Debrief NG User Documentation

Ian Mayo

Debrief NG User Documentation

Ian Mayo

Publication date Autumn 2017

Contact details

Should you wish to learn more about Debrief please make contact the Project Manager:

Debrief Project Manager.

Mr Ian Mayo
Deep Blue C Technology Ltd
Fareham
HANTS
UK
e-mail: ian@DeepBlueC.com

Acknowledgements

Third Party Libraries. Debrief uses a licensed copy of the WMFWriter library from Piet Jonas (http://piet.jonas.com/WMFWriter/WMFWriter.html).

Open Source Libraries. Debrief also uses these Open Source libraries:

- XStream - http://xstream.codehaus.org/
- Apache Commons - http://commons.apache.org/
- JDOM - http://www.jdom.org/
- Java Topology Suite - http://tsusiatsoftware.net/jts/main.html
- JFreeChart - http://www.jfree.org/jfreechart/
- Michael Flanagans Scientific Library - http://www.ee.ucl.ac.uk/~mflanaga/java
- Date/Time support - http://www.joda.org/joda-time/
- Xuggle Media Player - http://www.xuggle.com/
- Watchmaker for Evolutionary Computation - http://watchmaker.uncommons.org/

Table of Contents

Introduction

1. Welcome

Welcome to the documentation for Debrief NG. Debrief was developed by the Maritime Warfare Centre in Portsmouth UK in the mid 90's, to support the analysis and reporting of maritime tactical exercises.

Tip

There's lots more Debrief goodness on the Debrief Home Page: http://www.debrief.info

In December 2000 the Maritime Warfare Centre decided to give Debrief an Open Source status, opening the application and its source code into the public domain. Debrief has been made Open Source to facilitate its wider use, encouraging adoption of standard file formats, presentation and practices between analysis agencies.

Debrief provides the following features:

- 2-Dimensional (top-down) view of vessel tracks
- Ability for user to *step-through* exercise serials
- Full formatting of data presented on screen
- Palette of tactical, vector map and chart-related features for insertion onto *plot*
- Export of plot images for insertion into word processor
- X-Y plotting of relationships between tracks
- 3-Dimensional view of tracks using height/depth data
- Gridded database of bathymetry/elevation
- Display of time-stamped text and sensor data

2. History

2.1. Prehistory

Debrief was originally produced in 1995 in the *Maritime Warfare Centre* to act as a desktop viewer for results produced by the MWC's ASSET submarine simulator. In use it quickly became apparent that real exercise data could also be viewed in the application removing the requirement for clerical staff to produce paper plots for use in analysis. The initial version of Debrief was a 16-bit MS Windows C++ application.

Debrief was updated in late 1996 to 32 bits, in order to exploit the richer user interface components available for 32 bit Windows applications. It was at this stage that the application was demonstrated and subsequently issued under license to COMSUBDEVRON 12 of the US Navy.

In 1999 development of an updated version of Debrief, Debrief 2000 was started. Over the previous four years a number of fresh requirements had arisen, requirements which could not be economically met using the existing architecture. Accordingly development of the Debrief 2000 application started from a fresh-whiteboard, adopting a modern modular approach to allow incremental implementation and insertion of future modules as they were required. The rapid maturity experienced by Java together with the availability of cheaply available development environments, rich application libraries (serialisation, Java3D and XML in particular) and its platform independence made Java the natural choice for the application.

During 2000 Debrief gained wider use within the Maritime Warfare Centre, being used for more varied types of submarine exercise analysis together with analysis in surface-related warfare areas.

2.2. The Switch to Open Source

In Summer 2000 the Maritime Warfare Centre committed to switching the Debrief application to Open Source status. The adoption of the Open Source licensing model affects MWC in a number of ways:

Debrief's Open Source benefits

1. Organisations that are currently using Debrief will have full access to the source code of Debrief, allowing them to identify and correct bugs (provided they have sufficiently trained staff). The licensing is such that these modifications can be again made public through the re-insertion into the central, online "code base".

2. Organisations that are not currently using Debrief also have full access to the application and its source code. Since the application and its supporting documentation clearly describe its origins in MWC this will spread the name of the organisation together with enforcing its reputation as a centre of maritime tactical analysis.

3. Any organisation using Debrief that identifies a bug/algorithmic problem is able to independently correct the problem and submit the corrected code back into the central "code base". In time, this will greatly increase the accuracy and reliability of the application. MWC may then freely utilise these improvements, only incurring the administrative overhead of "checking-in" code modified by third party organisations.

4. The free, open source status of the application makes it easier for third party, commercial organisations to bid for development contracts to maintain or extend Debrief. This wider availability will only bring economic advantages to MWC and fellow organisations.

5. The wider national/international use of Debrief will also lead to easier exchange of exercise data between nations (through common file formats) and potentially offer an increase in efficiency and the general quality of naval exercise analysis

2.3. Debrief 2001 onwards

At the end of 2000, Ian Mayo, the developer and project manager (see Table 1, “List of acknowledged Debrief users”) of Debrief, left full-time employment at the Maritime Warfare Centre to setup his own software development consultancy, PlanetMayo Ltd, now known as Deep Blue C Technology Ltd.

A competitive open tender process was conducted during late 2001 to supply the Maritime Warfare Centre with Debrief support. The contract was won by Deep Blue C, who grouped up the implementation of the MWC's fresh requirements in a major update to Debrief, titled Debrief 2001. This update bought new, large areas of functionality to Debrief including vectored chart data, display of narrative text, and display of sensor-data.

Debrief development continued in 2002, with the update to Debrief 2002, which bought greatly improved 3D plotting facilities and a number of incremental improvements to other areas of functionality.

The year 2002 also saw the first conference paper extolling the virtues of Debrief and the Open Source principles behind it, at Undersea Defence Technology 2002 in La Spezia, Italy. The paper itself is available for viewing at the Debrief web site [http://www.debrief.info].

Another significant step forward for Debrief in 2002 was the contribution from NUWC of an algorithm and code suitable for shifting tracks. NUWC developed a set of experimental classes used to perform track shifting (see Section 1.3, “Track shifting”). The algorithms used in these classes were taken and modified to complement the Debrief look and feel, and to provide greater usability resulting in the track-shifting editor included in Debrief 2002.

Through 2003-2007 Deep Blue C continued to provide MWC with contracted support for Debrief. This support included user guidance, bug fixes and addition of new features. Additionally, this support

contract was used to deliver Debrief 2003 (including the provision of a bathymetric bottom in 3D views, presentation of TMA data, and improved time-variable graphs), and the fundamental rebuild of Debrief into Debrief NG (providing a modular architecture ready to seamlessly accommodate ad-hoc analysis tools).

In 2009 MWC extended Debrief to support Single Sided Reconstruction, that is the derivation of a Red track from a Blue track plus sensor data. These extensions also included Track Grooming changes that support the removal of jumps and general data smoothing.

2011 saw the introduction of support for plotting chart backdrops in GeoTiff format. Unfortunately, this exposed performance shortcomings in the render cycle and has been temporarily removed pending streamlining / cacheing of the render pipeline.

Charting support recommenced in Spring 2012, and a suitable implementation was found (See Chapter B.9, *Using chart backdrops*). Support was also introduced for using Debrief in the process of Exercise Planning (See Chapter B.7, *Exercise planning*).

In 2015, under DSTL [https://www.gov.uk/government/organisations/defence-science-and-technology-laboratory] sponsorship, support for the Natural Earth dataset and sponsorship for the development of Semi Automatic Track Construction, was added to Debrief. Both are of enormous value, the former to the wide body of Debrief users (see Section 1, “Natural Earth data”), and the latter to specialists interested in this small but select niche (see Section 3, “Semi-Automated TMA generation”).

3. Debrief users

Whilst Debrief was originally developed for use within the Maritime Warfare Centre, the switch to Open Source licensing has led to its adoption by the following organisations and companies:

Table 1. List of acknowledged Debrief users

	Original Debrief project sponsor. Lt Cdr David Whitehouse RN Maritime Warfare Centre Marlborough Building, HMS Collingwood Fareham, HANTS PO14 1AS United Kingdom
	Debrief Project Manager. Mr Ian Mayo Deep Blue C Technology Ltd Fareham, HANTS United Kingdom <ian@DeepBlueC.com>
	COMSUBDEVRON 12 Submarine Development Squadron Twelve Naval Submarine Base New London Groton, CT, 06349-5200 USA

Naval Undersea Warfare Center (NUWC)

Roger Howlett
Code 2212 Combat Systems
ICE Development Team Naval Undersea Warfare Center
Bldg 1171 CSTL Laboratory
Newport, Rhode Island, USA
<HowletRW@npt.nuwc.navy.mil>

Joe Sileo
COMSUBPAC Tactical Analysis Group (N7231)
Commander Submarine Force, U.S. Pacific Fleet
1430 Morton St, Bldg 619
Pearl Harbor
HI 96860-4664

Jay Spry
COMSEVENTHFLT/COMSUBGRU 7

If your organisation or company uses Debrief and wishes to be included as an acknowledged user please forward your details to the Debrief Project Manager as recorded above.

4. Document conventions

This document uses the following conventions

Table 2. Document conventions

Descriptions	Appearance
File Names	`file.extension`
Label of a screen item	View Toolbar
On-screen button	Auto Generate
Name of an application	Internet Explorer
Emphasized text	*word*
An entry in the glossary (click to view)	*Plot*
Source Example	`<para>Beginning and end of paragraph</para>`

Part A. Getting started

As the title suggests, this part of the tutorial will lead you through *Getting Started* with Debrief right from the installation through to more advanced topics such as creating custom sets of layers for annotation fresh plots.

If this is your first time using the User Guide and you haven't installed Debrief yet, then we would recommend reading the chapters and sections in the order that they appear. This way you know what you need to do before installing Debrief, how to perform the installation, configure the data, etc.

Also, don't forget that we have a series of tutorials and videos to help you get up to speed quickly. These don't take long to read or watch, and are probably the quickest way of getting to grips and up to speed with Debrief; you can follow these at your own pace.

Chapter A.1. First steps

In this first chapter, we will look at getting you up and running with Debrief. There are some tasks that you need to do before installing Debrief and, to ensure the whole process is as easy as possible, this chapter is grouped into the following sections:

1. Before installing Debrief (Section 1, “Before Installing Debrief”)
2. Performing installation (Section 2, “Performing installation”)
3. Configuring reference data (Section 3, “Configuring reference data”)
4. Start Debrief (Section 4, “Start Debrief”)
5. Importing track data (Section 5, “Importing track data”)
6. Moving around the view.

   Once you've completed these, you can then move onto Part B - Using Debrief (Part B, “Using Debrief”).

1. Before Installing Debrief

In this section, we will look at obtaining Debrief, via download or CD-Rom, and where you can get help on Debrief. Then, we'll look at installing Debrief.

1.1. Obtaining Debrief

A number of project support activities (news, feature-requests, bug-reports) for Debrief are conducted online at *GitHub* (https://github.com/debrief/debrief).

In addition, you can also go to this URL to obtain the latest version, examine the list of available downloads, and the relevant zip-file for your operating system: https://github.com/debrief/debrief/releases.

1.2. Where to get help on Debrief

If you get stuck with Debrief, the following sources of information are available:

1. **This document.** The Debrief NG User guide is a useful reference for determining where to find information on Debrief, as well as the overall capabilities of the tool, and assorted reference guidance. It's also accessed and presented online via web browser help from within Debrief NG (accessed via Help Contents on the Help menu).
2. **Tutorials.** A series of interactive tutorials are distributed with Debrief. These guide users through a series of complex tasks to achieve overall goals such as *loading data into Debrief* or *analysing narrative data*. The tutorials are accessed by selecting Debrief Tutorials from the Help menu in Debrief NG. A PDF version of the tutorials (suitable for printing) is in the Debrief installation folder.
3. **Welcome Page.** Debrief NG's welcome page provides a high level overview of Debrief and guides the user to further sources of information, tutorials, and samples.

1.3. Debrief on CD-Rom

Note

If you are unable to download Debrief from the Internet, please contact the Project Manager (see Table 1, “List of acknowledged Debrief users”), and a copy of Debrief on CD-ROM will be forwarded to you.

2. Performing installation

Warning

If you already have Debrief NG installed on your machine you should remove it using the Add/Remove Programs button from the Settings button on the Windows Start menu.

Once you've done this, just unzip the downloaded application zip file, and you're ready to go! Debrief NG needs no installer.

Tip

You don't have to but if you wish to follow MS Windows conventions you can copy the unzipped folder into your Programs folder.

Then, to start the application just double-click on the *Debrief NG* executable in the top level of the unpacked folder.

3. Configuring reference data

If you want to take advantage of Debrief's support for background datasets, don't forget to read the instructions on how to configure their locations (shown below). Note, due to their size, the datasets probably aren't present in your Debrief installation, so you will have to request and then load them separately. In 2015 Debrief added support for the Natural Earth dataset, superceding the VPF and ETOPO datasets - so you probably won't need to configure them.

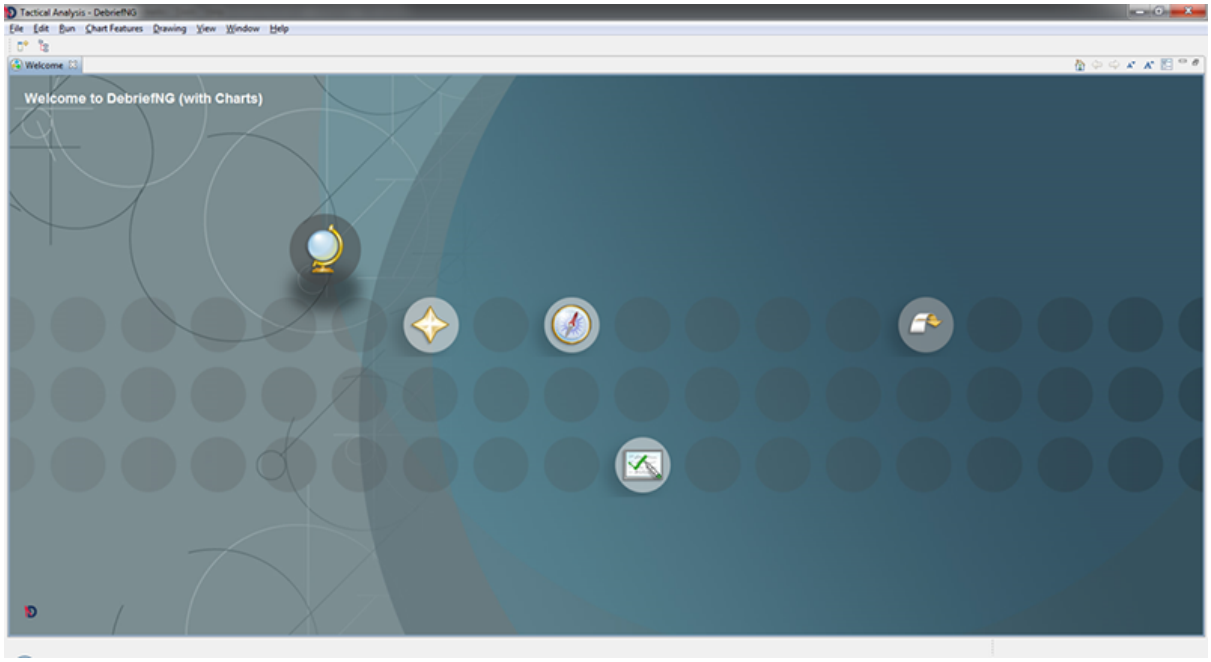
- Natural Earth instructions are in Section 1.2, “Using Natural Earth”
- VPF instructions are in Section 2.5, “Configuring Debrief to read VPF data”
- ETOPO instructions are in Section 5.1, “Configuring Debrief to read ETOPO data”

4. Start Debrief

4.1. Opening the application

Open the application by double-clicking on the executable in your Debrief folder. It may take a couple of seconds to load, but don't worry about this as, once the windows are open, Debrief runs quite quickly. If you are viewing this tutorial online, arrange this window and the Debrief window as best you can, so that both are visible (although you may need to let them overlap if you have limited screen space).

Figure A.1.1. Debrief's Welcome view



When Debrief is opened for the very first time you will see the Welcome page (shown above). This page includes a series of links to high-level introductory information. Though we do have a quick summary of each below, we do recommend that you go through each of these links and familiarise yourself with what's there.

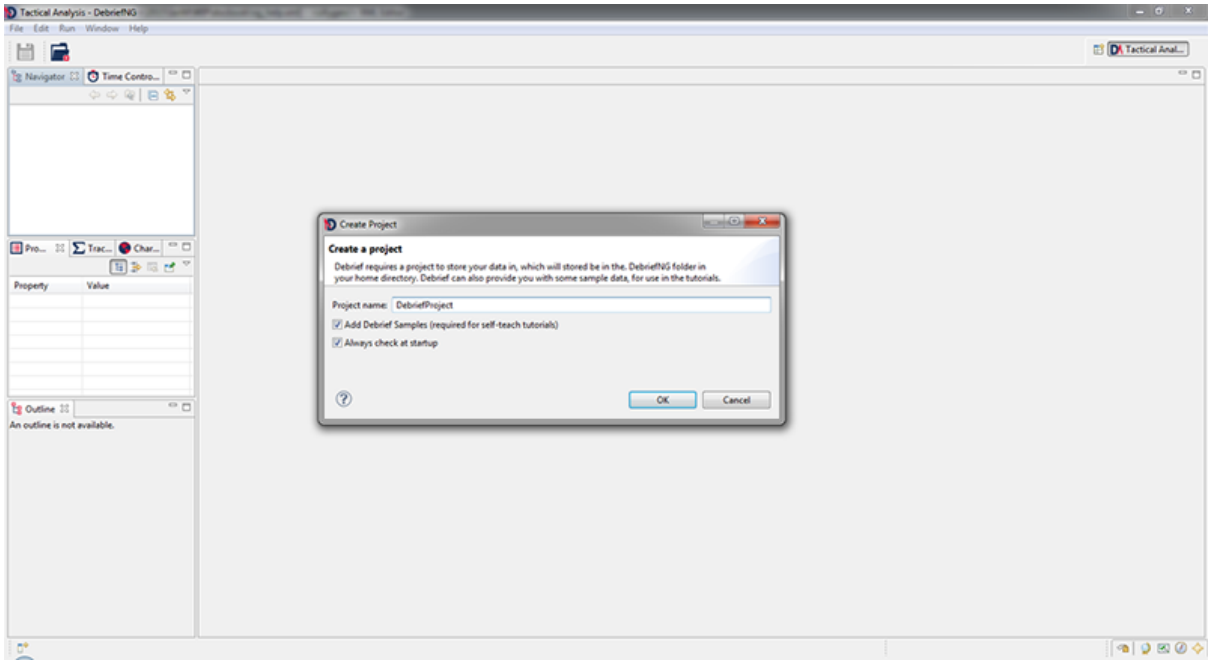
The Overview page contains guidance for new-adopters of Debrief, whereas the Tutorials page leads users through more specific tasks through the use of *tutorials* - small pages breaking a task down into a series of steps. The buttons at the top-left of each page navigate you around the welcome guidance. On the Tutorials page one tutorial in particular will help new Debrief-adopters.

4.2. A Debrief Walkthrough

It's a good idea to let Debrief itself lead you through your first session

Starting any of the tutorials will shrink the Welcome guidance to a smaller pane exposing the rest of Debrief NG as shown in the following diagram.

Figure A.1.2. Debrief's default view



Debrief requires you to have a working folder to store your data in (a project). As a convenience, when you open Debrief it will check if there are any projects present. If none are present, the Debrief New Project Wizard will invite you to create a project folder. Debrief will also offer to copy in some sample data - it's a good idea to do this.

The Debrief user-interface follows the Workbench convention, whereby a users edits individual files in an *Editor* assisted by a series of supporting information panes called *Views*. The specific arrangement of editors and views offered by Default is called a *Perspective* - in this opening instance it will be the *Tactical Analysis* perspective.

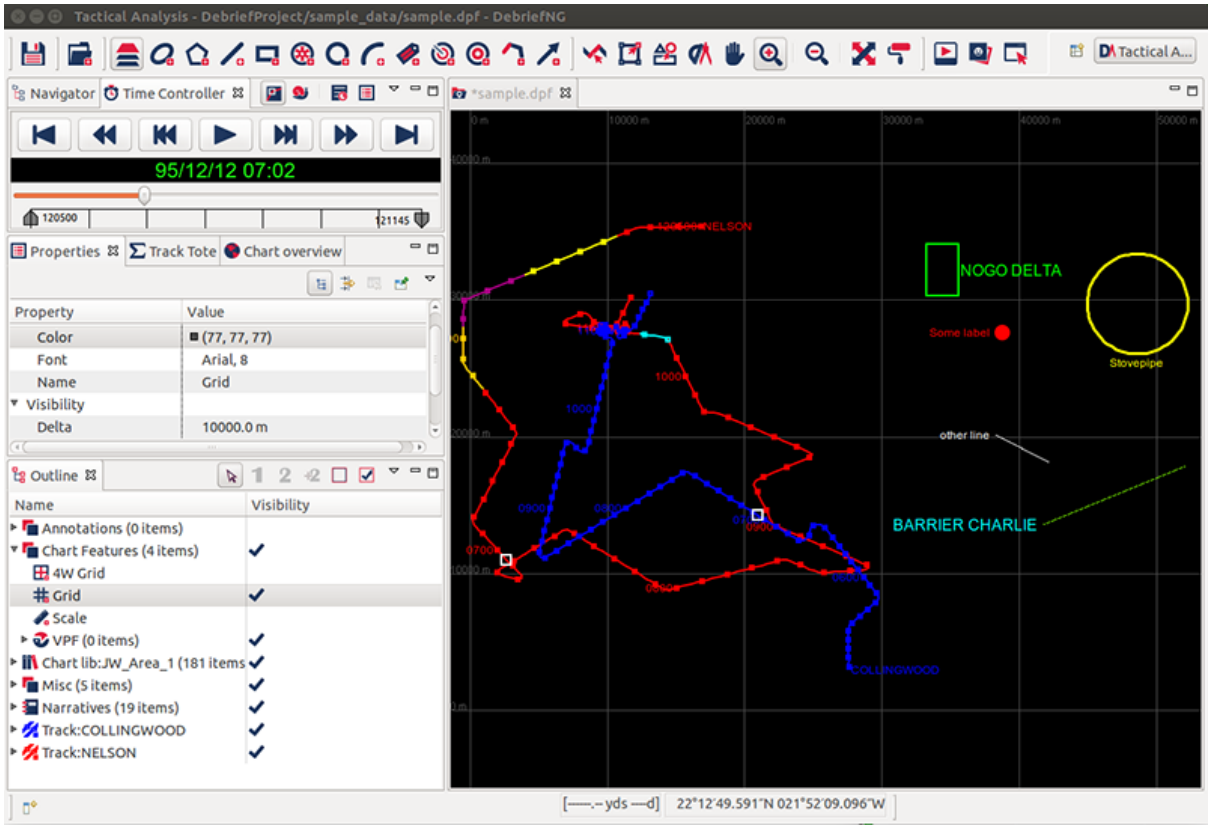
Though views can either be dragged to any other location on the workbench or floated above the workbench, the Tactical Analysis perspective arranges the views into sets of grouped tabs. Views frequently have their own toolbar and drop-down menu which provides actions related to that particular task.

Once selected, views are closed by clicking on the white diagonally-oriented cross. To re-open a view or open a fresh view select Show View from the Window menu; a list of Debrief-related views will then be shown. Other views can be selected from the Other... menu item.

4.3. Navigating around Debrief

Fundamentally, the Debrief user interface (generically termed the Workbench) is constructed from a plot editor, surrounded by a series of support panes (called views), both underpinned by a main menu and a series of toolbars. These elements are described further below.

Figure A.1.3. A Debrief plot



4.3.1. Editors

Any number of editors can be open at once but, as is common with many applications, only one can be active at a time. The main menu bar and toolbar for Debrief contain operations that are applicable to the active editor. Initially the active editor will always be the Debrief plot, but the extensibility of Debrief NG means other types of editor can be implemented. Tabs at the top of the editor area indicate the names of files that are currently open for editing (`Sample.dpf` in the previous screenshot). An asterisk (*) in the tab indicates that an editor has unsaved changes. By default, editors are stacked in the editor area, but you can easily tile them by dragging and dropping the tabbed plot-name to another location in the window; this allows you to view source files simultaneously.

4.3.2. Views

Views support editors and provide alternative presentations as well as ways to navigate the information in your Workbench. For example, the Navigator displays projects and other resources that you are working with (see Section 4.3, “Navigating around Debrief”), and the Outline View shows a list of selectable items within the current plot (see Section 4, “Layer management”).

Views also have their own drop-down menus. To open the menu for a view, click the icon at the right end of the view's title bar. Some views also have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view.

A view might appear by itself, or stacked with other views in a tabbed notebook. You can change the layout of a perspective by opening and closing views as well as by docking them in different positions in the Workbench window. We've attempted to provide a logical arrangement of Debrief views, but feel free to experiment. Debrief will remember your settings, but if at any point you wish to return to the presets select Reset Perspective from the Window menu.

4.3.3. Toolbars

There are four kinds of toolbars in Debrief.

The **main toolbar**, sometimes called the Workbench toolbar, is displayed at the top of the Workbench window directly beneath the menu bar. Items in the toolbar might be enabled or disabled based on the state of either the active view or editor. Sections of the main toolbar can be rearranged by dragging and droppign with the mouse.

There are also **individual view toolbars**, these appear in the title bar of a view. It's important to note that actions in a view's toolbar apply only to the view in which they appear. Some view toolbars also include a Menu button, shown as an inverted triangle, that contain actions for that view.

