</a>	On
MD5 Summary	<a href="#">Edit</a>	On
Transfer Logs	<a href="#">Edit</a>	Off

OpenVDM Required Directories	Action
Dashboard Data	<a href="#">Edit</a>
Misc. cruise docs, pictures and data.	<a href="#">Edit</a>
Transfer Logs	<a href="#">Edit</a>

Data Warehouses	Action
Shipboard Data Warehouse (SBDW)	<a href="#">Edit / Test</a>
Shoreside Data Warehouse (SSDW)	<a href="#">Edit / Test</a>


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 `./server/plugins` 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 v3.8).
- The name of the plugin is strictly defined. It is the lowercase version of the short name for the collection system transfer followed by “\_plugin.py” (i.e. If a vessel has a collection system transfer named “SBE911” the name of the plugin must be “sbe911\_plugin.py”)
- Plugins are stored in the ./server/plugins directory.

In the repository, within the “./server/plugins directory” there are two plugin templates (openrvdas\_plugin.py.dist, em302\_plugin.py.dist).

These plugins are just templates 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 “OpenRVDAS” and “EM302”

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

```
<CollectionSystemName (lowercase)>_plugin.py [--datatype] [filename]
```

Calling the plugin with the --dataType command-line argument returns the dashboard data file's data type. Returning a data type (i.e. 'gga', 'svp', 'hdt', etc) lets OpenVDM know that there is a processor for specified filename. If this command returns nothing then OpenVDM assumed there is no processor for this specified filename and continues on.

The dashboard data files's data type is used by the OpenVDM Data Dashboard to properly place and visualized the dashboard data files. (i.e. draw the data as a time-series chart or as a trackline on a map).

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 openrvdas\_plugin.py...

The OpenRVDAS plugin creates the dashboard data files for lots of data types ranging from GPS positions to hull-mounted sound speed profilers. The `openrvdas_plugin.py` is written to handle 6 raw data types but can be expended to handle more as needed. In some cases the same processing steps will be run on files from multiple sensors (i.e. a vessel with 4 GPS sensors sending NMEA0183 GGA formatted data to OpenRVDAS). To make parsing all the various types the plugin pushes the processing of individual data files to data format specific parser scripts. Take a look at the "FILE\_TYPE\_FILTERS" array toward the beginning of the `openrvdas_plugin.py` file:

```
FILE_TYPE_FILTERS = [  
    {"data_type": "gga",      "regex": "*/NAV/POSMV-GGA_*.Raw",      "parser": "GGA", 'parser_options': {}},  
    {"data_type": "vtg",      "regex": "*/NAV/POSMV-VTG_*.Raw",      "parser": "VTG", 'parser_options': {}},  
    {"data_type": "tsg",      "regex": "*/METOC/TSG-RAW_*.Raw",      "parser": "TSG", 'parser_options': {}},  
    {"data_type": "twind",     "regex": "*/METOC/TWind-RAW_*.Raw",     "parser": "TWind", 'parser_options': {}},  
    {"data_type": "svp",       "regex": "*/METOC/Sound-Velocity_*.Raw", "parser": "SVP", 'parser_options': {}},  
    {"data_type": "met",       "regex": "*/METOC/MET-M01_*.Raw",       "parser": "Met", 'parser_options': {}},  
]
```

This array defines the dashboard `data_type`, the regex expression for data files of a single type, the appropriate parser for the file type and any custom arguments that should be passed to the parsers. When `openrvdas_plugin.py` is called with the `--dataType` flag it will compare the filename with the regex expressions and return the matching `data_type` field or nothing. When `openrvdas_plugin.py` is called without the `--dataType` flag, the plugin will pass the filename to the appropriate parser.

New file types can be easily added to the plugin by simply expanding this array. On the vessels with multiple GPS sensors, the GGA parser can be assigned to multiple objects in the `FILE_TYPE_FILTERS` array.

### Now let's take a look at one of the parsers: gga\_parser.py...

Parsers are located with the `./server/plugins/parsers` directory. By default there are now active parsers, all the supplied parsers include a `.dist` suffix that must first be removed.

Most of the supplied parsers utilize pandas and numpy. The `gga_parser.py` also requires the geopy library. All of the required libraries for each of the parsers can be determined by the import statements at the beginning of the file. All of the libraries used by all of the supplied parsers are included as part of the python virtual environment creating at part of the OpenVDM install.

The next dozen lines of the parser following the import statements define the constants used

through out the script.

- RAW\_COLS defines the columns in the raw data file
- PROC\_COLS defined the columns from the RAW\_COLS that the file is actually interested in
- ROUNDING optionally defines how the output data should be rounded.
- MAX\_VELOCITY defines the max velocity of the vessel. This number is used to detect GPS “jumps”
- MAX\_DELTA\_T defines the maximum interval that can exists between GPS fixes. Also used for error detection.

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

This exact file structure is used for all the CSV-formatted parsers. Creating a new parser is as simple as copy/pasting once of these existing parsers, tweaking the RAW\_COLS, and PROC\_COLS variables, defining QA tests, and updating the output labels and units.

The supplied parsers can also be called independent of a plugin to help develop and troubleshoot parsers and data processing issues. i.e. `./venv/bin/python ./server/plugins/parsers/gga_parser.py <gga_file>`

## 6 Customizing the Data Dashboard

The Data Dashboard layout is controlled via the `datadashboard.yaml` file located in the `./www/etc` directory. This yaml-formatted file instructs the OpenVDM web-application on how to setup the various tabs in the data dashboard, what data types to display on each tab, and how the data should be visualized.

By default OpenVDM includes the javascript files for visualizing dashboard data files as time-series graphs and geographic maps. There are also ways to define new visualization types and custom styles that can 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.



## OpenVDM User's Guide

```
# 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

# 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
```

## OpenVDM User's Guide

```
# 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
#
```