Whilst Debrief doesn't currently make use of Perspectives (refer to the Perspectives note, below), a third type of toolbar is **the perspective switcher**. The perspective switcher allows quick access to perspectives that are currently open. It also has a button that can open new perspectives. The perspective switcher is normally located in the top-right, next to the main toolbar. However, it is also possible to position it below the main toolbar ("top-left"), or to position it vertically on the left-hand side of the workbench ("left"). The name of the perspective is shown by default, but it is possible to hide the text and show only the icons. To reposition the perspective or hide the text, right-click on it and choose the appropriate item from the context menu. When Debrief contains modules for Track Reconstruction it would be understandable for a Track Reconstruction perspective to provide a suitably tailored set of views.

Finally, **the fast view bar** is a toolbar that contains icons representing the current set of fast views. A fast view is a shortcut to a view that is frequently used; it is generated by right-clicking in a view's title bar. The fast view bar appears in the bottom left corner of the workbench by default. However, it is possible to position it on the left or right as well. In all cases, you can find out what toolbar buttons do by moving your mouse pointer over the button and reading the tooltip that opens.

Note

Perspectives. Each Workbench window contains one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors as well as providing a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. For example, the Tactical Analysis perspective combines views that you would commonly use while analysing and editing tactical exercises. Perspectives control what appears in certain menus and toolbars and define visible action sets that you can change to customize a perspective. Debrief allows you to save customized perspectives. These can then be loaded in the same or a new window via the General > Perspectives preference page.

4.4. Loading your first plot

Refer to the Debrief tutorials (Tutorials) for guidance in creating either a new Debrief plot, or in loading an existing plot.

5. Importing track data

5.1. Introduction to Replay files

One of the file formats that Debrief uses is the Replay format (**.rep**). These are ASCII files that contain vessel position date in a flat-file format. (*Replay* was an application used in Debrief-style tasks in Royal Navy analysis in past years.)

Tip

Debrief can also load data stored in the increasingly popular KML format. This is covered in the Section 3.1, “Introduction to KML file format”

A sample rep file is below:

Example A.1.1. Sample Replay file contents

```
951212 050000.000 NELSON @C 12 11 10.63 N 11 41 52.37 W 269.7 2.0 0
951212 050100.000 NELSON @C 12 11 10.58 N 11 42 2.98 W 269.7 2.0 0
;NARRATIVE: 951212 095700.000 COLLINGWOOD SUSPECTED DETECTION OF RED
951212 050200.000 NELSON @C 12 11 10.51 N 11 42 14.81 W 269.9 2.0 0
951212 050300.000 NELSON @C 12 11 10.51 N 11 42 27.27 W 268.7 2.0 0
951212 050400.000 NELSON @C 12 11 10.28 N 11 42 40.33 W 270.6 2.0 0
```

Note

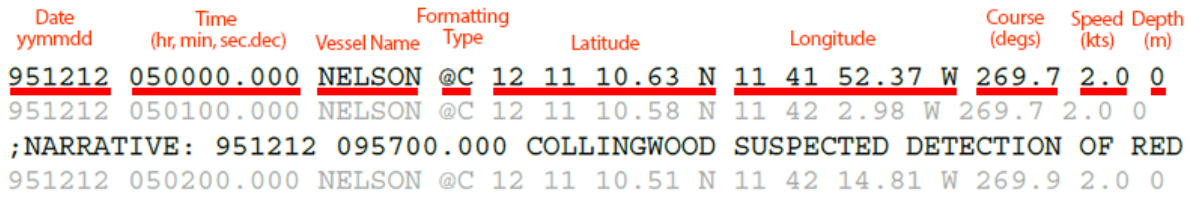
Since Autumn 2004 multi-word track names can now be read in if they are surrounded by quotation marks (see Section 1.1, “Track data”).

This data has a single vessel location per-line, with white-space separated columns containing this data:

- Date (year, month, day)
- Time (hours, minutes, seconds, decimal seconds)
- Vessel Name (single-word)
- Formatting to apply (see the maintainer documentation in Part C, “Maintainer's Guide” for details of this, but experiment with the second character to change the default colour of the *track*).)
- Latitude (deg, min, sec, hemisphere)
- Longitude (deg, min, sec, hemisphere)
- Course (degrees)
- Speed (knots)
- Depth (m)

To further illustrate this, refer to this image:

Figure A.1.4. Detailed Sample Replay file contents



Note the third line, which contains a *narrative* entry. This entry represents one of a group of types of data called *Annotation* entries. These entries allow inclusion of data other than vessel locations, and can be interspersed with positional data throughout the file.

Multiple vessel tracks can be contained sequentially in a single Replay file.

For more detail regarding this file format, including how to represent annotations which are only visible for a defined time period, together with lines, ellipses and rectangles, refer to the maintainers section of the this document (Part C, “Maintainer's Guide”).

5.2. The Debrief file format

Note that the Replay file format just stores sensor readings - it does not store any formatting details. As a result, any formatting applied to a plot cannot be saved as a Replay file. Thus, Debrief has

the *Debrief File Format*: This is a more complex file format that stores a wide range of data-types, formatting instructions, and user-interface settings (such as the current viewport on the data).

Thus, whilst you may load data from a *.REP* Replay file, any modifications made to the plot must be stored in a Debrief Plot File (*.dpf*). No changes are save to REP files by the Debrief plot editor (though it possible to make textual changes to REP files using Debrief's built-in text editor). DPF files use the XML storage format which is capable of storing structured information in predefined types, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <plot Created="Tue Nov 18 09:54:38 GMT 2008" Name="Debrief Plot">
     <details Text="Saved with Debrief version dated Tue Nov 18
 09:51:52 GMT 2008"/>
and so on....
```

5.3. Importing data

If this is the first time of doing this, refer to the tutorial covering importing existing data into Debrief (Track).

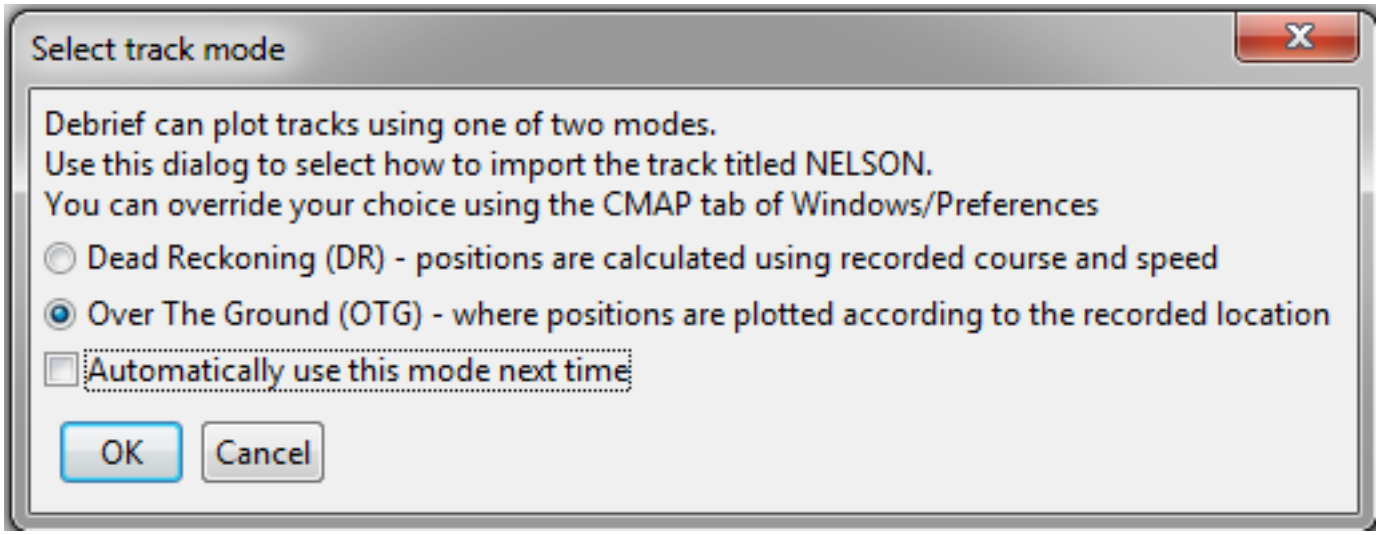
When data is imported, Debrief applies default values to the time labels for positions. The first point in a *track*, together with the first point every day, includes the day, otherwise just hours and minutes are displayed.

We do realise that you haven't learnt how to show the labels on a track yet - that's still to follow; but remember, when you switch on the labels for a *track*, the default labels are set at the import stage.

5.4. Import modes

Since the 2009 introduction of extensions to support Single Sided Reconstruction (see Section 2.3, “Debrief 2001 onwards”), Debrief is able to import data in either *DR* or *OTG* mode. The mode to be used is decided at the REP import stage via the Select track mode dialog:

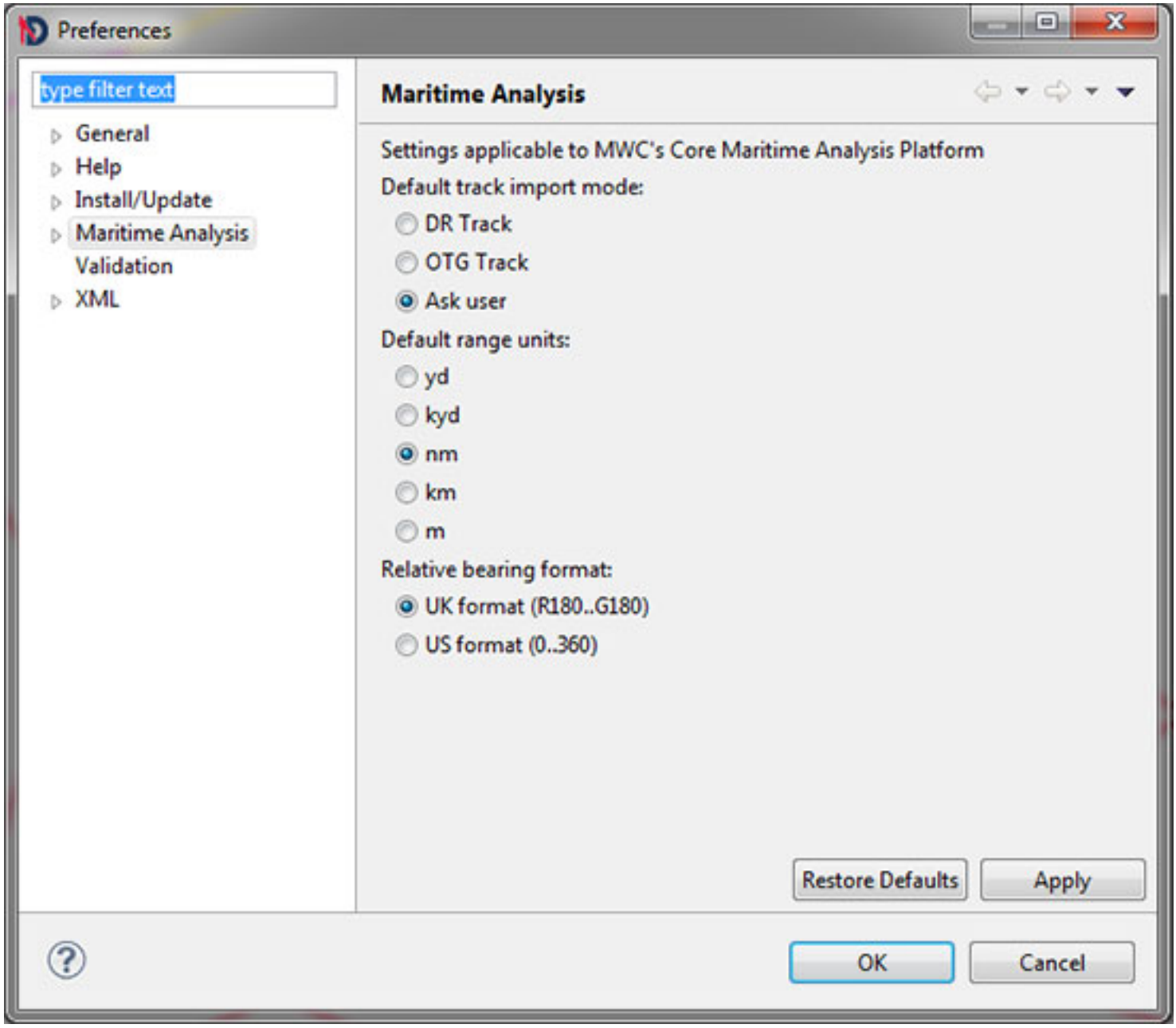
Figure A.1.5. Select Track Mode



Once imported, it is not possible to switch between the two track modes. As such, if you need to switch modes, delete the track, re-import it, and then select the track mode required.

If you want to set a different default import mode, then use the Maritime Analysis Preferences dialog located in Windows > Preferences:

Figure A.1.6. Debrief's Preferences



Part B. Using Debrief

You're now ready to start using the more advanced features of DebriefNG, the following chapters will guide you through specific areas of Debrief's analysis capability. More specifically, these are:

- Manipulating track data (Chapter B.2, *Manipulating track data*)
- Analysing data (Chapter B.3, *Analysing Data*)
- Exporting data (Chapter B.4, *Exporting Data*)
- Symbol sets (Chapter B.5, *Symbol sets*)
- External datasets (Chapter B.6, *External datasets*)
- Exercise planning (Chapter B.7, *Exercise planning*)
- Viewing narratives (Chapter B.8, *Viewing narratives*)
- Using chart backdrops (Chapter B.9, *Using chart backdrops*)
- Analysing sensor data (Section 2, “Analysing your data”)
- Management of TMA and TUA solutions (Chapter B.11, *Management of TMA and TUA solutions*)

Chapter B.1. Moving around the view

1. Introduction

Once data is loaded into Debrief, there are a number of ways of controlling the view of the track data, but the most efficient way is by using the buttons on the View toolbar:

Figure B.1.1. View toolbar

Tip

For most of Debrief's operations (such as adding drawing features) it needs to have an area to work with. When re-opening an existing plot file, or importing data, Debrief determines the area covered by the plot from the given data. However, when starting a new, blank session, Debrief doesn't know what area to use and defaults to its own origin at Fort Blockhouse, HMS Dolphin, Portsmouth, United Kingdom: the plot will be centred on 50 degrees 49 minutes North, 1 degree 19 minutes West (approx).

2. Mouse mode buttons

The first five buttons on the View toolbar represent modes of use and are selectable individually with the limitation that only one button at a time can be selected. When clicked, the previous selection is deactivated, and this button becomes active.

 Drag Track Segment

Drag Track Segment allows you to drag a single segment of track. It is of particular relevance when manipulating track segments in Single Sided Reconstruction (see Section 1.3, “Track shifting”).

 Drag Component

Drag Component lets you drag a single point within a large feature. For example, after switching to Drag Component mode you can drag the corners of a rectangle or a single point within a vessel track.

 Drag Whole Feature

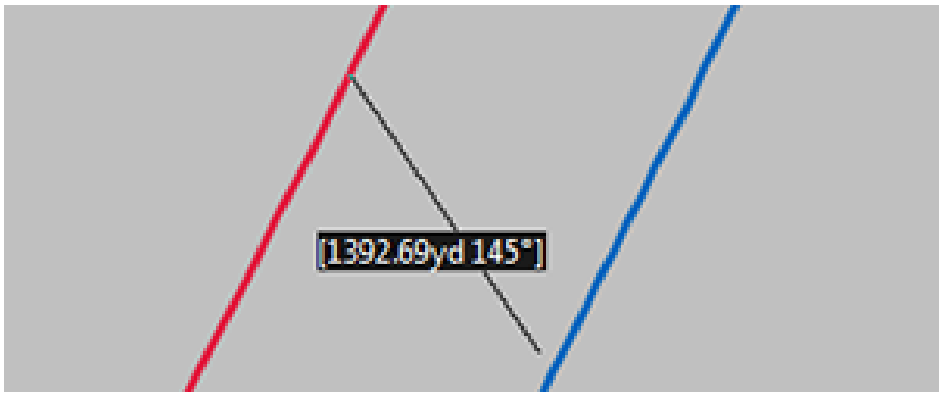
This setting lets you drag a complete feature. In Drag Whole Feature mode you can drag a rectangle around the plot, or move a complete vessel track (see Section 1.3, “Track shifting”).

 Pan

The Pan control lets you move your current viewpoint. Click on the Pan button, and then drag the mouse around the *plot* -- you will see the view change as you do it.

 Range Bearing

The Range and Bearing control neither allows you to move around the view nor move objects, but selecting it allows you to measure the precise range and bearing between two points on the plot. As you click and drag, you will see the calculated results displayed at the mid-point of the connection, as shown below. The default units are yards, but different units may be selected from the CMAP section of the properties shown in the Window/Preferences dialog.




Zoom in

You are in the Zoom In mode by default. So, click and drag the mouse across the area where the two *track*s are located near the centre of the *plot*. When you release the mouse button you will see a zoomed in view of the data.

Tip

In addition to the mouse mode buttons described above, the middle mouse button may be used to navigate irrespective of the current mouse mode. Hold the middle mouse button down and drag the cursor to pan around the plot, or hold down the CTRL key and roll your mouse wheel backwards and forwards to zoom in and out of the plot.

Tip

Handy tip no. 2 is when the plot is selected you can use the tab key to cycle through the toolbar drag modes.

3. Click buttons

The remaining buttons on the View toolbar differ from the previous ones in that they don't involve mouse interaction with the plot, and selecting one of these doesn't deselect the already selected mouse mode button.


Zoom out

If you click on the Zoom out button, you will "zoom out" of the *plot* (surprise, surprise). However, as mentioned in the brief introduction (above), if the Zoom in button is already selected, selecting Zoom out doesn't deselect it - they're designed to work in conjunction with each other. You click on Zoom out as many times as is required and, as soon as you stop, you can then drag an area to zoom in on and work. If you have a wheel-mouse, holding the Ctrl key down allows you to zoom in or out, as described above in *Toggle Buttons*.


Fit to Window

Clicking on Fit to Window will display all currently visible plot data within the confines of the *plot* area. This is good if you're zoomed in too far and need to see all the data, a single click will zoom out so you can see all visible information; conversely, if you've zoomed out too far, and the plot shapes are mere pin-pricks, a single click will zoom in, fill your plot, and allow you to see exactly what you have displayed. Note, this operation only applies to visible data, not that there may be data on the plot which is not visible (hidden layers). In addition, this phenomenon relates to the data once you have pressed "Filter to time period" from the Time Controller - the

plot is resized to fit the data contained in the indicated time period (you'll learn about this later in Section 2.8.1, “Filter to time period”).

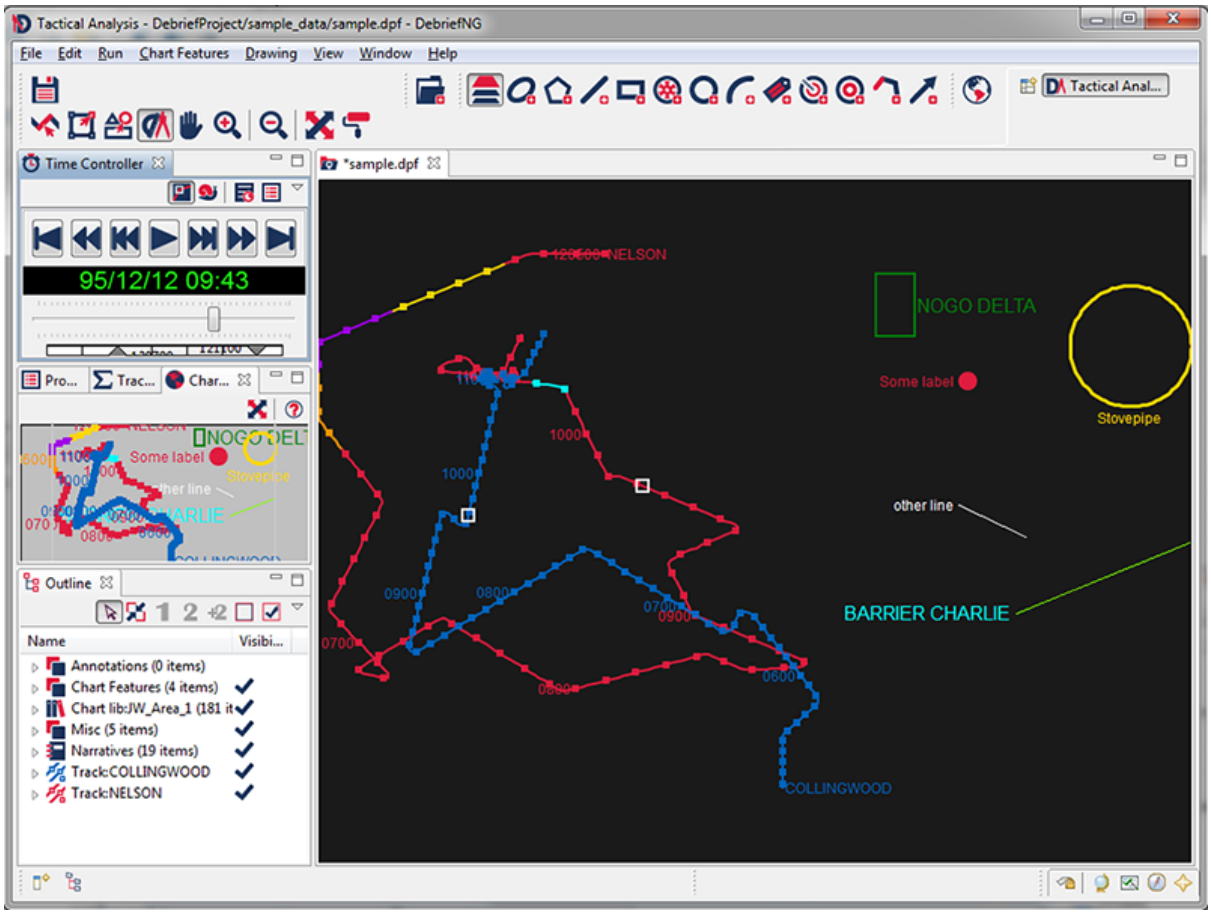

Refresh View

This button refreshes the current view. Refreshing is occasionally required when switching between display modes.

4. Chart Overview

The Chart Overview view provides an large scale view of the entire exercise area. This view is opened from the Window/Show View... drop down list (if you can't see it in the drop-down list, click on 'Other' at the bottom of the Show View list, and a dialog will open where you can select from unlisted views). Using it is quite simple: drag a region in the overview and the current plot will quickly zoom to show that area. A highlight rectangle shows the current plot view. Once you've zoomed in a little you can double-click in the overview to recenter the main plot on that point.

Figure B.1.2. Overview chart in use



5. Undo

Debrief contains an unending list of undoable operations which are only cleared when the current session is closed. Most changes to the Debrief plot are undoable, whether they be creating a new feature, changing an attribute of a feature (such as color), changing the current perspective, etc. If something doesn't undo, then the changes are it's something we've overlooked, so please let us know about it using the procedure in Section 5.1, “Reporting bugs” and we'll correct it as soon as possible.

Each open Debrief plot has its own undo buffer, so if the undo button is disabled on a particular plot you must first make sure it's active (single click on it to do that). Furthermore, even though you may

have made the edit via the Outline View, you must still click on the plot before conducting an undo operation.

6. Formatting the plot

The Debrief plot itself can be customised in a number of ways. For example, the background color can be changed to suit different projection/printing mediums, or just to suit the analysts' personal choice. Another way is by changing the line widths of tactical objects which are displayed. Either of these options are easily accessible by right-clicking on the plot and selecting *Edit base chart*. A properties panel will then appear, and you can use the drop-down list to change the background colour, move the slider to change the line width itself, etc.

Now we know how to move around the view, use the mouse buttons, and format the plot, we can now move onto manipulating track data.

Chapter B.2. Manipulating track data

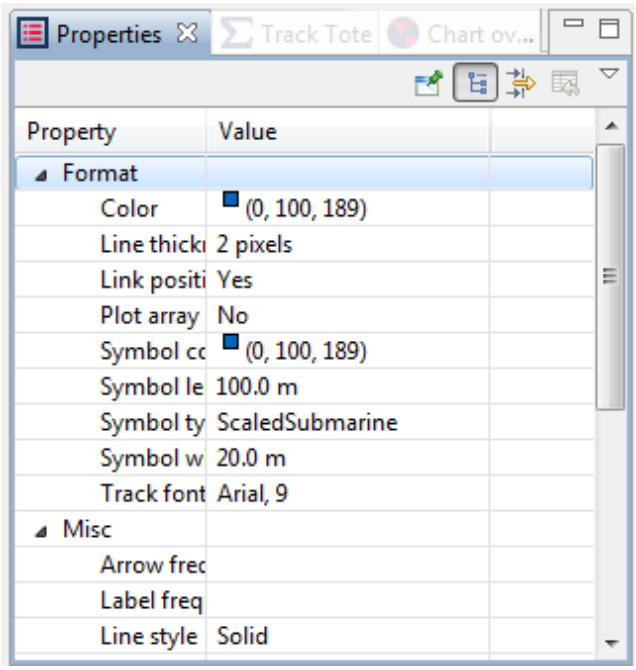
Before you are able to analyse or export plots from your data you need to know how to format what you see to improve its legibility. Learn more about formatting tracks in the *Editing your data* tutorial.

Note

Apologies in advance to the British users of Debrief for the American-spelling of colour. The software environment within which Debrief is developed is American, and in this case superimposing the UK spelling upon it is not a battle worth fighting.

1. Property editing

Figure B.2.1. Properties view



1.1. Edit track

Track editing is performed on the Properties Window (see Properties window), and is initiated by selecting the track either from the plot or the Outline View, as described in the *Editing your data* tutorial.

Note

Many of these editing operations can be performed without using the Properties Window, simply by right-clicking on a track either in the Outline View or on the plot

1.2. Set label, symbol and arrow display intervals

In addition to switching the symbols, labels and arrows on and off for individual positions, you can instruct Debrief to show symbols or labels at one of a series of frequencies - allowing you to quickly add 15 minute symbols and hourly time labels to a track. The timings of the particular positions that get highlighted do not have their origin at the start of the track - but from 0000 hrs - thus the 15 minute positions will be marked for 00, 15, 30, 45 minutes past the hour.

Note

Where a position on a track is marked as having its arrow and symbol visible, the arrow takes precedence - so only one will be displayed. In this way can request 5 mins symbols, and 30 min arrows: this will give you 5 min symbols, except on the 1/2 hour, when an arrow will be shown.

1.3. Track shifting

The track-shifting function has been incorporated to allow users to move a track and its associated sensor data. The principle requirement for this functionality is to enable the user to move the track to a specific geographic reference point (i.e. GPS fix) or to lock the relative position of one track to another using sensor data. This positioning relative to another track is sometimes termed plot-lock or tie-point .

Note

The Track-shifting implementation was originally provided by Mr David Gong at NUWC.

Debrief NG provides three modes of track shifting - moving individual track segments, moving individual points in a track to correct a potentially erroneous data point, moving whole tracks to overcome navigational system inaccuracies - such as during a plot-lock or tie-in. See the three modes above in Section 2, “Mouse mode buttons”.

A Stacked Dots view is used to show bearing-errors when the track-shift is being conducted to line up one or two sets of sensor data (see Chapter B.10, *Analysing sensor data*). For each sensor bearing on a track (see Sensor contact), Debrief identifies the point on the opposing track nearest to that time. Each time the track is shifted (dragged) Debrief calculates the error between the sensor bearing and the current measured bearing, showing these in a plot (bearing error against time).

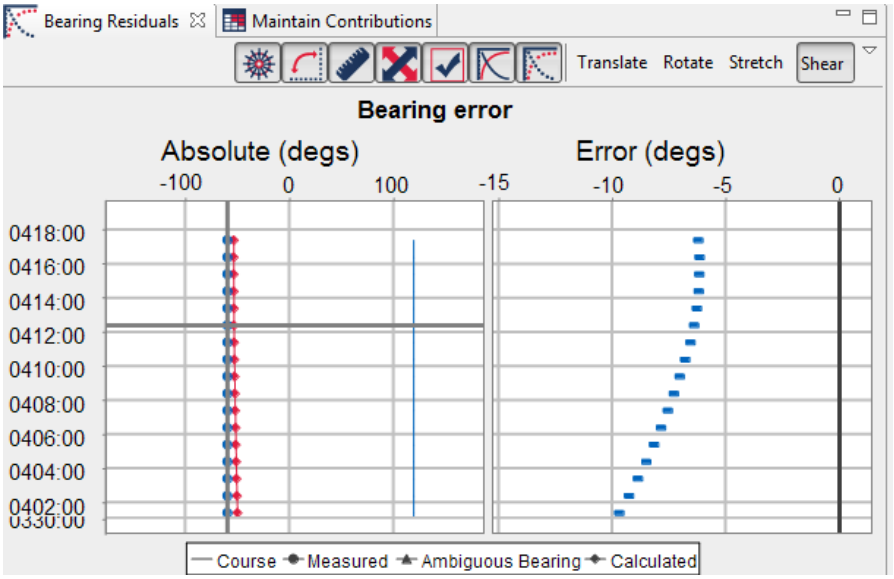
Note

A Sensor Offset measurement (see Section 1.4, “Sensor offset lengths editor”) is applied to the sensor bearing line where there sensor datum is offset from the attack datum of the platform

Debrief displays a symbol for each visible sensor data, with the symbol being plotted the same colour as the sensor data. When track-shifting a long exercise serial, the time-variable plot may become difficult to use due to the y-axis always auto-scaling to accommodate all available data. Overcome this by instructing Debrief to filter-to-time-period (see Section 2.8, “Manipulating according to time period”) from the time controller (Section 2.1, “The Time Controller”) for the plot-lock period, thus viewing a much smaller data-set in the time-variable plot. Additionally, you can zoom in on a particular area (by dragging an area downward and to the right) of the time-variable plot for detailed analysis, dragging the mouse up and to the left to clear the zoom.

Note the four modes of drag supported by the bearing error view. For the non-translate operations you are still able to perform a translate by picking up the track segment at its mid-way mark.

Figure B.2.2. Stacked Dots view



For the track-shifting to work, your data must be configured as follows:

1. You must have a Debrief plot open (duh)

2. You must have a vessel track as the primary track on the Tote (see Section 1.1, “Tote area”) - not an annotation or other time-related object

3. You must have a secondary track on the Tote, but only one secondary track. This must also be a vehicle track.

4. The primary track must have sensor data (see Chapter B.10, *Analysing sensor data*).

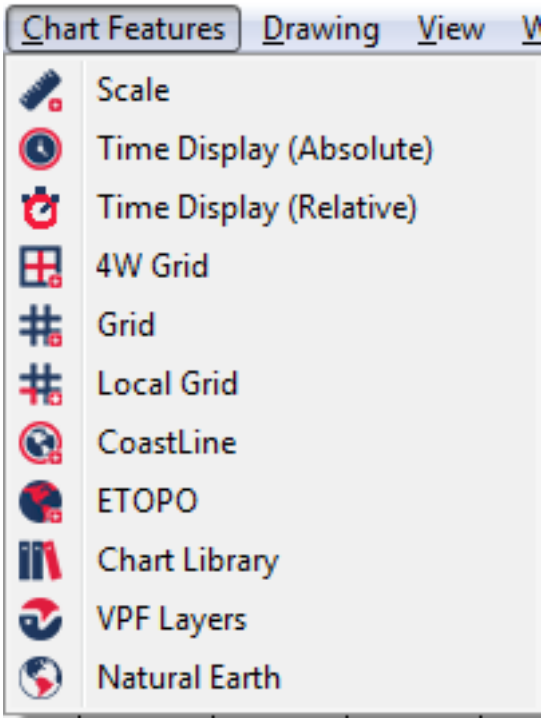
2. Adding chart features

2.1. Chart features toolbar

Debrief also allows you to add items to the *plot*.

These items are contained in two menus; Chart Features and Drawing. In Debrief, hover the mouse over them to see what type of item they create.

Figure B.2.3. Chart features menu



Important

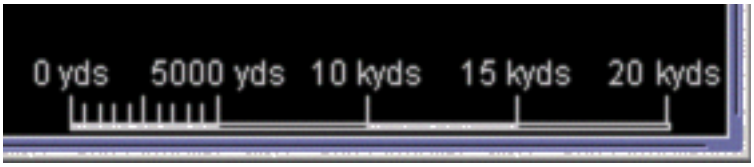
It is important to note that each time you click on an item from the toolbar, a new instance of it is created, it does not re-open an existing item.

2.2. Scale

The Scale button provides a scale, indicating to the viewer the current area of coverage of the plot. Once created the scale values can be set automatically or manually, as described below:

Auto Mode	In auto-mode Debrief assesses the current screen size and area of data covered, and attempts to set the most appropriate range of values and step size for the scale. A good working practice is to switch to auto-mode to allow Debrief to estimate the optimal values, then switch out of auto-mode to fine-tune the ScaleMax and ScaleStep values provided.
Color	The colour used to draw the scale.
Location	The corner of the plot where the scale is placed.
ScaleMax	The maximum value of the scale (in yards)
ScaleStep	The size of the steps used to break up the scale (again in yards)
Visible	You can clear the visibility flag to temporarily hide a scale, allowing you to switch between scales, for example.
Units	Use this list to select the units displayed in the scale

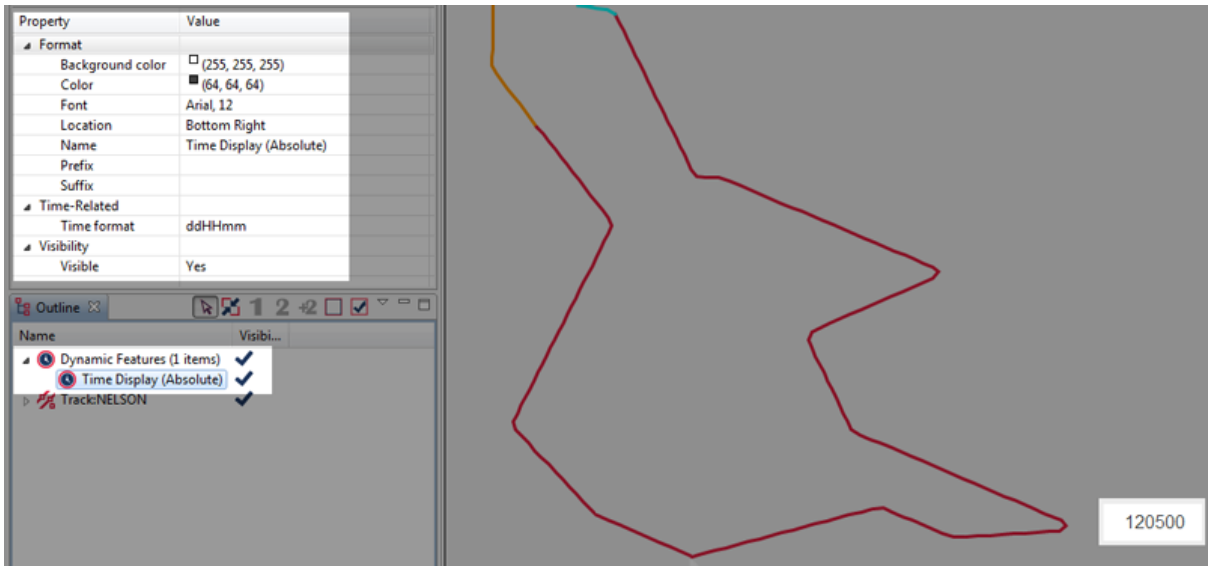
Figure B.2.4. Sample scale



2.3. Time Display (Absolute)

Debrief offers 2 types of time display within the *plot editor*: absolute and relative. We will cover Section 2.4, “Time Display (Relative)” in the following section. The *Time Display (Absolute)* button opens up a new layer within the *Outline view*:

Figure B.2.5. Time Display (Absolute)



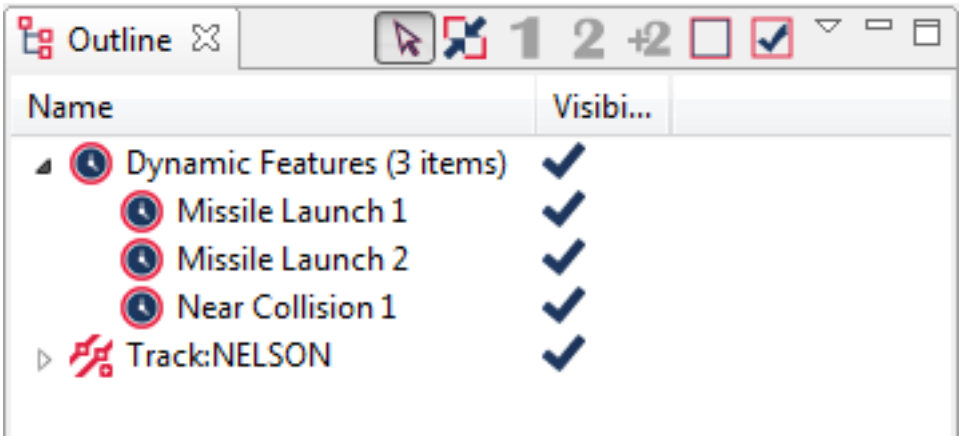
Additional timers can be added, as required.

As can be seen in the screenshot above, **Dynamic Features (1 items)** is added to the *Outline view*, the *Time Display (Absolute)* properties display in the *Properties view*, and the **Absolute Time** is shown in the bottom right of the *plot editor*; this is its default position.

In the *Properties view*, you can modify the display properties:

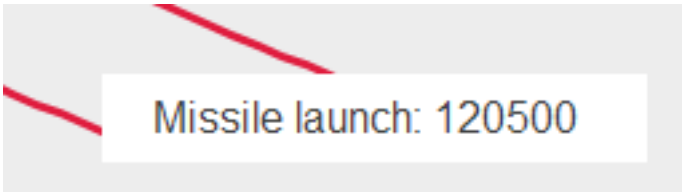
Background color	By default, the background color for the time display is white (RGB 255, 255, 255), but can be changed as required.
Color	The font colour used in the display.
Font	The font type used.
Location	The default location for the time display is Bottom Right, but Bottom Left, Top Right, and Top Left are also available.
Name	You can rename this time display as required. Once done, its name will change in the *Outline view*:

Figure B.2.6. Renaming Time Display



Prefix

By default, the display does not have a prefix, but you can set this here. For example:

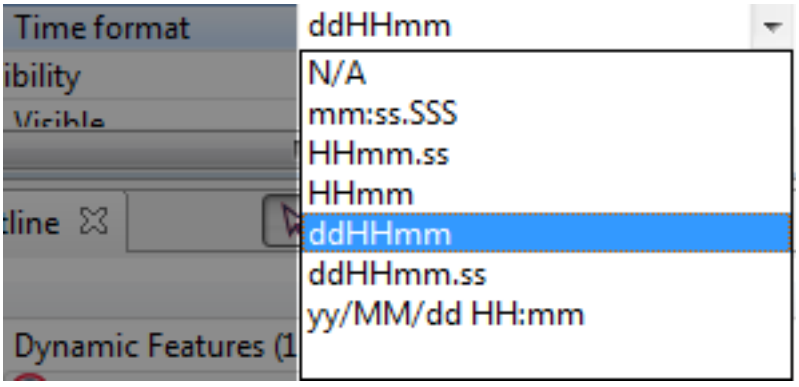


Suffix

Similarly, you can also specify a suffix:

Time-related

Use this option to change the **Time format**:



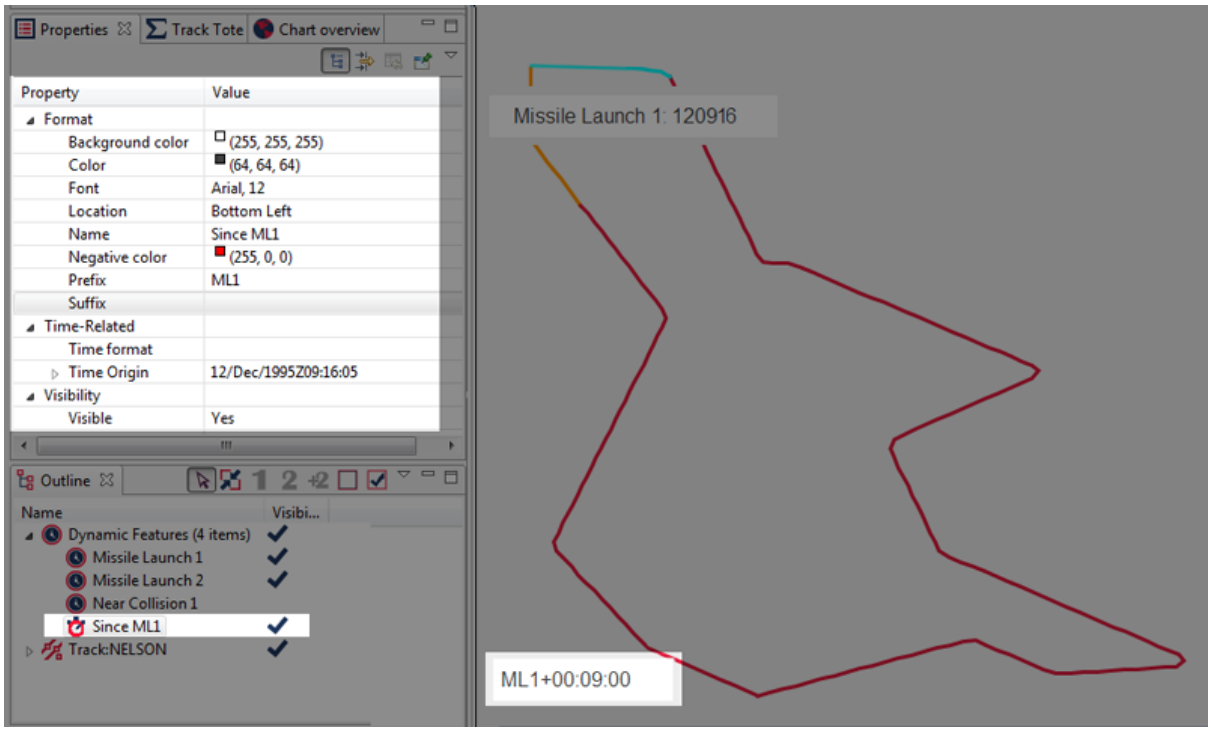
Visibility

This option is a simple toggle option which allows you to show or hide the time display.

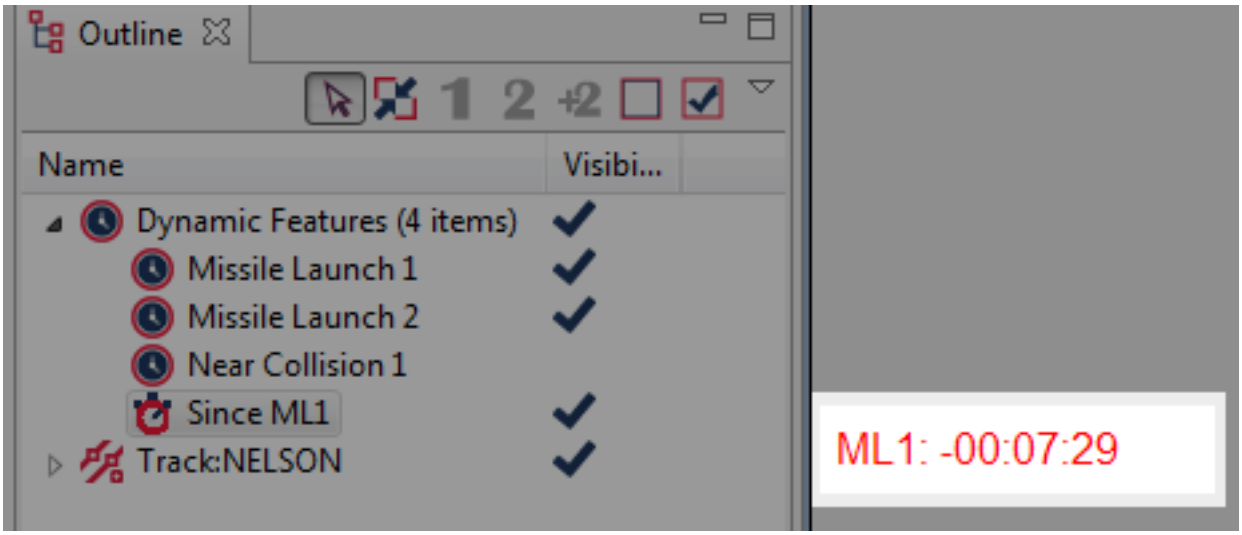
2.4. Time Display (Relative)

Debrief also allows you to add a relative time display to the *plot editor*. You add this in the same manner as you add a *Time Display (Absolute)*. It will show as follows:

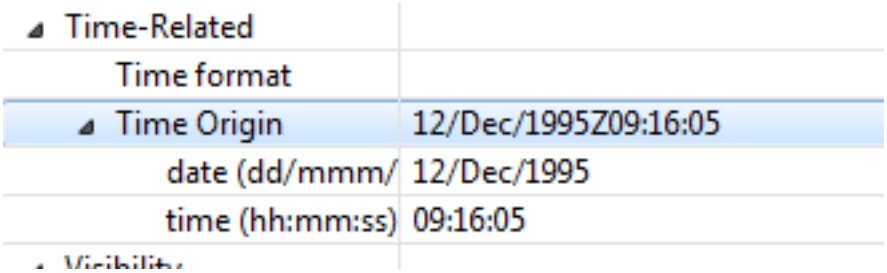
Figure B.2.7. Time Display (Relative)



Background color	By default, the background color for the time display is white (RGB 255, 255, 255), but can be changed as required.
Color	The font colour used in the display (set to RGB 64, 64, 64 by default).
Font	The font type used.
Location	The default location for the time display is Bottom Right, but Bottom Left, Top Right, and Top Left are also available.
Name	You can rename this time display as required. As shown in the image above, this will change in the *Outline view*.
Negative color	If you step back through the timer in the *Time Controller view*, the absolute time could precede the *time origin* specified for this relative time display. If so, specifying a negative colour provides a clear indication here:



Prefix	As with the *Time Display (Absolute)* you can specify a prefix for the relative timer (the image above has been assigned ML1:) .
Suffix	If your time display warrants a suffix, you can add this here.
Time Related	The absolute display only has a single time related option (time format), but the relative display has an additional one (time origin).
Time format	Select from a number of time format options.
Time origin	Unlike the absolute time display, which mirrors the time from the *Time controller view*, the relative time is set to coincide with an event in the current scenario. You can specify the exact start time here:



Visibility	This is a simple toggle option which allows you to turn the time display on or off without deleting it from the *Outline view*.

2.5. Grid

Next, try with a new grid: 

Auto Mode	In auto-mode Debrief assesses the current screen size and area of data covered, and attempts to set the most appropriate range of values and step size for the scale. A good working practice is to switch to auto-mode to allow Debrief to estimate the optimal values, then switch out of auto-mode to fine-tune the ScaleMax and ScaleStep values provided.
Color	The colour used to draw the scale.
Delta	The interval between plotted lines
PlotLabels	Whether to label the grid. See tip below for details regarding how the labels are formatted
Visible	Whether you can hide the grid, of course.

Tip

Two methods are used to produce grid lines:

- *Lat/Long Grid*. Where angular delta units are selected (degrees, minutes), vertical and horizontal grid lines are calculated relative to the latitude of the plot (thus a 1 degree grid requested at 60 degrees North will have grid lines of 60nm separation in latitude, but with lines of longitude at 30nm separation).
- *Square Grid*. Where distance related delta units are selected (m, yd, km, nm, etc), the vertical and horizontal grid lines are constructed using the same delta distance (thus a selected delta of 1 kyd will have lines of 1 kyd separation in the horizontal and vertical).

2.6. Local Grid

The Local Grid () is a modified grid for which the origin has been over-ridden. Change the Origin attribute to move the grid origin. The PlotOrigin attribute has been provided to draw a small point at the origin of the grid - useful when initially designing/recording the grid.

2.7. Coastline

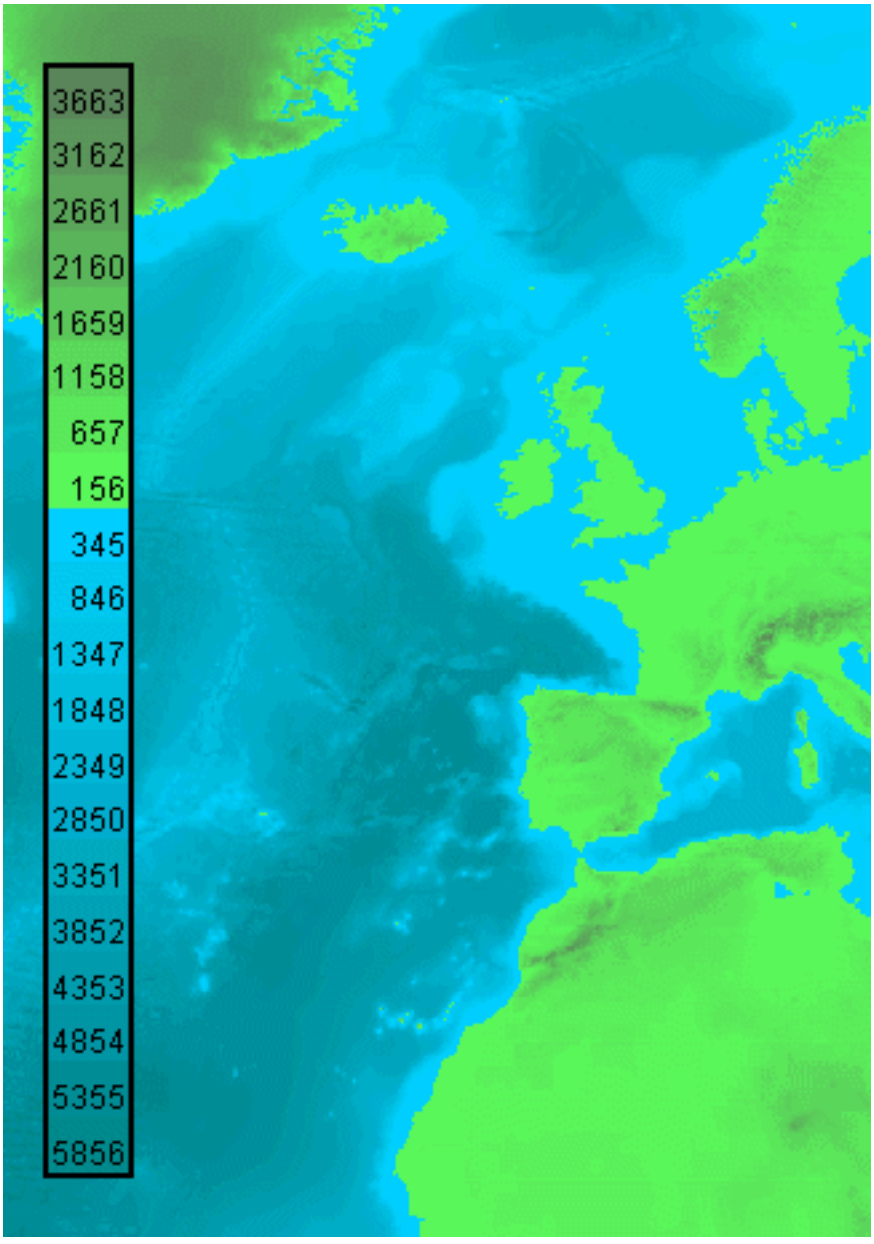
The Debrief installation includes a low-resolution coastline datafile. Whilst it does cover the whole globe, it does so at a low resolution, so is only useful for an overview. The vectored chart data discussed later provides a much lower resolution of data.

Figure B.2.8. Sample of default coastline data

2.8. ETOPO gridded bathy

Whilst the VPF dataset provides a contoured bathymetry within broad depth steps, the ETOPO dataset provides a gridded bathymetry in 5' or 2' steps. When you ask Debrief to plot an ETOPO background, Debrief will try to load an ETOPO-2 dataset first, followed by an ETOPO-5 dataset if that is unavailable. The image below provides a sample of the level of detail supplied.

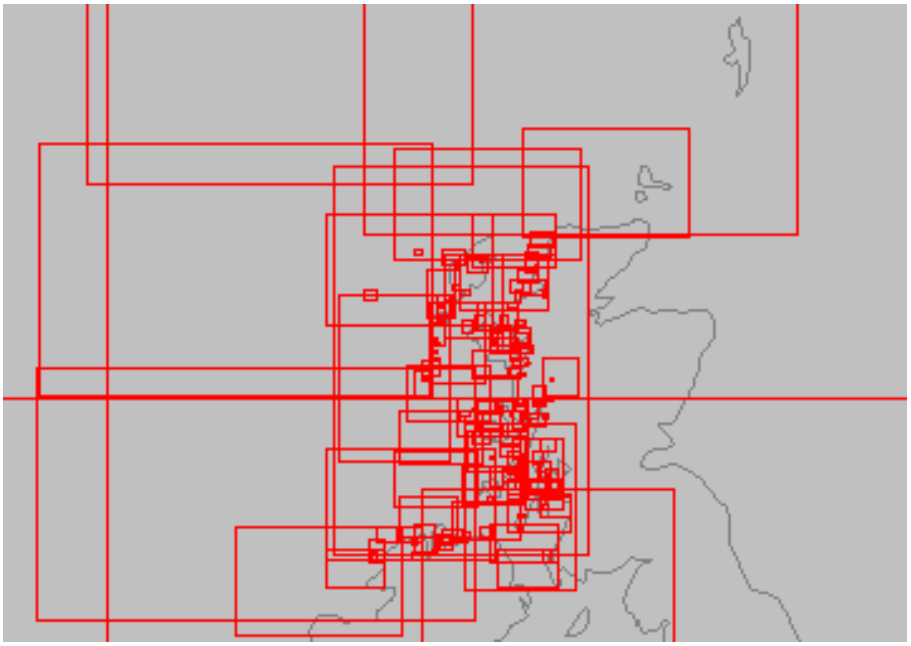
Figure B.2.9. Sample of ETOPO gridded bathymetry



2.9. Chart Library

Debrief also allows you to load 3rd-party chart libraries. This is covered in Section 2, “Loading data”

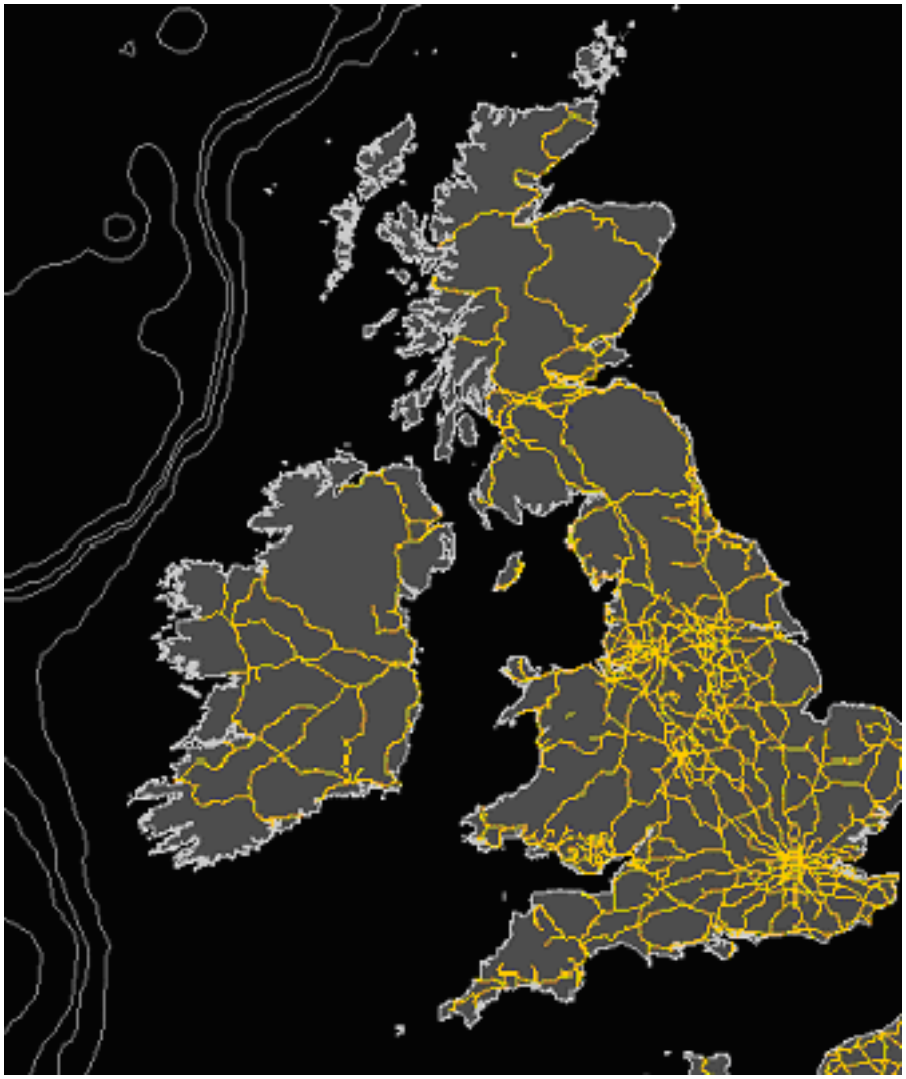
Figure B.2.10. Sample of chart library portfolio



2.10. Vectored data

The addition of vectored chart data is also covered later in this document, in Section 3, “Viewing VPF data”. The image below provides a sample of the level of detail supplied.

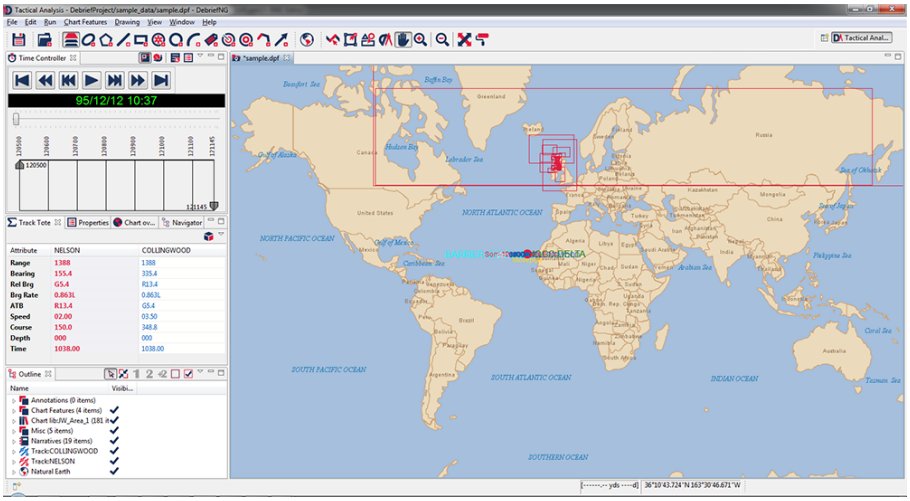
Figure B.2.11. Sample of vectored coastline data



2.11. Natural Earth

In the same manner that we loaded the Chart Library, we do exactly the same thing with Natural Earth.

Figure B.2.12. Sample of Natural Earth



For further information, refer to Section 1, “Natural Earth data”

3. Adding drawing features

The Drawing toolbar and menu are used to place geographic features on the plot; features which are geographically fixed. In general, when you create them, their corners must be specified by copying a location from the plot and posting it into the property of the relevant corner.

Figure B.2.13. Drawing toolbar

Note

By default, new drawing features are placed in the `Misc` layer, from where they can be moved to other layers as required. If you're creating lots of drawing features, and want to place them directly into target layers, select on Manually select target layer from the top of the Drawing menu. When this setting is ticked, Debrief will prompt you to select a target layer for each new drawing feature added to the plot. Clear the ticked setting to return the target layer to the `Misc` layer.

3.1. Label

The Label drawing item allows you to place a labelled symbol on the plot. Typically this may be used to annotate events on the plot, or to add an extra feature which did not warrant its own Debrief REP entry. Like all annotations Labels have time start and end properties. Because of this they can be placed on the Tote (see Section 1.1, “Tote area”) and used in analysis, as you will learn later.

Tip

A custom editor is supplied (see below) for editing locations of data items, it is used frequently in Debrief. Clicking on the expand button will let you edit individual lat/long fields (shown in figure Figure B.2.15, “Location Editor (expanded view)”). Alternatively, you may set the location to a specific point on the plot. Right-click on the plot and select Copy cursor location. Then click once on the location editor to reveal the Paste button (shown in Figure B.2.16, “Location Editor (paste view)”), and click it to paste the cursor location.

Figure B.2.14. Location Editor (initial view)

Figure B.2.15. Location Editor (expanded view)

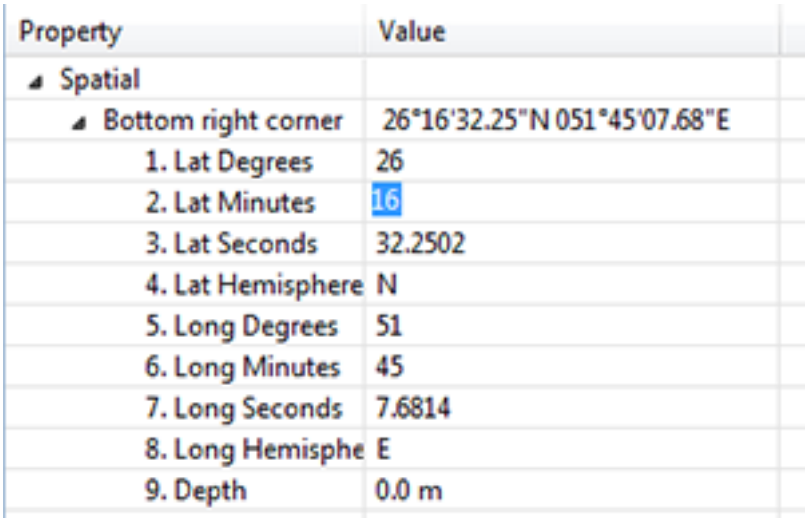
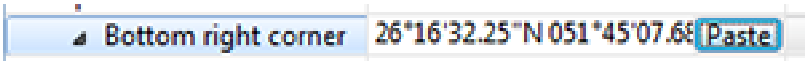


Figure B.2.16. Location Editor (paste view)



Note

Note that the text for the label itself can be a multi-line piece of text. When in the text-editor box for the label, just press the return key on your keyboard to move to the next line. The multi-line piece of text will be centre formatted on screen according to the Label Location property. The multi-line label is also available for all labels on the Debrief plot. Note that when the label is stored to disk in the Debrief plot-file format, the '\n' character used internally to represent the new line is converted to a '\\n' string to allow its easy storage.

3.2. Ellipse

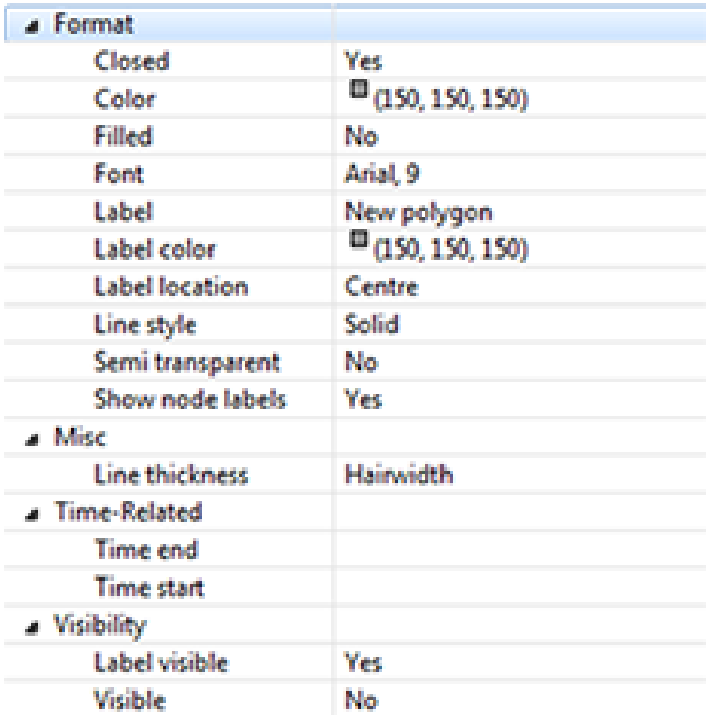
The Ellipse shape works in much the same way as a Label, except an ellipse is drawn on the plot instead of the labelled symbol. The size of the ellipse is dictated by the Maxima and Minima values which specify the lengths of its semi-major and semi-minor axes, expressed in user-selectable distance units. These values are the distances from the centre of the ellipse to the furthest and closest points on its perimeter, respectively. The direction of the ellipse is specified by the orientation, expressed in degrees. Debrief does not check that the maxima is larger that the minima, it merely plots an ellipse oriented about the semi-major axis.

3.3. Polygon

The Polygon drawing feature allows more varied shapes to be plotted within Debrief. A series of points (called a Path) are added to a Shape which are the connected up to create a Polygon. The points in the path may be typed in, double-clicked, or dragged to produce the correct polygon.

When a new polygon is created, its editor panel will open as shown below:

Figure B.2.17. Initial view of polygon



The properties shown are similar to those shown for most other shapes, with the exception of the actual locations of the polygon. Change the location of the nodes that form the polygon by selecting them in the Outline View then editing them the Properties View (see Figure B.2.1, “Properties view”). New nodes are created by right-clicking on the Polygon (in the Outline View or on the Plot), and selecting Add Node.

3.4. Rectangle, Circle, Line, Arc

The Rectangle, Circle and Line shapes work in the same way as the others described here, the Location Editor described above being used to set the corners, centre, or ends as required.

Note

Some of the more basic drawing items have a single DTG parameter instead of TimeStart and Time_End properties. Where there is a single time, the time-analytical features of Debrief (such as Snail mode) treat the feature as *alive* for three minutes either side of the time value.

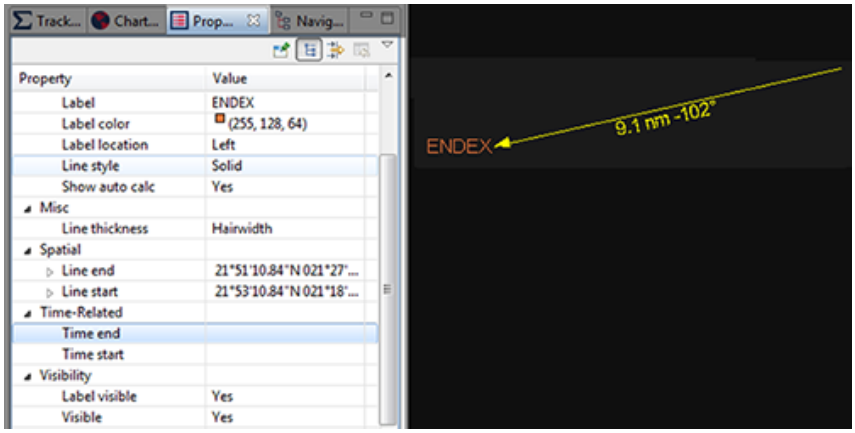
Tip

To draw a line created from an origin with values of range and bearing, first create the line, and put the start point of the line at the origin. Now switch the mouse mode to Range/Bearing (). Now click on the button labelled Select Point for LineStart, and drag the mouse out from the origin point. The current range and bearing will be displayed at the bottom-left of the screen. When the mouse is at the desired range and bearing right-click and copy the position onto the clipboard. Next, paste that location into the LineEnd value.

Note

In Summer 2013 the Line Shape annotation received some love and gained new capabilities. One is the ability to have an arrow displayed at one end. This is useful in indicating the direction of travel, or in highlighting a particular location. The second capability is Show Auto Calc - this displays the calculated range and bearing 1/2 way along the line. Unfortunately this label is only visible on screen - the structure Windows Metafile format doesn't allow us to copy the rotated text across the clipboard to copy it into MS Word. So, you'll only get this rotated text in MS Word is you capture a screenshot, using software such as SnagIt or Hypercam. If you do want to include such a label in your analysis report, and you wish to copy/paste the image via the clipboard, then you can always manually put the range/bearing into the line label, and specify that the label be shown 1/2 way along the line.

Figure B.2.18. Calculated Label



3.5. General

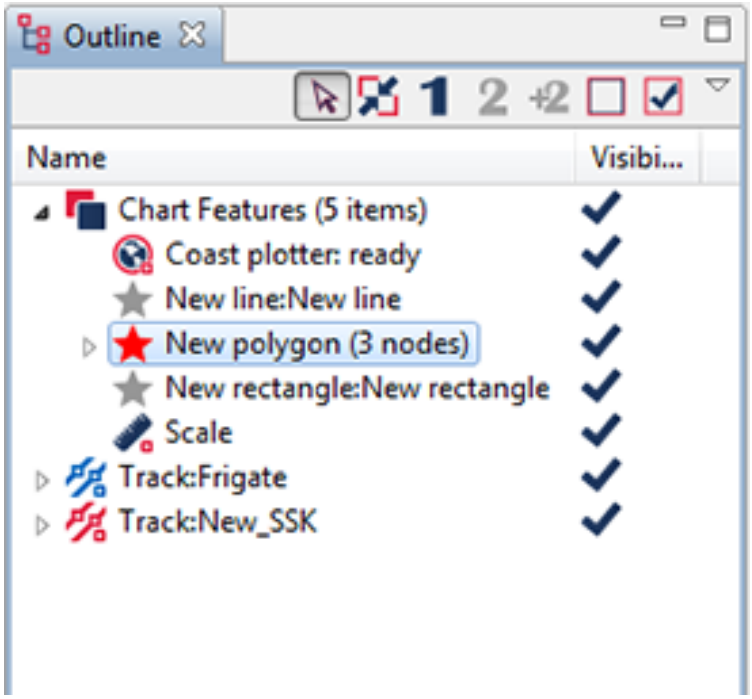
To edit any existing annotation, double on it on the *plot* or the Outline View to select it - its attributes will then be available from the Properties View. Alternatively, right-click on an item on the Plot or Outline View, and editable attributes are available from the object's drop-down list. Only boolean (true/false) and list-related (top/bottom/left/right etc) are editable from the right-click menu.

4. Layer management

4.1. Outline View

The Outline View provides a tree-oriented view of all of the editable data within a plot, organised into layers. New items added to the plot are placed into a *Misc* layer. From here they can be reorganised into suitably themed layers via cut/paste (see Section 4.2, “Cut/Copy and Paste”).

Figure B.2.19. Outline View



Reveal the Outline View by selecting it from the Window/Show View... drop-down list. Once open you will see the data shown as a series of layers - each with a tick indicating if it's visible or not. A toolbar provides command buttons to make items visible or not visible, and buttons are enabled to make an entity the primary or secondary track, or to add the selected item as another secondary (when applicable). Note that not just tracks may be denoted primary or secondary: and graphic item with a position and date-time can be selected: thus Debrief can be configured to show a running indication of bearing and range for a series of vessels from a single label datum (representing a sonar-buoy or mine).

The drop-down menu provides further commands, significantly including the ability to add a new layer; Create Layer. Further commands are available when right-clicking on one or more Outline View elements: in particular the ability to cut/copy/paste elements between layers and plots, together with the command to view a time-variable graph of the selected elements (see Section 4, “Show time-related variables”).

4.2. Cut/Copy and Paste

Items may be cut, copied and pasted between Layers using the commands contained in the menus obtained by right-clicking on the item. The cut command may be used on its own to delete an item. Copying items is a quick way of making duplicates of a correctly formatted annotation or screen item such as a rectangle or ellipse.

In addition to copying/moving items between layers, items and layers themselves can be copied or moved between sessions within Debrief.

To paste a layer (or track) into the top level of the Outline View just click on the white space below the displayed layers and select Paste Item form the popup menu.

Note

All top-level layers (that it screen items which appear at the top level in Layer Manager, such as Tracks, ETOPO data and Layers themselves) can have a line width assigned to them. This setting is observed when plotting all items in that layer. The smallest line width

supported is *hairwidth*, which plots the finest line the particular output device (screen or printer) can produce. The line-width parameter can be accessed by right-clicking on the screen item directly in addition to via the Outline View.

Warning

In addition to the Cut menu command, is the Delete command. This doesn't store the removed items on the clipboard, and is suited for bulk data point removal. To further ease the memory burden of this large operation it isn't possible to Undo the Delete command.

Warning

The Outline View performs particularly poorly when it has layers that contain 10s of 1000s of entries - sometimes taking several minutes to refresh all of the labels/icons in that layer. So, if you have a layer that contains more than 10,000 items, the Outline View will not let you expand it - avoiding the performance issue. If you do have, for example, a track that contains many 10s of thousands of positions in it, then you should reduce the data volume by:

1. If you need high frequency data, reduce the time period covered by the track by selecting a reduced *Time Period* in the *Time Controller*, then selecting Trim Tracks to Time Period from the track's popup menu. This will reduce all data points that fall outside the current period indicated in the Time Controller slider bars

2. If you need to cover a long period of time, then select Resample data at xxxx, and select a larger time period (10 mins, 30 mins?). This will reduce the number of positions in the track.

.

5. Saving and re-opening plots

5.1. Save

You may (or my not) know what the *Replay* file format used to load data into Debrief looked like. Whilst it is compact and easy to use, unfortunately the Replay file format does not allow us to store all of the formatting we have applied to the Debrief *plot*, nor the coordinates of what you are currently viewing, nor how you have arranged to store your data in *layers*. This problem is overcome by the use of the Debrief *Plot-File* format, an application defined using XML (see XML). Have a quick read about *XML* in the Reference Guide or Glossary if you're not aware of it and are sufficiently interested, otherwise, here's an overview:

- Debrief stores plots in XML format with a custom DPF suffix
- XML stands for eXstensible Markup Language, which allows structured data to be stored in text format, and is a world-wide standard promoted by http://www.w3c.org.
- XML files can be edited outside Debrief, allowing you to cut and paste between *plot-files* to build up a new *Plot-File*, all in a text editor

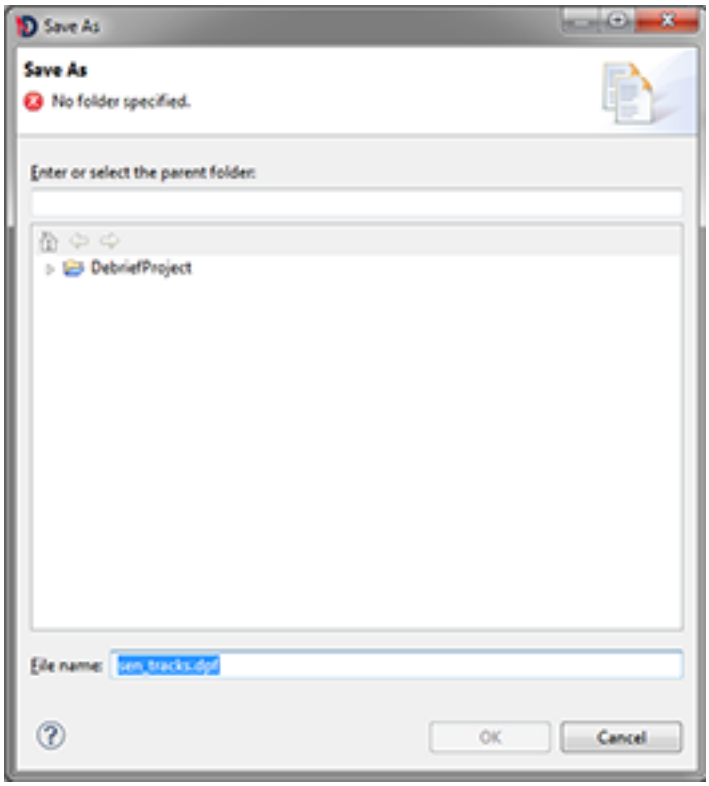
Whilst a plot is typically created from an REP file, it may not be saved back into that REP file - it must be saved as a Debrief *Plot-File* (DPF).

To save your work as a *DPF* file, click on the Save button:  on the Workbench toolbar (or via the File menu). A file dialog will open (see below), allowing you to save the current session in this format.

Note

Debrief will only let you save the new plot into one of your project folders, so you must both select a folder and a provide a filename for the new file. Usefully, Debrief guesses that you probably want to name the plot the same as the REP file first loaded, but you can change this as required.

Figure B.2.20. Save-file dialog



Save-as functionality is provided through the Save As button, next to the Save button in the File menu.

Debrief plot files can grow very large, and on occasion the memory required for the save operation can reach the upper limit provided to the application by default. When Debrief fails during the Save operation due to reaching the upper memory limit it stops trying to save to file and shows a message dialog advising you to increase the limit:“Ran out of memory whilst saving plot, try adding -Xmx256m to the command line”. This extra parameter passed when Debrief is starting instructs your PC to provide Debrief with 256 Megabytes of memory. If you continue to receive the error message try increasing the memory allocation to 512 Megabytes.

The save operation itself is conducted in a two-stage process. If the operation entails saving over an existing file (when you just do a plain Save, or Save As over-writing an older version) Debrief first writes the plot to a subtly different filename in the same directory (`plot.dpf` would get saved to `~plot.tmp`). If the save operation completes successfully then the older file is deleted and the temporary file renamed to the originally requested filename. Thus, your existing file only gets overwritten on successful completion of the save.

5.2. Adding more data to a plot

With a plot already open, you are able to drag a Debrief DPF or REP file into its chart, adding the content to the existing session. Dragging multiple files onto the plot area adds them all to the existing

session. To create a composite plot like this, you can start by using Debrief's *New Plot Wizard* (see relevant Cheat Sheet, Tutorials)

6. Lightweight tracks

6.1. Introduction

The Debrief track object is a heavyweight entity that contains many, many capabilities. Some of the more complex capabilities include the ability to compose a track from multiple TMA solutions, storing measured sensor data within the track, or constructing a track using *DR*, rather than *OTG* measurements. These capabilities come at a performance (and UI) cost, so the concept of *lightweight tracks* has been introduced.

In addition to having higher performance (through reduced capability), lightweight tracks can be organised into layers, enabling them to be collectively switched on and off. On import from Replay file, lightweight tracks are denoted by specifying the name of the target layer into which they should be stored. See an example in Table D.1.5, “Debrief extended symbology data fields”. Here is an example of how to indicate the folder into which a lightweight track is to be stored:

```
100112 131314 F003 VC[LAYER=red] 00 01 59.10 N 001 10 58.22 E 70.6 6.0 0.0
100112 132314 F003 VC[LAYER=red] 00 02 18.96 N 001 11 54.84 E 70.6 6.0 0.0
100112 133314 F003 VC[LAYER=red] 00 02 38.82 N 001 12 51.45 E 70.6 6.0 0.0
```

So, in the previous example we're indicating that the track titled *F003* should be placed into a folder called *red*. Note: no other changes are made to the line of REP data, we just provide the indication of the target folder.

6.2. Managing lightweight tracks

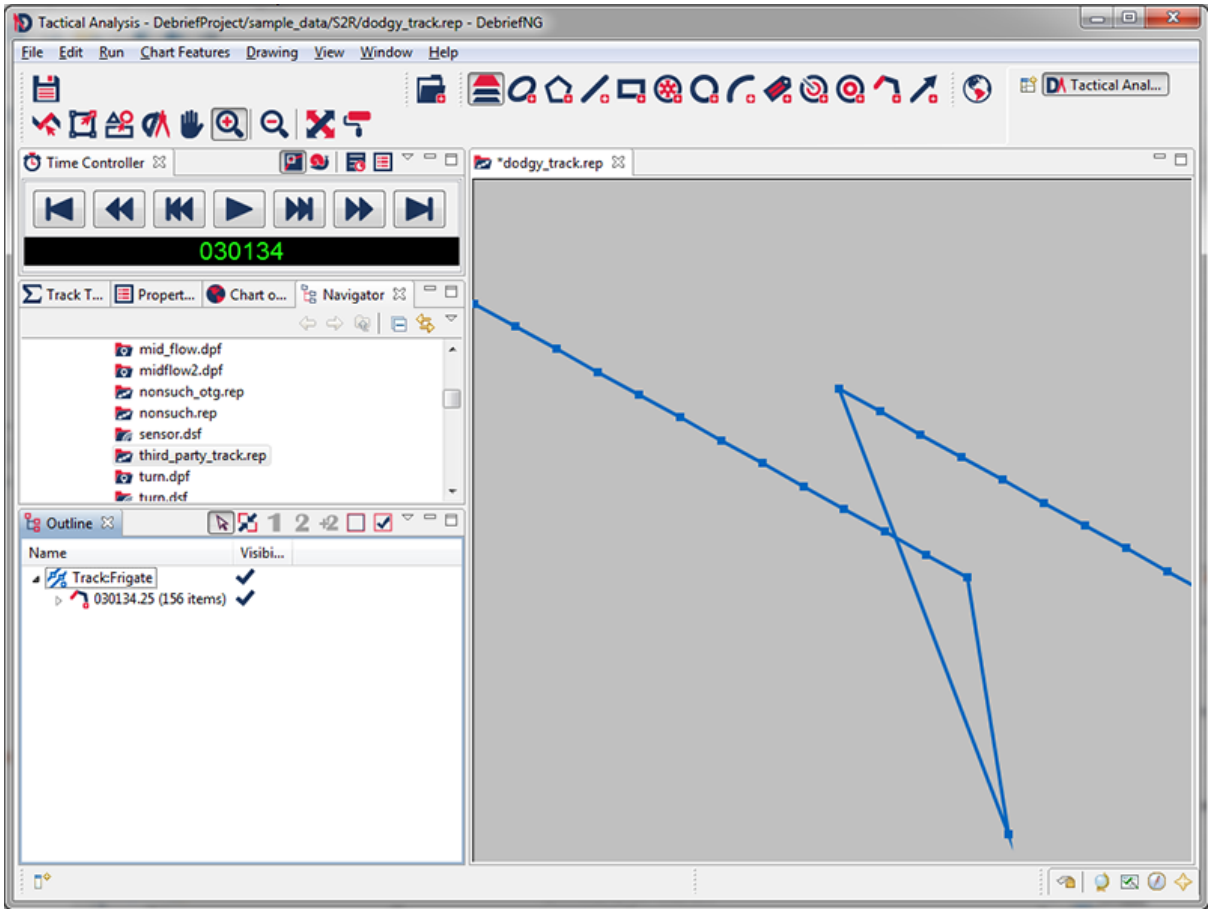
In addition to loading lightweight tracks directly from REP files, it's possible to convert conventional heavyweight tracks to/from lightweight tracks (via right-click), and to drag lightweight tracks between folders.

7. Grooming track data

7.1. Introduction

Track data can arrive at Debrief in vary degrees of quality. Traditionally MWC pre-process track data to remove jumps and generally smooth the data. The 2009 Single Sided Reconstruction extensions (see Section 2.3, “Debrief 2001 onwards”) added functionality to Debrief to perform elementary track operations, including smoothing and removal of jumps. One approach to avoiding the jumps commonly associated with data from inertial navigation systems is to load the data in DR mode (see Section 5.4, “Import modes”). The alternative is to move the track before/after the jump to line up the two sections.

Figure B.2.21. Grooming Track Data



7.2. Resampling track and sensor data

On occasion the data loaded into Debrief is of the wrong frequency:

Too dense — a track may have data items at a frequency higher than that necessary for the current analysis, resulting in slower than necessary Debrief performance. Alternately, a period of sensor data may be too dense, obscuring the underlying patterns in the data (such as a bearing fan)

Too sparse — During track reconstruction, it may emerge that the sensor data is of much higher frequency than the position data. After reconstruction, when stepping through a track in time the positions may *jump* to the specified time - indicating the positions are too sparse. Resolve this by either resampling the positions or by directing the track to use interpolated points.

Problems associated with data density are handled differently between tracks and blocks of sensor data. Tracks are resampled using the ResamplePositionsAt property - which adds or removes position observations as necessary. Sensor data is resampled using the VisibleFrequency property - though note that this does not add or remove points, it merely changes their visibility.

Note: sensor and TUA data have a characteristic where there may be empty periods when the contact is not held. It would be wrong to continue interpolating data points during these periods, so data points are not generated if there is a gap of more than 3 minutes between data points.

7.3. Splitting track sections

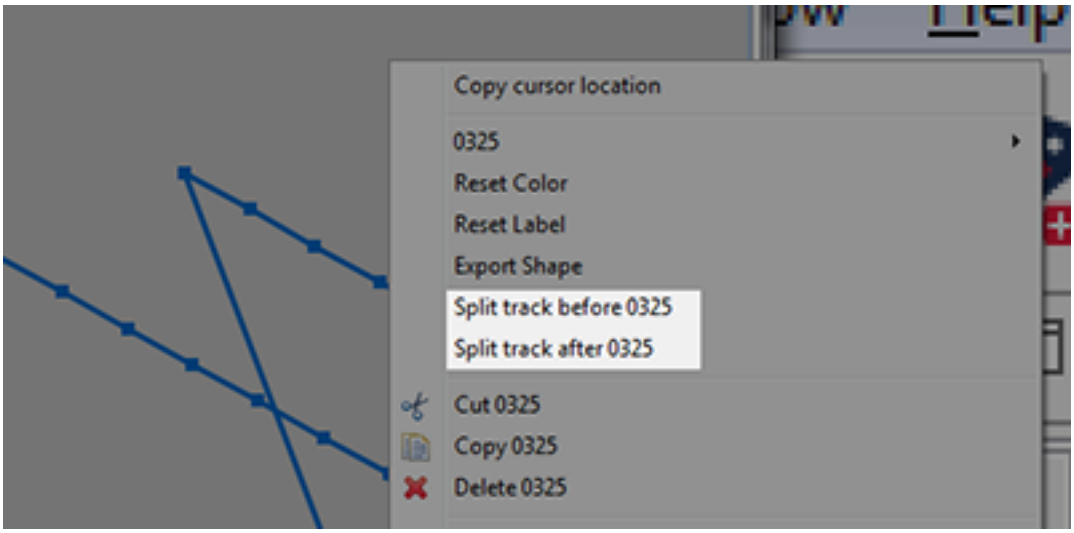
Before a jump can be removed from a track, the track must be split. To split a track, first decide where the split should be made, and right-click on the point where the split should happen (either on the plot

or in the Outline View). The popup menu will provide options to split the track immediately before or after the indicated point.

Tip

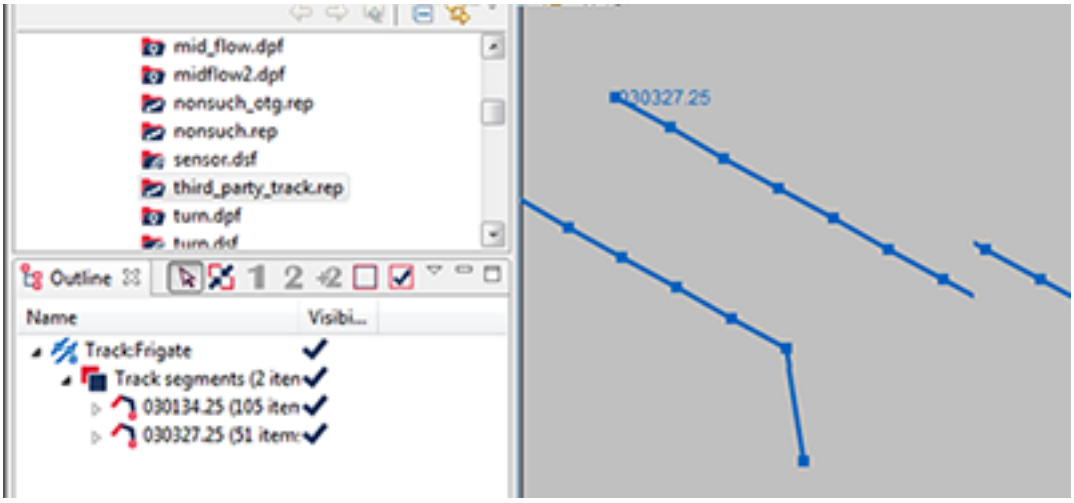
It might be worthwhile enabling the **Start/End time labels** attribute so you can see exactly where the start and end times for each track segment begin and end. Refer to tutorial 2, **Viewing some data**, and the procedure for **Adding data to a plot**.

Figure B.2.22. Splitting Track Sections



On completion you'll see a visible split in the track, and the track represented as two segments in the Outline View.The track segments are named according to the DTG of their start point. The two track segments can now be manipulated separately (dragged) using the 'drag TMA segment' tool as described later (see Section 2.3, “Dragging tracks”).

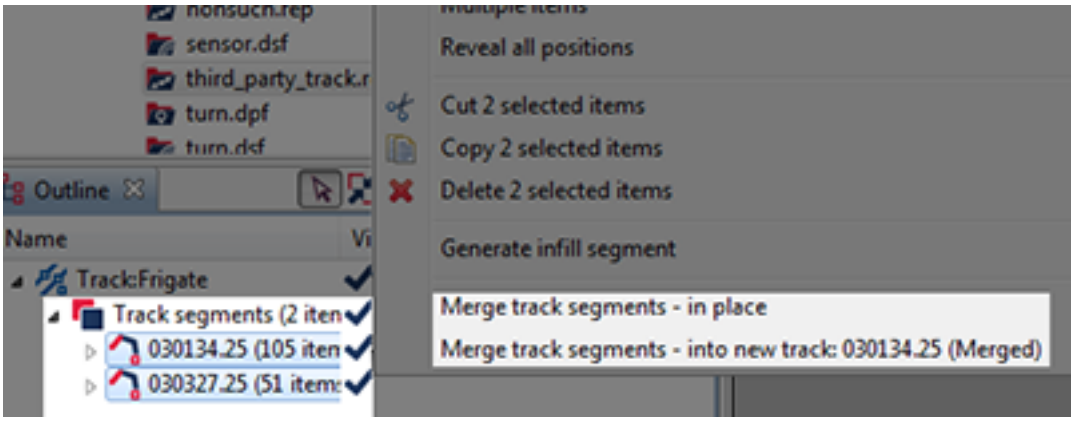
Figure B.2.23. Track Sections Split



7.4. Joining track sections

Once track segments have been manipulated (see Section 2.3, “Dragging tracks”), they can be rejoined. To rejoin tracks, select them both/all in the Outline View (using the Control key as appropriate). Then right-click and select Merge Tracks as shown below. Note, you can select which track becomes the 'host'.

Figure B.2.24. Joining Track Sections



The Debrief merge algorithm is not able to handle overlapping tracks, since it cannot resolve which overlapping segment to favour. This is a task best undertaken by the analyst. So, Debrief will produce a warning when the analyst tries to merge overlapping tracks. When it does, just delete one or more overlapping points and trigger the operation again.

Once you have performed a merge operation on a collection of track segments they are no longer relative to the ownship track (in terms of range/bearing). They are now standalone tracks and can by Copy/Paste-d into another Debrief plot if you wish. The equivalent operation to make a TMA Segment standalone is to select Convert [xxx] into standalone track

7.5. Combining track sections

On occasion, once a series of track segments have been built up (or created using the Generate TMA wizard) you may wish to *combine* them in order to make their management easier. If you wish to keep them as individual entities instead of merging them into a single new track segment, you may combine the track segments (available when you right-click on more than one TMA Solution). Performing this operation puts the segments into a single parent track whilst retaining them within this as single entities. This is particularly relevant for TMA Track Segments, since you may wish to continue dragging them individually as you refine your estimates of course/speed.

7.6. Generating infill segments

In addition to combining periods of track (legs), Debrief is able to generate sections of track to join existing sections. It performs this using a series of Cubic Spline calculations. The Splines cover Lat, Long, Depth, Course and Speed. In this way, the infill segment introduces track points that provide a smooth transition in the above attributes from one track to the other. The time interval used for the data points in the infill section is taken from the time interval between the first two points of the second segment. On the plot, the infill segment is shown as a dotted line, to indicate that its series of purely calculated positions, and not based on any observed measurements.

Tip

These infill sections are actually dynamic. So, as you move/drag your manual TMA legs around, Debrief will recalculate their positions each time you drop the track. Go on, have a go. It's really flash.

Note

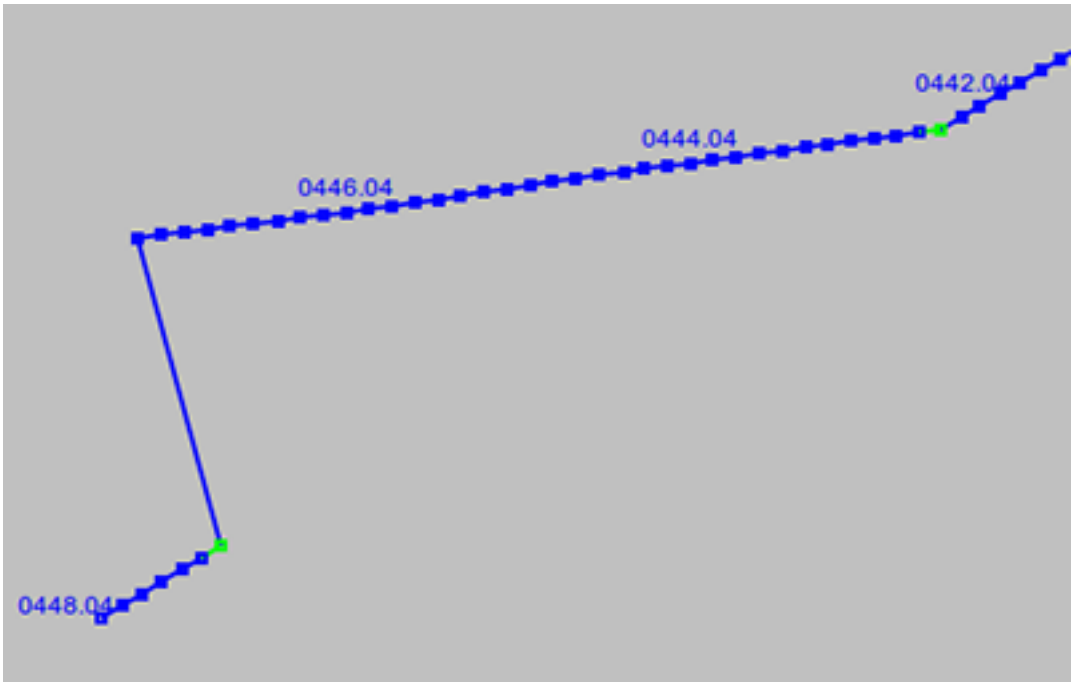
If you delete the track segment either side of the dynamic infill track section, then the infill section will also be deleted - since its positions can no longer be calculated.

7.7. Removing Jumps

A consequence of submarine dived inertial navigation is that the recorded track shows large jumps each time a GPS position fix is introduced, correcting the accumulated drift whilst dived.

Debrief is able to automatically remove jumps, transforming less accurate track periods so that they tidily match high-confidence (but infrequent) GPS fixes. A jump is determined as two consecutive updates where the speed in the second update is 3 or more times larger than that in the previous update. The following fictional diagram shows a period of drift, with two green markers indicating the start/end of the dived period. A GPS fix was obtained at the green marker near the North-West. Then the submarine dived, and the drift error grew cumulatively. The next GPS fix was obtained at the time of the green marker near the South East.

Figure B.2.25. Removing Track Jumps



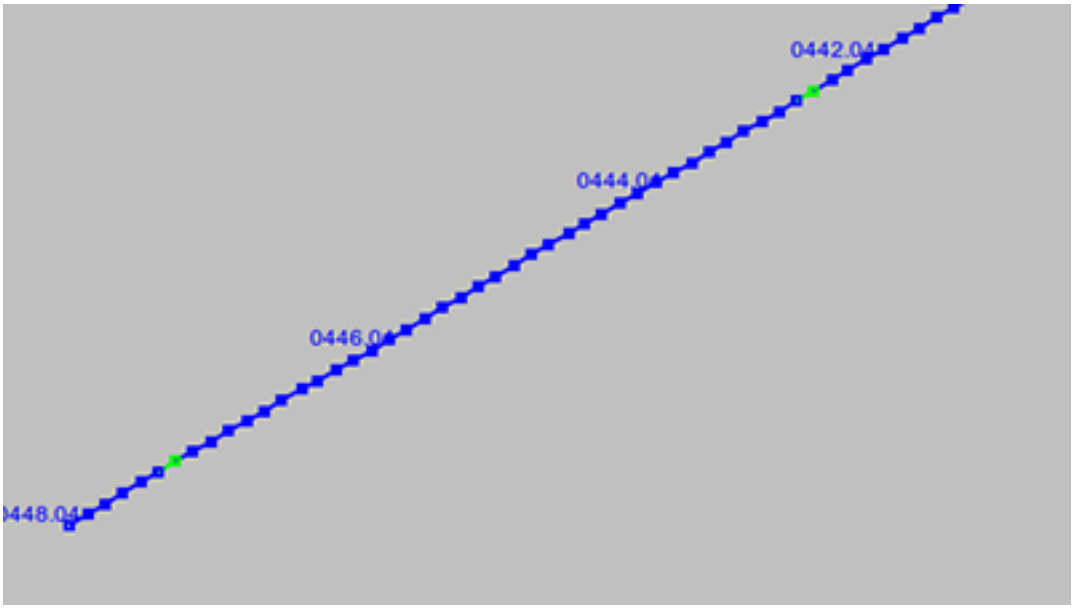
To remove jumps for this period, the analyst opens the Outline View, then selects positions for the whole period representing the drift section of track, from the first green marker to the second green marker (inclusive).

Tip

Use the shift-key to select a continuous sequence of items

After right-clicking on this period of position data the analyst will select Remove jumps in selected positions. The algorithm determines the size of the large jump at the end of the period, then works through the selected positions, applying a proportion of that large jump to each position, ensuring the green positions remain unchanged.

Figure B.2.26. Track Jumps Removed



In addition to selecting a discrete period of positions, it is possible to right-click on a track's *Positions* element in the Outline View, then selecting Remove jumps for selected track. But, please note that this will probably give a sub-optimal solution. When the whole track is selected, Debrief uses the first point in that track as the first lock point. For a more accurate tidied track, manually select the positions, ensuring a real GPS fix is used as the first point.

Note

Read more about the algorithm underpinning Remove Jumps in Chapter 3 of the Reference Guide Section 4, “Remove Jumps”

7.8. Transferring a relative track to another host

Sometimes single-sided reconstruction is performed at a range of fidelities. A DR reconstruction may be used to get maximum accuracy over a couple of hours. In parallel, an OTG reconstruction may be considering the route of a platform over several days. The track segment generated for the first can't be cut/pasted onto the second, because the ownship platform will have followed a different route (DR vs OTG). The following procedure can be used to solve this issue

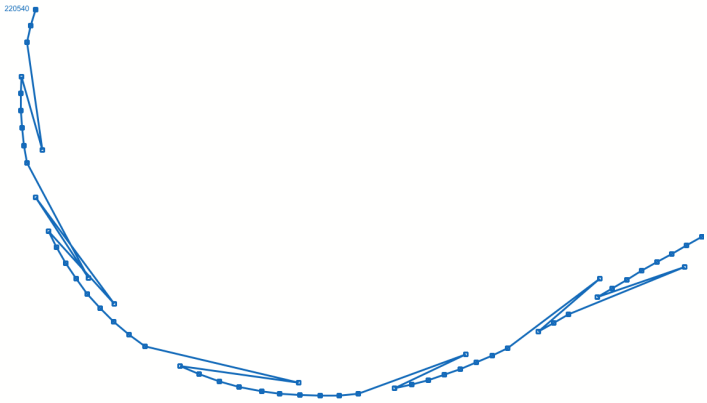
The track segment from the DR reconstruction can be considered as a set of range/bearing vectors from the DR primary track. We can then generate a new leg of target data by using these offsets to the primary track on the OTG plot.

To do this, right-click on the reconstructed track and select *Copy to clipboard as offsets from [Primary track name]*. This operation has put the set of offsets, time-stamps, and original track name onto the clipboard.

Next, either from this plot or a large-scale OTG plot, right-click on the new ownship track, and select *Create new track by adding clipboard bearings to [Selected track name]*. A new track will be added to the plot, where each point was generated by adding an offset to the nearest point-in-time on the subject track.

7.9. Smoothing out back-tracking jumps

Figure B.2.27. Smoothing back-tracking jumps Jumps



The above illustration shows a phenomenum sometimes encountered in some track recorded track data. Individual points jump back down the path already travelled. These jumps jar with the attention of the person viewing the track plot, and can introduce errors to the single-sided reconstruction.

There is a pattern to the errors. They always just involve a single measurement (not a series of measurements), and the jump is always back along the path already travelled.

An algorithm has been derived to locate, and remove these jumps. The algortihm works as follows:

1. Walk forwards along the track segment, remembering the previous 3 positions

2. Once the three previous positions are all populated, they are notionally titled n-3, n-2 and n-1, with hte current position at n.

3. Calculate the direction from n-3 to n-2. This is the general direction of travel

4. If n-1 represents a step backwards along this direction, a jump may have happened

5. If the direction from n to n-2 is back on the original direction, then a jump certainly occurred.

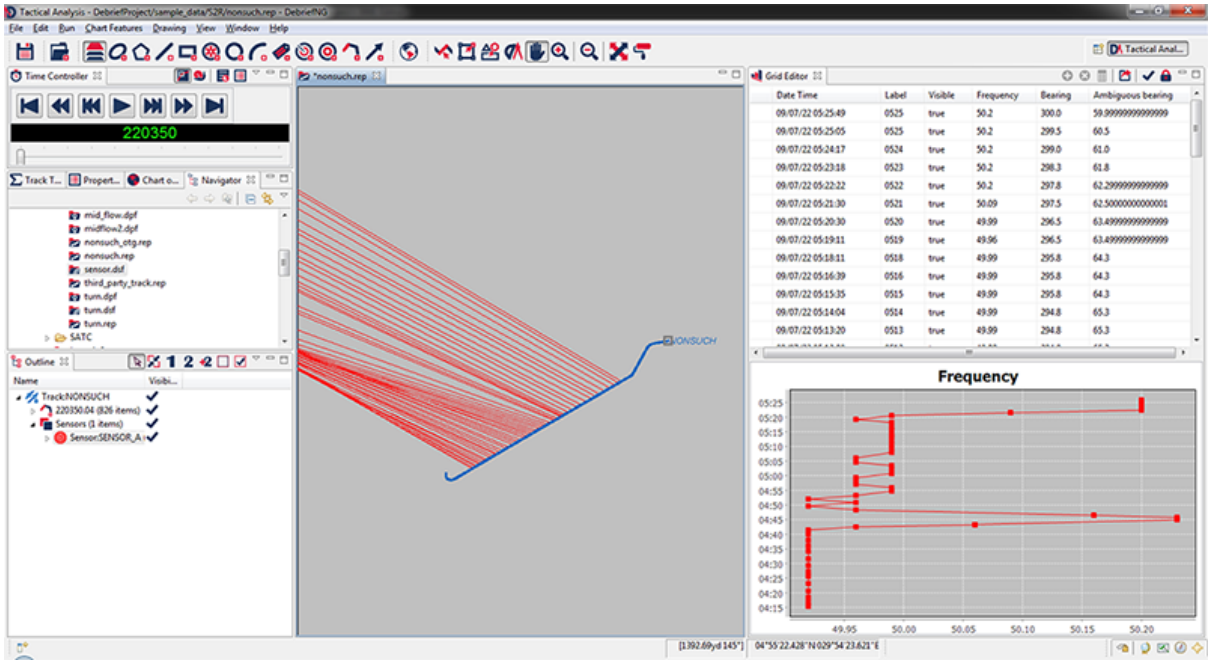
6. If a jump has occurred, then replace the location of n-1 with a point at the same time that has been interpolated between n-2 and n.

8. Using the Grid Editor

8.1. Introduction

The Grid Editor is a tabular editor with an associated xy plot that allows bulk editing of data points, together with data smoothing operations.

Figure B.2.28. The Grid Editor



8.2. Use of Grid Editor

The Grid Editor tracks the current selection in the Outline View. Thus, when you select something capable of being edited in a grid in the Outline View (such as a set of positions or sensor cuts) then it is shown in the Outline View.

Note

If this becomes inconvenient you can stop the Grid Editor from tracking the Outline View selection by clearing the Track Selection lock icon.

When a set of data is first loaded into the Grid Editor it is shown as a plain table. Click into any cell to edit its data. Buttons in the Grid Editor toolbar allow you to insert or delete a row. But, the Grid Editor becomes significantly more capable when you focus on a single attribute. Do this by clicking on the title label for a field (such as depth, bearing, etc).

When you focus the grid editor on a single field it shows an xy plot of that field. This plot can be used to highlight/remove/fix outliers and for general data smoothing operations. Drag points in the graph in order to smooth them, or for Debrief to take on the smoothing for you, select a series of rows either side of the dodgy items then select the *interpolate* calculator icon to smooth the unselected rows.

See the *Groom Sensor Data* tutorial for more guidance on these operations.

Chapter B.3. Analysing Data

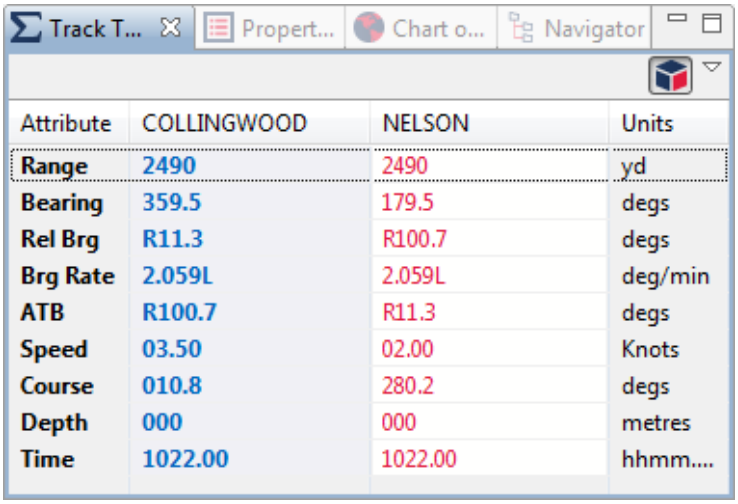
Analysis is the core function of Debrief. In this section (at last) you are going to gain some familiarity with how Debrief can be used to analyse maritime exercises.

1. Assigning tracks as primary and secondary

1.1. Tote area

We'll have a brief explanation of the Tote at this point, using the image below.

Figure B.3.1. The Debrief Tote



Attribute	COLLINGWOOD	NELSON	Units
Range	2490	2490	yd
Bearing	359.5	179.5	degs
Rel Brg	R11.3	R100.7	degs
Brg Rate	2.059L	2.059L	deg/min
ATB	R100.7	R11.3	degs
Speed	03.50	02.00	Knots
Course	010.8	280.2	degs
Depth	000	000	metres
Time	1022.00	1022.00	hhmm....

The *Tote* area is used to show the current and relative status between two or more platform tracks. The information on the *Tote* is dynamic, showing vessel information at the time indicated in the Stepper Control.

Note

The primary and secondary tracks may not contain values exactly at the time in the time stepper control. For both the tote display and the plot-highlights, Debrief uses the data values recorded on or immediately after the indicated stepper time (see the Time for more detail).

One *track* currently loaded is assigned as the Primary track (the red track in this instance), and then one or more tracks are assigned as Secondary tracks. The primary track always shows absolute data such as current course, speed and depth. With just a single secondary track specified, it also shows relative (calculated) data - as illustrated above. The secondary tracks always display both absolute data (course, speed, depth), plus the relative data respective to the primary. Thus, remember that if you want to see the primary track's relative bearing to the secondary, make sure there's just the one secondary track selected.

Not only tracks can be placed on the Tote. Most annotations (labels, circles, etc) can also be set as primary or secondary track. If no time is available for an annotation it is deemed to always be valid, and calculations are shown although the time field is displayed as "n/a". However, if the annotation has start and end times the tote displays "n/a" when outside that period and calculated data when inside it.

Note

It may be useful to remember that the Primary Track is usually assigned to the Target, thereby allowing a constant display of target bearing and range.

There can be any number of secondary tracks. Debrief displays calculated results of the relationship between each one and the Primary Track. To see the relationship between a pair of secondary tracks, one of them must be set to the Primary track.

The following calculated data is presented, where the current point on each track is used for the calculation:

Range	The range between the current point on the secondary track and the current point on the primary track using the *earth model* as described in the Glossary, displayed in the units stored in the Debrief preferences window. Debrief NG introduced the option of showing *slant range*, accessible from Debrief's Preferences window. If you have indicated that you want to view slant range, then the range shown is a function of horizontal range and vertical separation.
Bearing	The bearing between the points.
Rel Bearing / ATB	The relative bearing and Angle on The Bow from the perspective of that column's track.. There are two formats used for relative bearing, depending on the setting of *Relative bearing format* in the Window/Preferences dialog. If the format is specified as UK, the R and G characters at the start of the result are short for Red and Green, which indicate that the contact is to the Port or Starboard of the secondary track. If the format is specified as US, the value uses 0 as directly ahead and continues clockwise through 180 (astern) and back round to 359.9.
Speed	The current speed of the indicated track (no actual calculation is performed here; the value from the data-file is displayed directly).
Course	The current course of the indicated track (no actual calculation is performed here; the value from the data-file is displayed directly).
Depth	The current depth of the indicated track (no actual calculation is performed here; the value from the data-file is displayed directly).
Brg Rate	The instantaneous bearing rate of the secondary track as observed from the secondary track. This value is explained in the Glossary.
Time	Where track data is not recorded at regular steps, and tracks do not have data at the same time steps, there is a likelihood that the value displayed on the tote will not have been recorded at the current tote time. The time field shows the actual time at which the data value was recorded for that track.

Note

When more than two tracks are loaded, the value n/a is shown in Tote calculations for the primary track which rely on other track data (range, bearing, rel bearing, brg rate). This is because it is unclear which inter-track relationship is being calculated. Where only two tracks are loaded (one primary and one secondary), the tote is able to show relative calculated data for both.

1.2. Assigning tracks

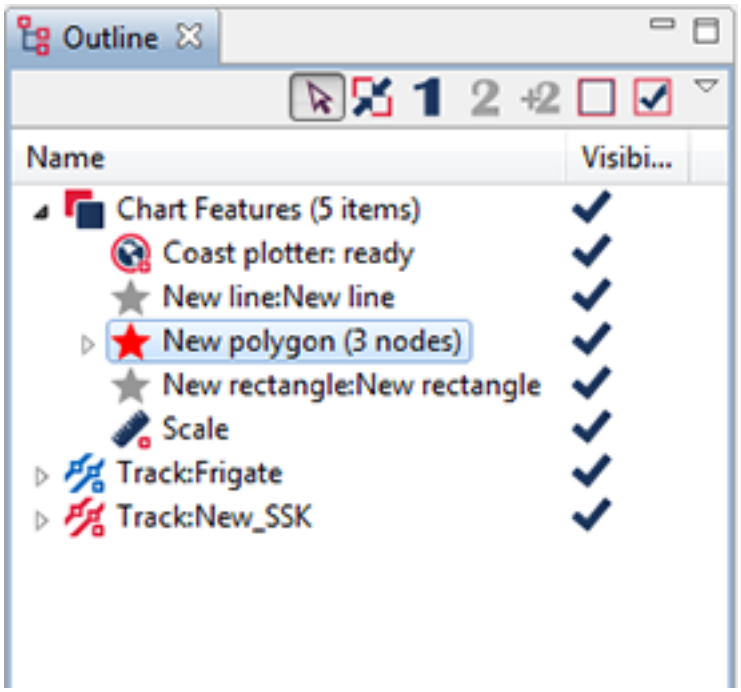
Assigning primary and secondary tracks is covered in the respective *Cheat Sheet*. Icons are provided in the Outline View toolbar to specify if the selected track should be:

1. Made primary

2. Made secondary

3. Added to the list of secondaries

Figure B.3.2. Assigning Tracks



Tip

If you have more than two tracks, it can be quicker to assign the primary track manually and then press Auto Populate to assign the remaining tracks as secondary.

1.3. Notes

Warning

A Circle currently only has a single "Centre" DTG value - so it will only be highlighted for 3 minutes either side of this point. Better results are obtained by using a Label (which has a picture of a text label on it), since Labels have start and finish DTGs.

Note

It is not just tracks that can be added to the Tote, experiment with right-clicking on features on the plot and see if they have the "Set Primary Track" or "Set Secondary Track" commands available (although read the note below about these). In this way you can make a timed data point (represented by a Circle with a very small radius) the primary track, then add a number of vessel tracks as secondary tracks, and then as you move through the data you can constantly see the vessel range and bearings from this data point. This is particularly useful for seeing vessel ranges and bearings from a *sensor* such as a sonar buoy. Remember to set the DTG data for the data point to time(s) near those of the track - or else in your example Debrief will assume the "sonar buoy" is not yet active.

Note

The limits on the time period are the outer time limits of the visible data in the primary and secondary tracks currently displayed on the Tote, so following a filter operation (introduced later) the start/end times will be changed to reflect the time limits specified.

Note

The keyboard can also be used for moving backwards and forwards, although Debrief has to know that it's the Stepper that you want the keys to control. So, after clicking on the time-slider control you can then switch to keyboard control of the Tote as follows:

- Page Up/Down keys: these control small step backward/forward resp.
- Page Up/Down arrows: these control large step backward/forward resp.
- Home/End keys: these control goto First/Last resp.

Tip

Debrief has an interpolates points property which, when ticked, interpolates the positions between actual data points. The highlight cursor changes appearance when on an interpolated data point.

Figure B.3.3. Display of an interpolated point

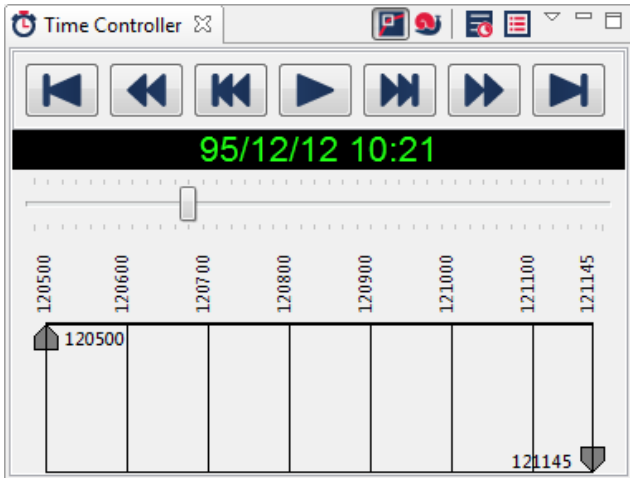
2. Controlling time

Time is managed within Debrief through the use of the Time Controller. As with all views, this is accessible through the Window > Show View menu.

2.1. The Time Controller

The Time Controller is a view that provides a number of functions, including displaying the current serial time, allowing control of that time, and providing access to a series of time-related functions in Debrief.

Figure B.3.4. Time Controller view



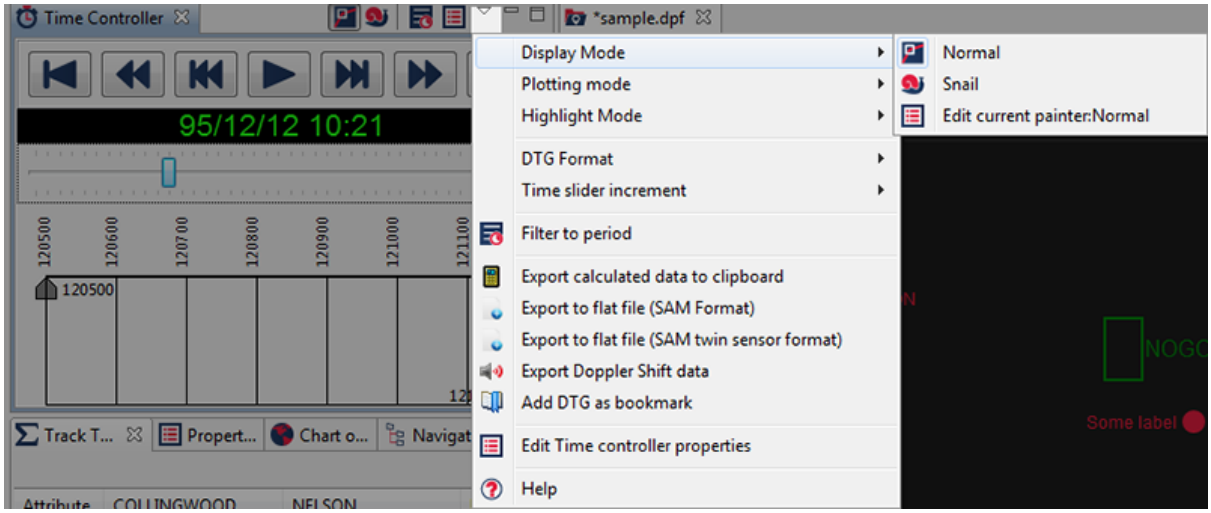
Your temporal (time-related) view of track data is dependent on three settings:

Display Mode	How the track is displayed
Plotting mode	Whether the data dynamically adjusts to follow the location/orientation of the primary track

Highlight Mode How the current position is displayed

You access these modes using the buttons on the Time Controller toolbar and the Time Controller drop-down menu.

Figure B.3.5. Time Controller menu



As you can see, the first three items on the menu allow you to select the current Display/Plotting/Highlight modes. In each sub-menu is a command to edit the currently mode. Later menu options allow you to format how information is displayed on the menu, and perform other time-related activities.

2.2. Display modes

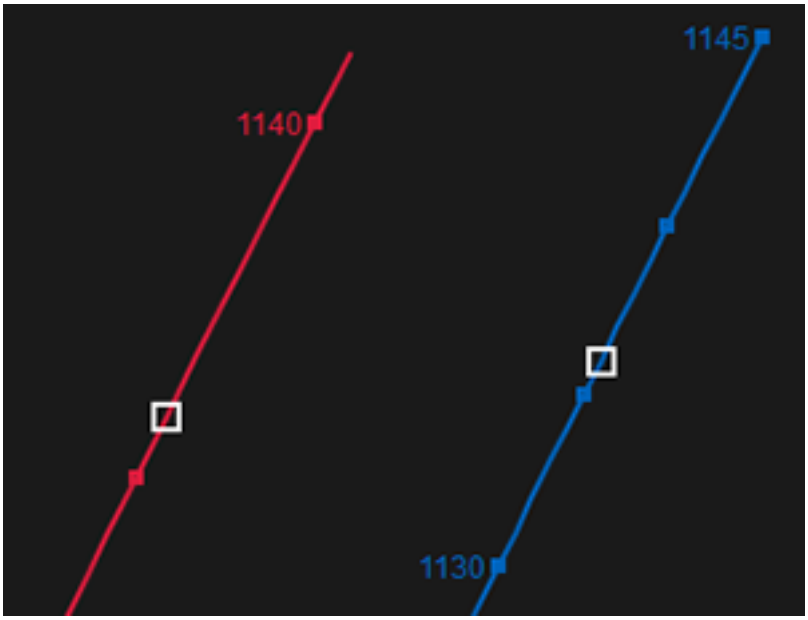
The first three icons on the Time Controller toolbar allow you to choose two combinations of plotting modes. The first two control how data is displayed: in Normal Mode, all exercise data is displayed, whereas in Snail mode, only the current position and a series of recent points are displayed (similar to a Snail with trail following behind it).

As you'd imagine, Normal Mode is the mode that is used for most analysis tasks. It's North oriented and shows all relevant data. It's also quite simple, only having two properties, both of which control the presentation of the highlight:

Color Yes, the Color of the highlight

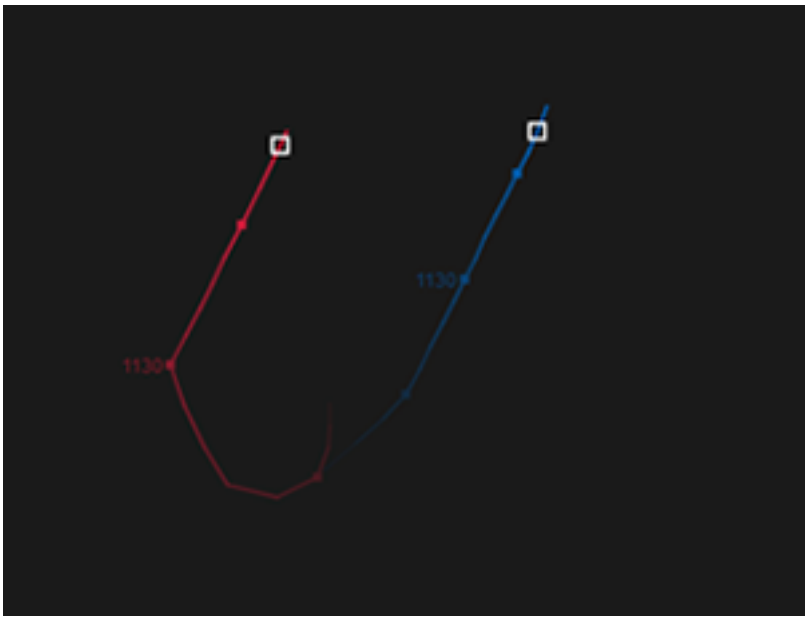
Size I know, I know, it's the size of the rectangle used to plot the highlight (measured in pixels)

Figure B.3.6. Example of a normal trail



The Snail Trail is used for specific analysis tasks when you need to concentrate on the specific activities around a certain time, without the clutter of the remaining track data.

Figure B.3.7. Example of a snail trail



The circle represents the current position, the stalk direction represents the current course, and its length gives a relative idea of the vessel speed (when compared to the length of the other vessel's stalk, *boys will be boys*). The dots trailing back from the current position are a *snail trail* of points going back in time. If you move forward and backward with the stepper control you will see these trails moving. The following properties are editable for a snail trail:

Snail trail properties

Fade Points	this will cause the points in the trail to fade away to the background colour
Link positions	this will plot a line between the points in the trail

Plot Track Name	this will plot the track name alongside the current position
Point Size	this will change the size of the points together with the thickness of the lines drawn on the plot,
Trail Length	this will change the time period covered by the trail
Vector Stretch	this will change the *amplification* applied to the speed when drawing the speed vector; very fast vessels (or weapons) will need the this stretch reduced to allow stalks of sensible length.

2.3. Plotting modes

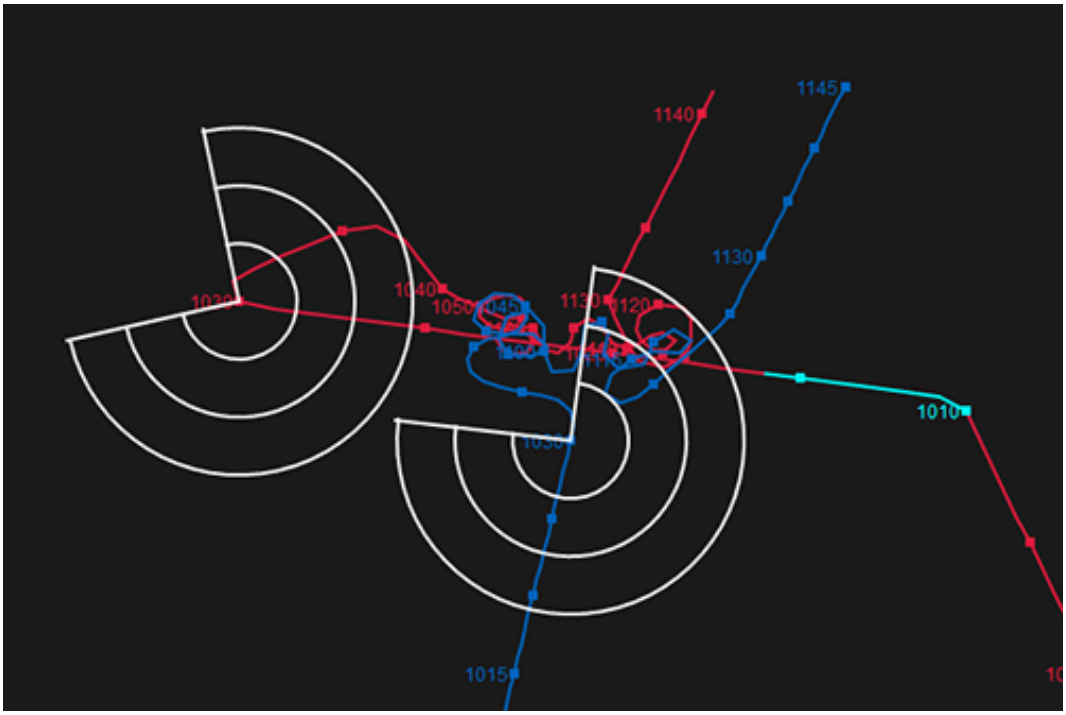
The plotting mode affects the origin and orientation of the plot. In *Normal* mode the plot viewport stays static as the time changes, but if *Primary Centred/North oriented* mode is selected, the viewport moves to track the primary participant. Beyond that, the *Primary Centred/North oriented* mode orients the plot to match the heading of the primary track. This mode is particularly useful for presenting a scenario from the perspective of the primary participant, but the quickly changing orientation can be off-putting.

Tip

Primary Centered/North Oriented mode is particularly useful for analysing one vessel trailing another. If you make the trailing vessel the primary track and the vessel being trailed the secondary track, as you step forward through the serial you will clearly be able to see the relative bearing of the contact as held by the trailing vessel.

The sample shown below gives a demonstration of the use of this mode. You can quickly see that the blue vessel is directly ahead of the red trailing vessel, and your use of the Range Ring Highlighter gives us a quick indication of range.

Figure B.3.8. Sample of primary centered/North oriented mode



2.4. Highlight modes

Three highlight modes are provided:

Default Highlight	Shows a rectangle at the current position. From the default highlight properties you are able to select the colour and size of the rectangle to plot. If a track has sensor data present, and the sensor has a non-zero array offset, then it is possible (via the PlotArrayCentre track property) to direct Debrief to plot a diagonal cross icon at the array centre.
Symbol Highlight	Shows the symbol (Chapter B.5, *Symbol sets*) currently assigned in the properties for each track. From the symbol highlight properties you are able to select the size at which the symbols should be plotted.
Highlight off	For when you don't want a highlight to be shown (such as when taking a screenshot using PrintScreen)
Range Ring Highlight	This mode shows a series of around the current location. The editable properties are listed below.

For range rings you are able to edit the following properties:

Arc Start	The start angle for the arc of coverage. The arc will be plotted relative to platform heading, with negative values plotted to the Port-Side.
Arc End	The ending angle for the arc of coverage, travelling clockwise
Color	The colour to plot the range rings
Num Rings	How many range rings to plot, uniformly distributed from the center to the outer radius
Radius	The radius of the outermost range ring
Spoke separation	The angular separation of the spokes plotted within the range rings, centered on current heading.

2.5. Time display

The green digits of the time display are tied closely to the slider-bar immediately beneath them. Dragging the slider controls the current display time together with how the current data is displayed. Other Debrief views, such as the Narrative Viewer (Chapter B.8, *Viewing narratives*) and the Time-Variable (Section 4, “Show time-related variables” graphs) update in response to time changes from the time slider.

A range of display formats are provided to make the displayed time more consistent with that in a supporting document, or of sufficient fidelity to support the current analysis.

2.6. Time slider

Beneath the time-display is the time-slider, used to quickly move through a time-period. By default the slider is of infinite resolution, stopping exactly on the second/millisecond proportionate to the position of the slider. Debrief can be configured such that the slider stops on higher resolutions by selecting the relevant increment from the Time slider increment list in the Time Controller's drop-down menu.

You can also use your mouse wheel to move forwards and backwards in time, though you have to ensure the Time Slider has focus first. Note that pressing the Shift key causes extra large steps, and the Alt key causes extra small steps.

2.7. Bookmarking

Debrief NG introduces the concept of *Bookmarks*. These represent the combination of a DTG, a remark, and the name of a plot-file, and are displayed in the Bookmarks view. With the view open you can quickly move between significant events across a number of files. Bookmarks are added by clicking on the  Add DTG as bookmark command from the Time Controller's drop-down menu.

The bookmarks view will not automatically open, but the bookmarks themselves will be present when it is. The current DTG is used as a default remark - but you'll get most mileage our of the bookmarks by describing the event that you're bookmarking.

2.8. Manipulating according to time period

The pair of Time Filter Bars at the foot of the Time Controller view allow you to set start and finish times. These times are not set in support of a single Debrief operation, but are used across a range of operations. When dragging the sliders, hold down the shift-key to move in whole segments (hours, days, as appropriate). Drag the shaded section to retain the period length but change its origin (again using the shift-key if appropriate).

Tip

On occasion it's not possible to put the time slider markers on exactly the right time value - particularly if your plot covers a long timer period. In these circumstances, if you double-click on either the start or finish arrow, their exact time is made available for editing in the Properties View (see Section 1, “Property editing”). Alternatively, hold the CTRL key down whilst clicking on a start/end marker, and a cute little window will open to let you put the time on a whole minute value.

2.8.1. Filter to time period

When the  Filter to time period radio button is depressed, changing the time of the start or end time-sliders will automatically filter the displayed data to the indicated period. In this mode, you can select a 6-hour period (for example), and drag it through the full serial time with shift-key depressed to view a moving "window" of data. In addition to filtering the visible data to the indicated period, the period covered by the time-slider is also reduced. Drag out the start/end time-sliders to return to the original time period.

2.8.2. Export to Flat File (SAM) format

This option allows you to export primary and secondary data as a flat-file format. As part of this, data can be constrained to a particular time period and can also include any visible sensor data for those tracks. This file format consists of a data file of tab-separated variables and is described further in Section 4, “Flat file format”. When you perform this export, Debrief will remember the folder and sensor type used in previous export operations.

2.8.2.1. SAM Export algorithm

This algorithm works as follows:

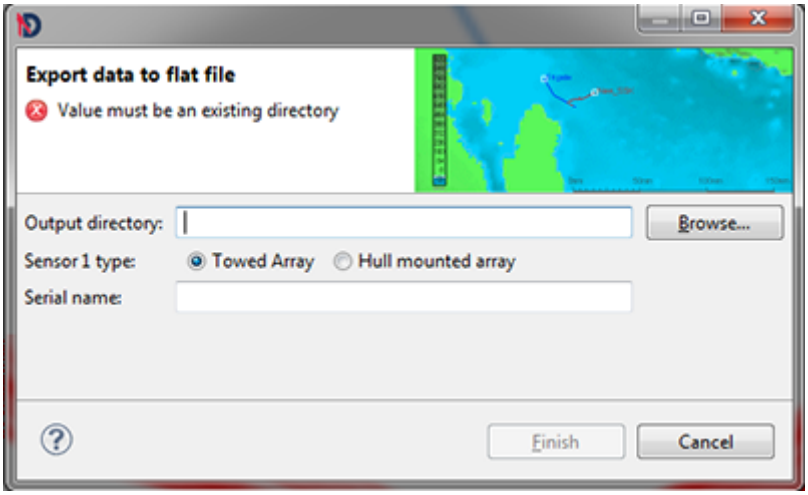
1. Check data is compliant (primary & secondary tracks, and some sensor data present for specified period
2. Write header information to file
3. Looping thorough specified period in 1 second steps:
   a. Calculate primary track location at specified time (via interpolation)
   b. Calculate sensor location at specified time (via interpolation)
   c. Calculate secondary track location at specified time (via interpolation)
   d. Output this data record calculating values from above as applicable

2.8.3. Updated Export to Flat File (SAM) format

In Spring 2012 the capability was extended to include the following enhancements:

- Include doppler calculations
- Optionally support second sensor
- Support a wider range of sensor types
- Allow the protective marking to be specified
- Allow sensor depth to be specifed (incl Aft depth for towed sensor)

Figure B.3.9. Sample improved SAM Export dialog



2.8.4. Copy to clipboard

Another operation that relies on the selected time period is exporting calculated data to the clipboard.

This operation is available from the  Time Controller's drop-down menu, and it performs a series of calculations for each data-point in the indicated time period. These calculations are then placed on the system clipboard in Comma-Separated-Variable format for reuse in other applications, Microsoft Excel, for instance.[1]

Tip

The Colour parameter shows the colour of the track point used in that calculation. On occasion analysts colour a track according to whether that participant is in contact or not.

[1] A header line is written first, indicating the contents of each column:

- Track Time(hhmmss)
- Depth(metres)
- Speed(Knots)
- Course(degs)
- Range(yards)
- Bearing(degs)
- Rel Bearing(degs - using `Relative bearing format` specified in the Window/Preferencesdialog)
- Brg Rate(deg/min)
- Color (for this track point)
- Name
- PrimaryName

The results from the primary track are listed first, which (as in the *Tote*) do not show results of calculated operations:

```
NELSON 12/Dec/95 05:00:00 000 02.00 269.7 n/a n/a n/a n/a 0500 0500
```

Then the secondary tracks are listed:

```
 BUNKUM 12/Dec/95 05:50:00 000 00.00 000.0 12381 311.0 R49.0 R0.264 F5 0550
```

Exporting the colour flag to Excel allows the post-analysis data to be filtered according to periods in contact - or any other time-dependent aspect specified by the analyst. The application of a particular colour to sections of track is performed within the Outline View.

Note

The last two columns output give identifier information: the first of these is the name of the current item in this *track*, and the second is the point in the primary *track* nearest to the current time stamp: that-is the point in the primary track which has been used for the calculated results.

2.9. VCR controls

The VCR controls allow you to move forwards and backwards in time through the plot. Looking at the order of buttons in the Time Controller screenshot above, the commands allow you to move to the beginning, move a large step backwards, move a small step backwards and repeat the last time step continuously (small or large step, backwards or forwards). The remaining buttons repeat these operations in the "forward" direction. The size of small and large steps is controlled by the Time Controller properties window, accessed by selecting Properties/Time Controller. Also available from this set of properties is the Step Interval; the time interval that Debrief waits before automatically moving forwards.

2.10. Other time operations

Beyond the operations available from the Time Controller, the time-period is used to support other Debrief operations. The most significant of these operations is when producing time variable plots (see Section 4, “Show time-related variables”). The current time period settings dictate the extent of what information is calculated for these plots.

3. Measuring range and bearing

3.1. Range bearing tool

It's worth reminding you at this point about the Range Bearing measuring tool ) which is frequently useful in analysis. The calculated range and bearing is displayed at the mid-point of the line and at the foot of the DebriefNG screen (where it remains until you make another measurement).

The default units for the range displayed are configured using the CMAP section of the Preferences dialog in the Window menu. The preferences page can also be accessed by double-clicking on the range/bearing slave display at the foot of the screen.

3.2. Earth Model

The Range Bearing calculation is performed using the algorithms in the current Earth Model, as described later in Section 1, “Range/Bearing calculations within Debrief”.

4. Show time-related variables

Time-variable plots can be opened via the Outline View. When one or more items that are candidate subjects [2] for a time-variable plot are selected in the Outline View, it adds the Show XY Plot operation to its right-click menu. After requesting the plot, the user is invited to indicate which calculation is to be plotted, and which track is to be used as the primary *track* (where relevant).The example below shows the results of a Range calculation between two tracks. Note that dragging the mouse downwards in a

[2] Items are suitable for inclusion in a time-variable plot when they have both temporal and spatial attributes, so this includes tracks, individual locations and annotations, but does not include grids, scales, or background datasets.

rectangle inside the plot zooms in on the data, dragging the mouse upwards into a rectangle zooms out on the data, and that the Fit to Window button zooms out to show all of the current data. Where data is not present, a gap is inserted into the data-line.

Figure B.3.10. View of time-variable plot

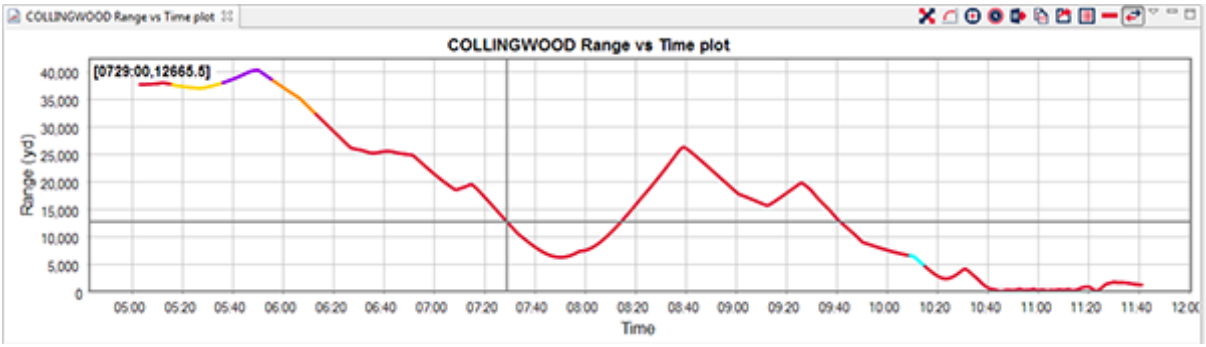
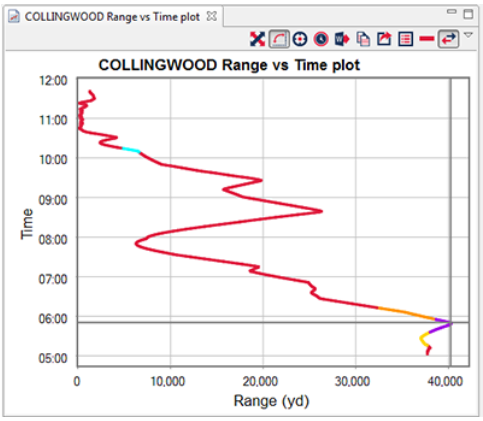


Figure B.3.11. View of time-variable plot in waterfall mode



Note

Debrief performs special processing depending on whether the selected items contain valid time data or not. In general, when an annotation (such as a label) does not contain DTG data Debrief assumes that it is valid throughout the selected time period. Debrief produces the time-variable plot according to the following tables:

Table B.3.1. Relative calculations (range, bearing, etc)

	Primary contains DTG	**Primary does not contain DTG**
Secondary contains DTG	For each point on secondary, find nearest primary point (in time), use these for calculation	For each point on secondary, use single primary point for calculation
Secondary does not contain DTG	For each point in primary, use single secondary point for calculation.	Produce single calculations at start & end of time period

Table B.3.2. Absolute calculations (range, bearing, etc)

	Calculation
Data series contains DTG	For each point on series, calculate result

	Calculation
Data series does not contain DTG	Produce single calculations at start & end of time period

The Export to WMF  button writes the current graph to a Windows Metafile in the current WMF_Directory, and the Export to Clipboard ( places a copy of the plot on the Windows clipboard for onward insertion into MS Word

The Switch Axes  button changes the orientation of the plot from its default format to a waterfall-style display with time plotted down the left-hand side.

The Grow Times  button changes the time axis to make the top of the waterfall display the current Time Controller time, with the data displayed changing dynamically with time slider..

The Configure Plot() button opens a property window (see Properties window) allowing you to control the presentation of the time-variable graph using the following properties:

Parameter	Description
DataLineWidth	The width to plot the data-lines on the graph
DateTickUnits	The interval (and format) to use on the date axis
RelativeTimes	Whether to plot absolute or relative times (used when analysing time-zero data). See Section 4, “Show time-related variables” .
ShowSymbols	Whether to show datum symbols (useful to indicate different data recording rates)
Title	The Title of the graph
X_AxisTitle	The x-axis label on the graph
Y_AxisTitle	The y-axis label on the graph
xxx Font	The font to use for the respective label
DisplayFixedDuration	When data is being displayed in *Grow Times*mode, this checkbox specifies whether time-axis should grow to continuously show all data, or whether it should just show data for a fixed time period.
FixedDuration	The period of time (duration) that the time-axis should display when in *Grow Times* mode with *DisplayFixedDuration* set to True.

4.1. Time-Variable Plot Tracker-Bar

If there are tracks on the Debrief *Tote*, a vertical bar is drawn through the plot at the correct time. If the current time on the Tote occurs before the earliest time on the Time-Variable plot then the bar is drawn at the minimum value, and drawn over the maximum value if the current time on the Tote occurs after the latest time value. The bar redraws itself automatically as the time in the Tote changes.

4.2. Analysing time-zero data in time-variable plot

By default the time-variable plot shows absolute times (e.g. `12:34.00`). Some forms of analysis conducted with Debrief rely on the use of relative times. To show these relative times on the lower

axis of the time-variable plot, select RelativeTimes from the time-variable plot property window. Once this mode is selected, times will be shown relative to the last time-zero value set. By default, the times are shown in a normal `HH:mm:ss` format, but the DateTickUnits drop-down list contains a number of display formats which specifically support presentation of elapsed times.

Chapter B.4. Exporting Data

1. Exporting images

1.1. Resizing, ready for export

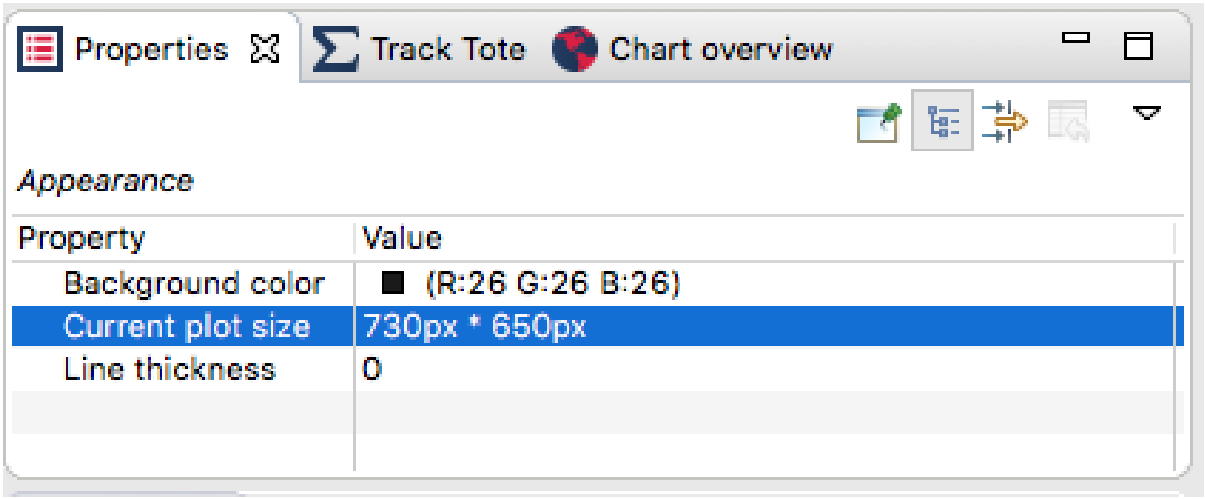
It's common for Debrief users to capture still images or captured video screen recordings when they wish to share analysis findings with another audience, either within a document or a presentation.

When Debrief plots are exported in WMF format, elements of the captured image resize, to make sure they're still legible. But, when bitmap images get rescaled, they can either become pixelated (blocky) or some pixels can disappear. When a line is only 1 pixel wide there is a chance that parts of it will disappear.

The solution to this is to adjust the Debrief plot to the same size as the target device. For example, if an analyst knows that his screencast is going to be shown on a projector with a resolution of 1024px by 768px, then if the plot is resized to exactly that size she will know the video will be shown at the greatest possible level of detail.

Sadly, the technologies involved in Debrief don't (currently) allow a user to type the width and height. But, Debrief is able to inform the analyst of the current size as the plot editor is resized. To do this, double-click on an empty area of a Debrief plot. The plot background settings will be shown in the properties editor. One of the properties is a read-only one, titled Dimension. As the editor is resized (by dragging the vertical/horizontal sashes that separate the Debrief panels), this property will give live feedback on the current size.

Figure B.4.1. Property for current plot size



1.2. Exporting to WMF

In addition to conducting tactical analysis, Debrief allows the analyst user to create plots for insertion into Microsoft Windows applications; particularly Word.

To do this load the data into Debrief and format the plot, adding scales, grids, coastlines, and annotations as desired.

To export this image as a Windows Metafile (WMF), select the Export WMF () operation from the File menu. Alternatively to copy the image to the clipboard (still as a WMF) select Copy from the Edit menu whilst the Debrief plot is active.

This will place the image in a time-stamped file named `d3_minute_second.WMF`, located in the directory indicated in the Debrief properties file. If the location is not specified in the properties file, the `WMF` image will be created in the Debrief installation directory.

Note: exporting vector plots via the clipboard is limited to MS Windows PCs. But, an equivalent capability allows you to export the vector plot inside an RTF wrapper via the clipboard for pasteing into any Word Processor (on OSX or Linux). The shortcut for this is <shift><control><c>.

2. Exporting engagements to PowerPoint

2.1. Introduction

It's quite common for analysts to include Debrief screenshots in PowerPoint presentations. On occasion an analyst will record a screencast of a running engagement, to give a dynamic presentation that conveys more information. But, these video recordings can be very large (10's of Mb), and while it isn't easy to email them around, it's practically impossible to get them to a ship.

An alternate mechanism is to represent the vessel tracks as animated objects in PowrPoint. These consume very little disk space (typically a couple of kilobytes) - and can be added to an existing PowerPoint briefing at negligible cost incurred.

The engagement is exported to a specically configured donor file, formatted as described in Section 5, “Master template for export scenario to PowerPoint”

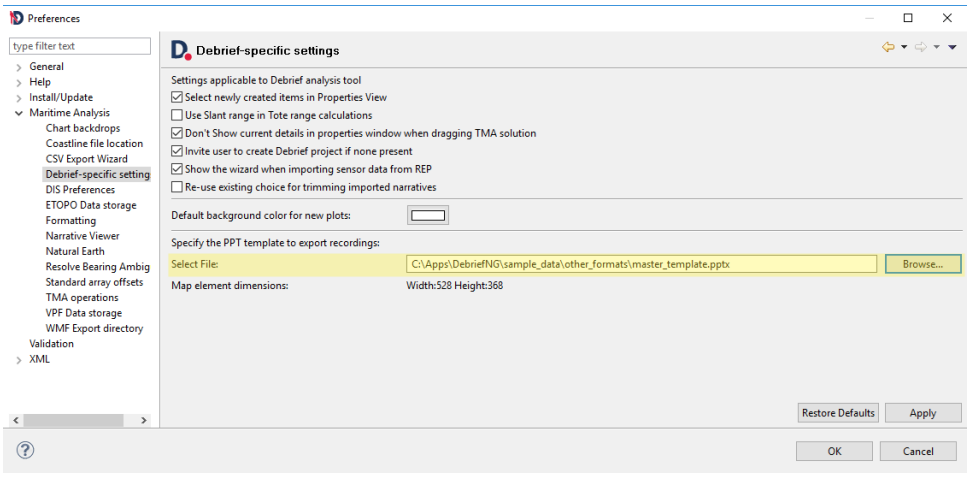
2.2. Export process

2.2.1. Assigning donor template

Debrief injects the exported data is into a donor template. You may use one provided by your organisation, or you can use the sample one provided with Debrief, titled `master_template.pptx`, in the `sample_data/other_formats` folder.

Assign the donor template by selecting it in the Debrief preferences window:

Figure B.4.2. Specifying donor file location



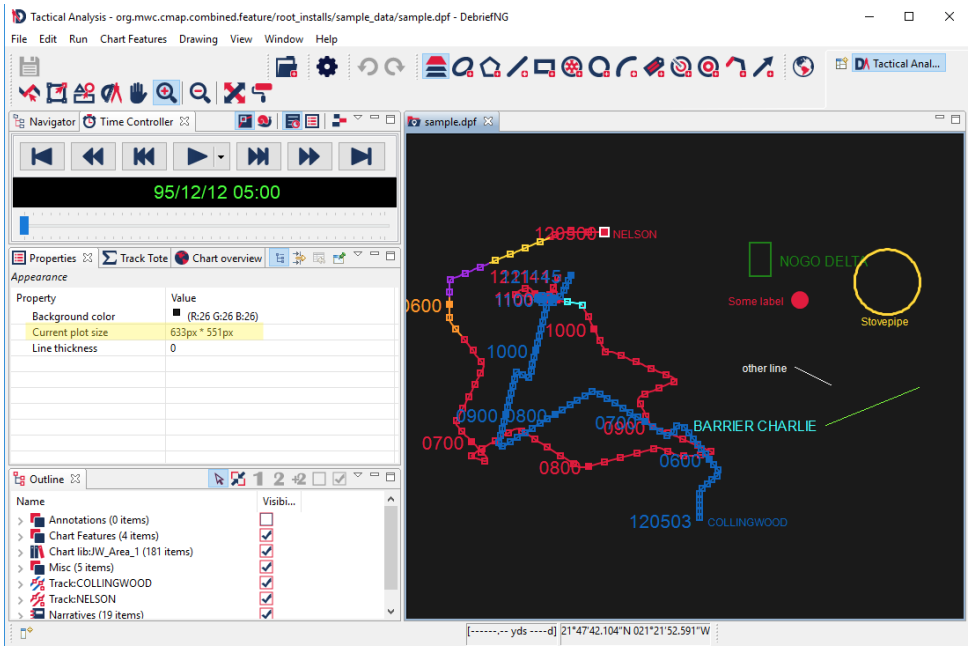
Note that the preference window also displays the size of the map frame that it found in the donor template.

2.2.2. Resizing plot area

To get an optimal quality of export, the Debrief plot area should be resized to the same number of screen pixels as the map element in the donor template (note the the element dimensions displayed on the preference window above). This resizing is performed by you dynamically resizing. Start by

loading your Debrief plot, then double-clicking into any area of blank background. This will open the plot properties in the Properties View. Then drag the editor (and/or) Debrief until the plot size displayed matches that in the donor template.

Figure B.4.3. Controlling Debrief plot size



2.2.3. Collate data

Next step is to prepare your plot, as if you were about to give a live debrief, using Debrief's debrief capability. Zoom in on the area of interest, and ensure the relevant tracks are visible throughout the entire period of interest.

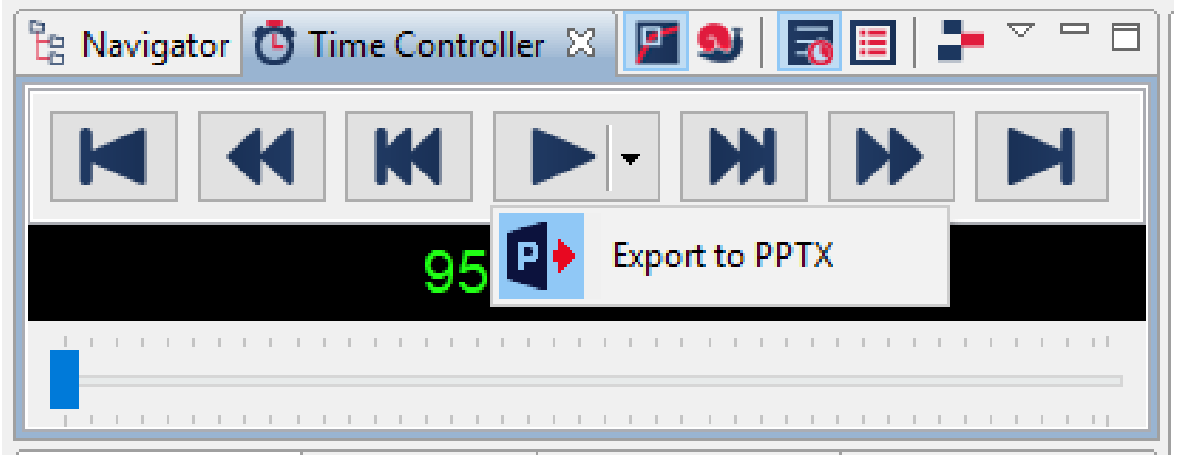
Also configure the Time Controller replay settings, so that the scenario moves forward at time steps relevant to the data of interest. These settings are explained more in ???.

It may even be worth pressing Play and performing a trial run through the engagement, as a final check that the data (and time controller properties) are configured in the optimal way to convey your message.

2.2.4. Perform export

Once you're happy with your data, n export is performed using these steps:

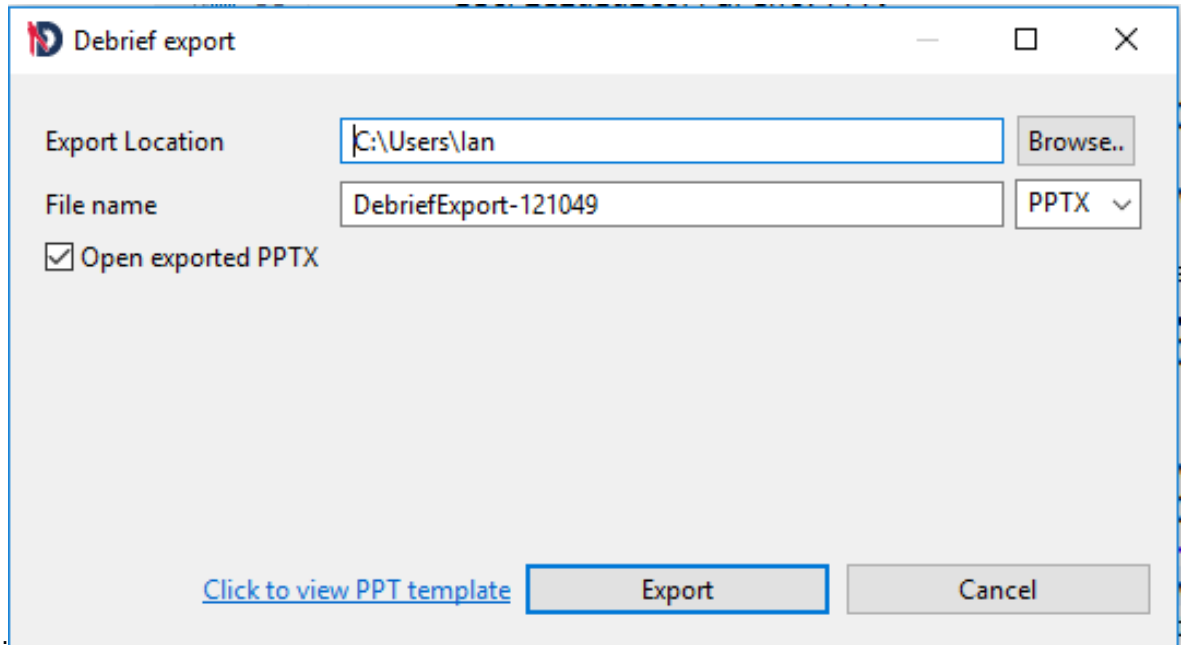
1. Click on the down-arrow next to the Play button, and select Export to PPTX.



2. Collapse the drop-down menu, if necessary

3. Click on Play, and let the scenario unfold. Note that the time controller indicates that it is recording the engagement.

4. When you click on Pause, the export to PPT dialog will open.

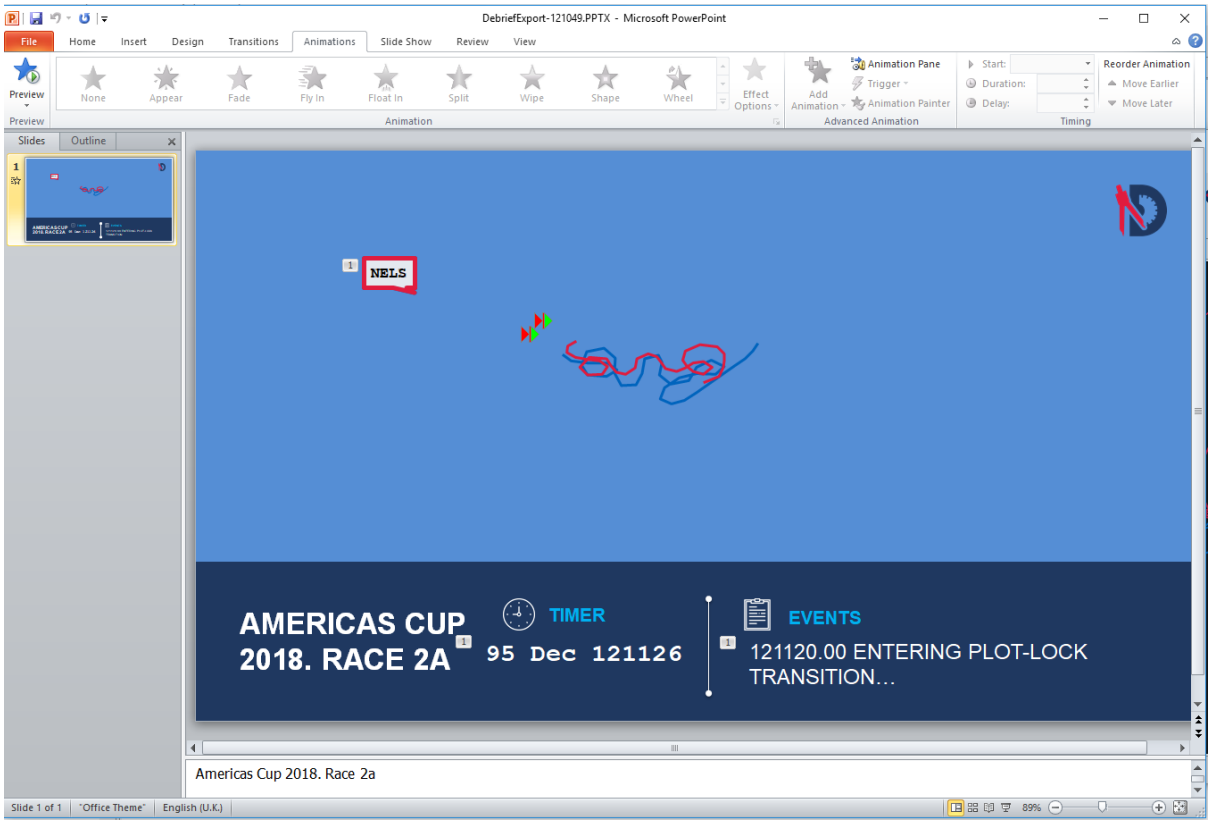


5. Verify that the export location is correct, and give a filename for the file that gets producued.

6. Finally, click on Export. If you clicked on Open exported PPTX, the presentation will open.

Once you have opened the exported file, you can switch to the Animations tab, then press Preview to watch/verify the dynamict tracks.

Figure B.4.4. Sample of exported scenario



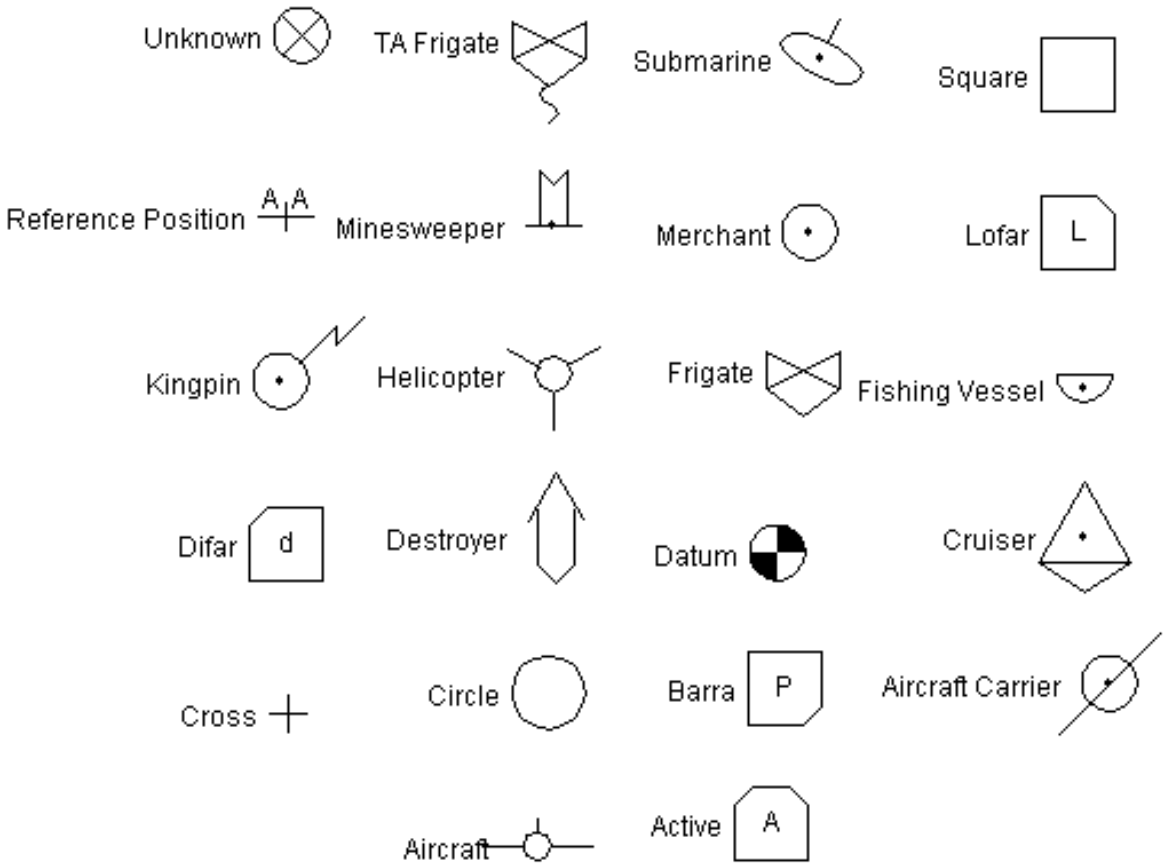
Chapter B.5. Symbol sets

1. Introduction

A set of symbols is provided within Debrief. The symbols can either be attached to a text label, or used to highlight a current vessel location when stepping through tracks.

Twenty-two Symbols are provided in 3 sizes. When the 2 smaller sizes are shown, any internal characters (as used in the sonar buoys) are not plotted.

Figure B.5.1. Symbols provided by Debrief



Debrief also contains a series of symbols that are recorded in SVG files. These files are plain SVG drawings that Debrief reads from file, and then renders at the required location. Some of the symbols can be assigned directly from their REP file symbology characters, as described in Table D.1.3, “Debrief symbology symbol codes”. The other, non-indexed symbols can be specified from REP file using the extended symbology attributes described in Table D.1.5, “Debrief extended symbology data fields”

Figure B.5.2. Indexed SVG symbols

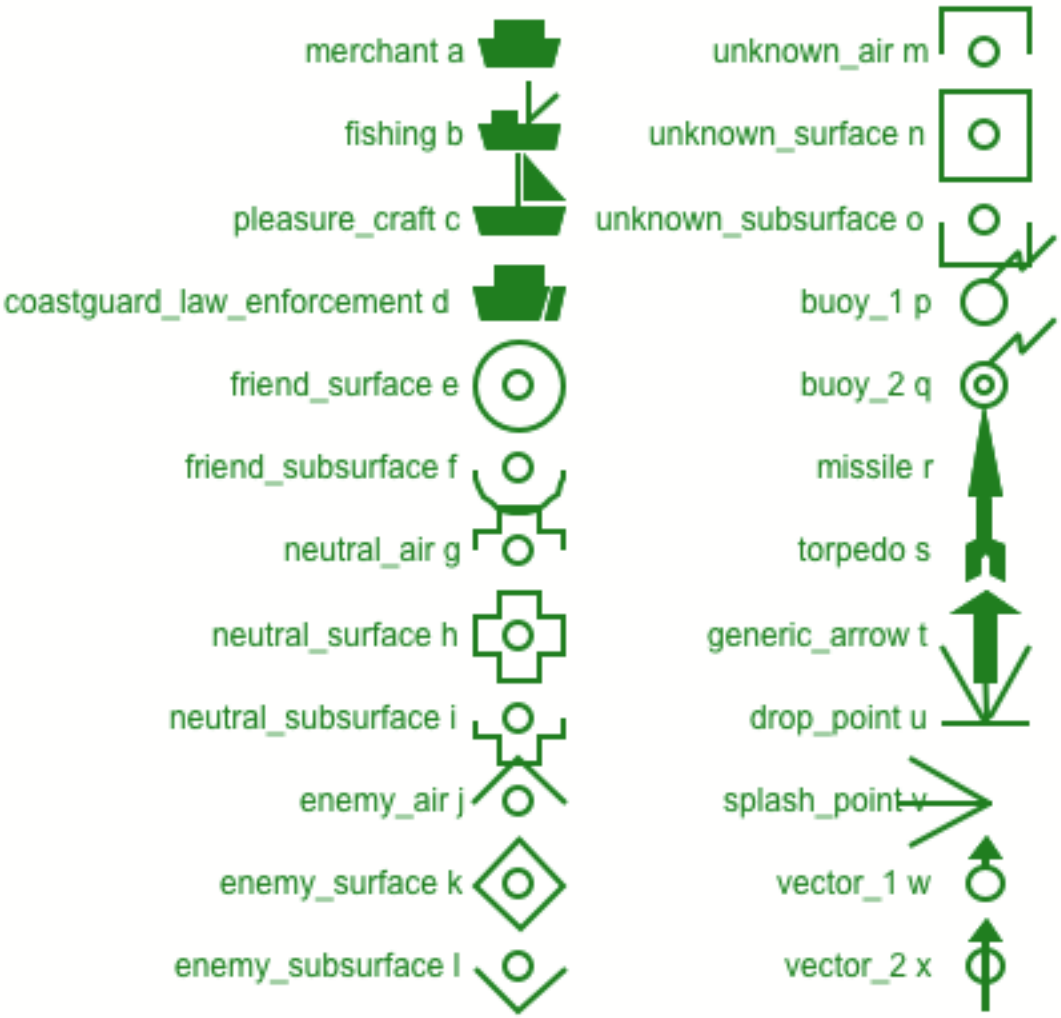
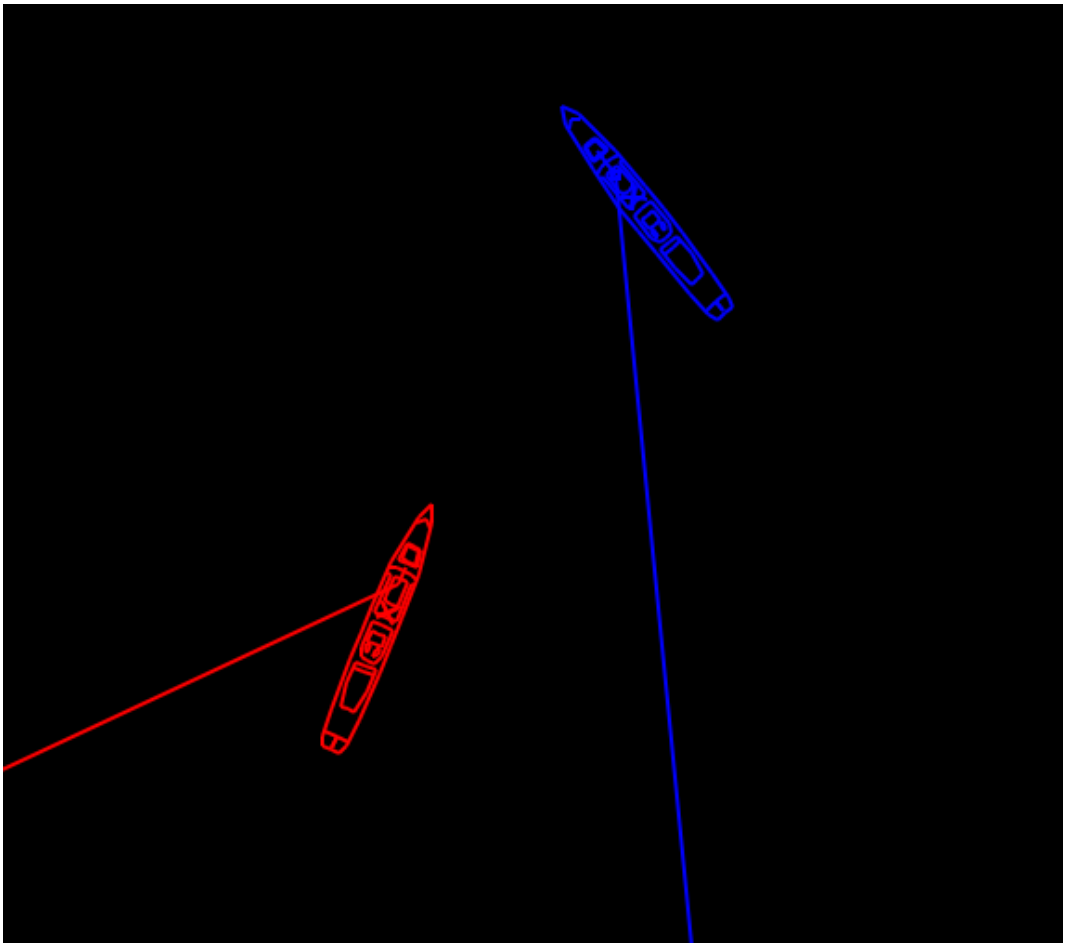


Figure B.5.3. Non-indexed SVG symbols

Note

The scaled symbols are not shown at fixed screen sizes, but as representative sizes of the subject vessel - so they resize as you zoom in on a plot. It is also possible to specify the length and width of the subject platform for a realistically sized vehicle representation.

Figure B.5.4. Scaled vessel symbols in Debrief



1.1. Use in labels

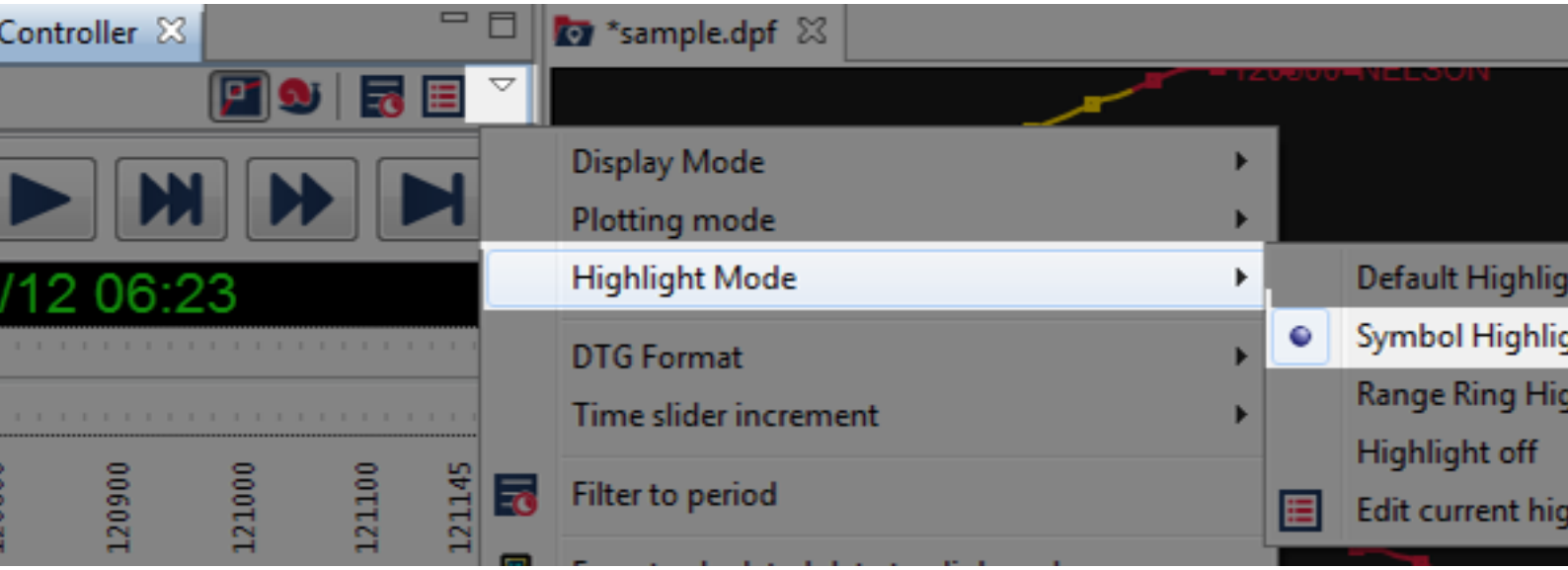
Any label placed on the plot has a symbol type property (although the symbol itself may not be visible). The symbol can be changed through the properties window, or by right-clicking on the label on the plot.

1.2. Use in tracks

The final area where symbols are used is when stepping through tracks. Each vessel track has a symbol type property which is shown when the plot has SymbolHighlighter selected as the current highlighter.

To access this property, select the Properties button on the *Time Controller* (the drop-down button on the view), and change to Symbol Highlight in Plotting Mode. As shown below:

Figure B.5.5. Highlighting Symbols in Tracks



Tip

By switching to Snail mode and reducing the reducing the Trail Length to zero, a GOP-type plot can be shown.

Chapter B.6. External datasets

1. Natural Earth data

1.1. Introduction

Figure B.6.1. Natural Earth logo

“Natural Earth is a public domain map dataset available at 1:10m, 1:50m, and 1:110 million scales. Featuring tightly integrated vector and raster data, with Natural Earth you can make a variety of visually pleasing, well-crafted maps with cartography or GIS software.”`http://www.naturalearthdata.com/`

Whilst Debrief users have enjoyed VPF and ETOPO reference datasets for vector and raster backdrops (resp) for over 10 years,these have now been superceded by the Open Source *Natural Earth* dataset. Natural Earth vectored data doesn't come with any styling, so the team employed a very clever professional cartographer (Julie at http://jewelcartografx.com/) to produce a set of themes to apply to the various Natural Earth feature types.

The Natural Earth download size has also be reduced by focussing on a specific set of features of interest to the maritime domain. Lastly, the data-points in this set of data-files have been subtlely trimmed to remove points at the poles, to make them suitable for Debrief's mercator projection. The data-point trimming was performed with the most excellent QGis application (http://qgis.org).

The detailed Natural Earth data is around 80Mb in size, so it's too large to include with Debrief. So, only they very highest scale, lowest resolution data is included with Debrief. This gives a world coastline and named countries. If you require more detailed data (which you almost certainly will), just download the more detailed dataset and unpack it on your disk - as detailed below.

1.2. Using Natural Earth

1.2.1. Downloading

The customized set of Natural Earth data can be downloaded from this GitHub repository: https://github.com/debrief/NaturalEarth. You don't need to download the individual files, just click on the Download Zip button to the bottom-right of the page. The zip file is about 25Mb. This will download a zip-file of the data (obviously).

Once you've downloaded the data, unzip it into a nice safe folder on your system. It should expand to around 80Mb.

1.2.2. Configuration

Ok, you've got the data on your system. Now you need to tell Debrief where it's stored. You do this from Debrief's Natural Earth preferences page. So, go into Window/Preferences and the dialog will

open. Then select Natural Earth from beneath Maritime Analysis. In that form, just browse to the unzipped Natural Earth folder.

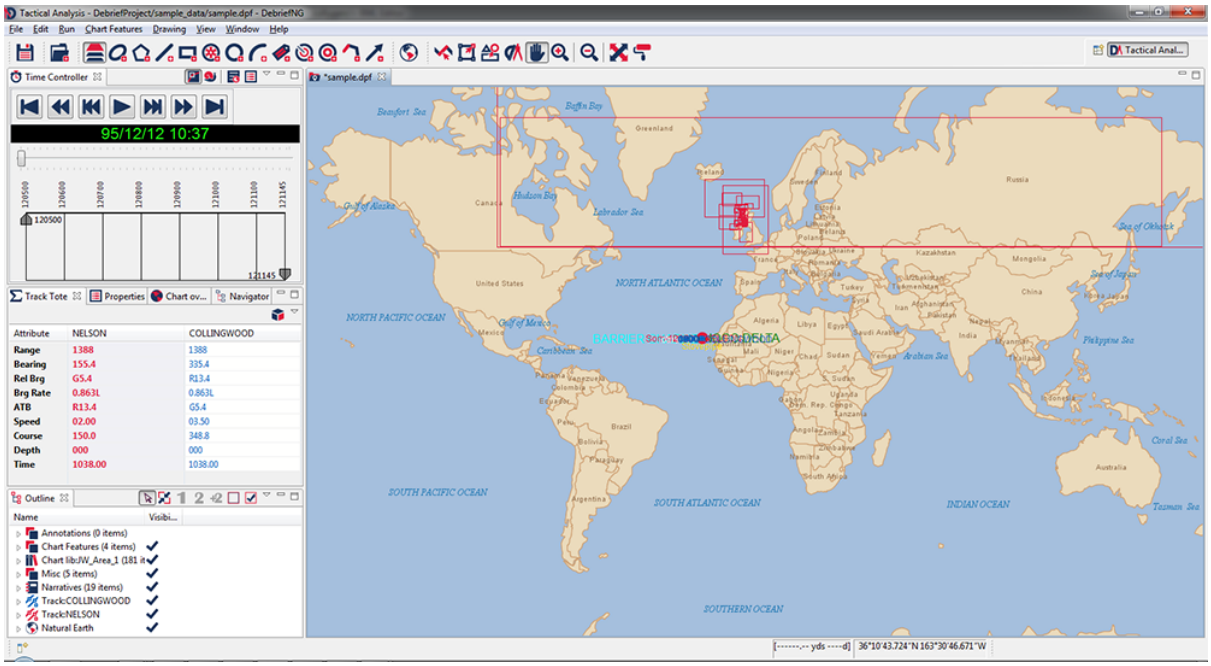
1.2.3. Usage

Unlike the previous external reference datasets, Natural Earth is kid's play. You really do just have to click on the Natural Earth icon, from the Chart Features menu, or the main application toolbar: 

The Natural Earth layer doesn't have any configurable properties - all of that has been done for you already.

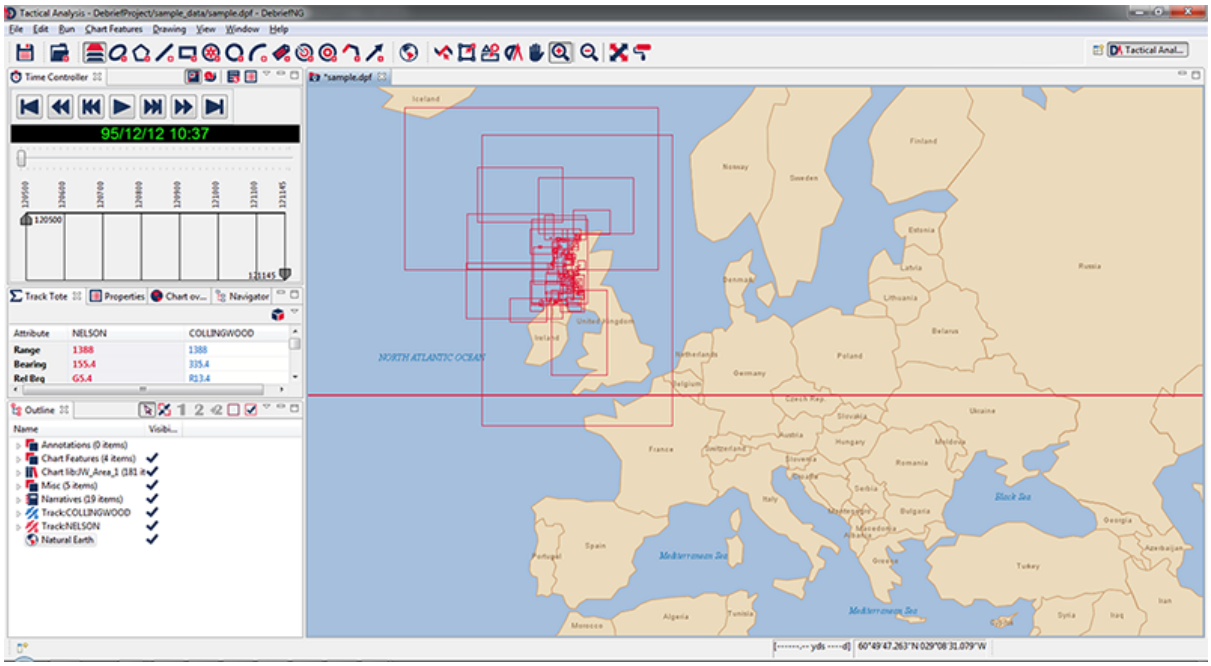
If you haven't downloaded and configured the detailed dataset, or if you're zoomed out a long, long way, you will see the 110M resolution dataset:

Figure B.6.2. Natural Earth in use (110M resolution)



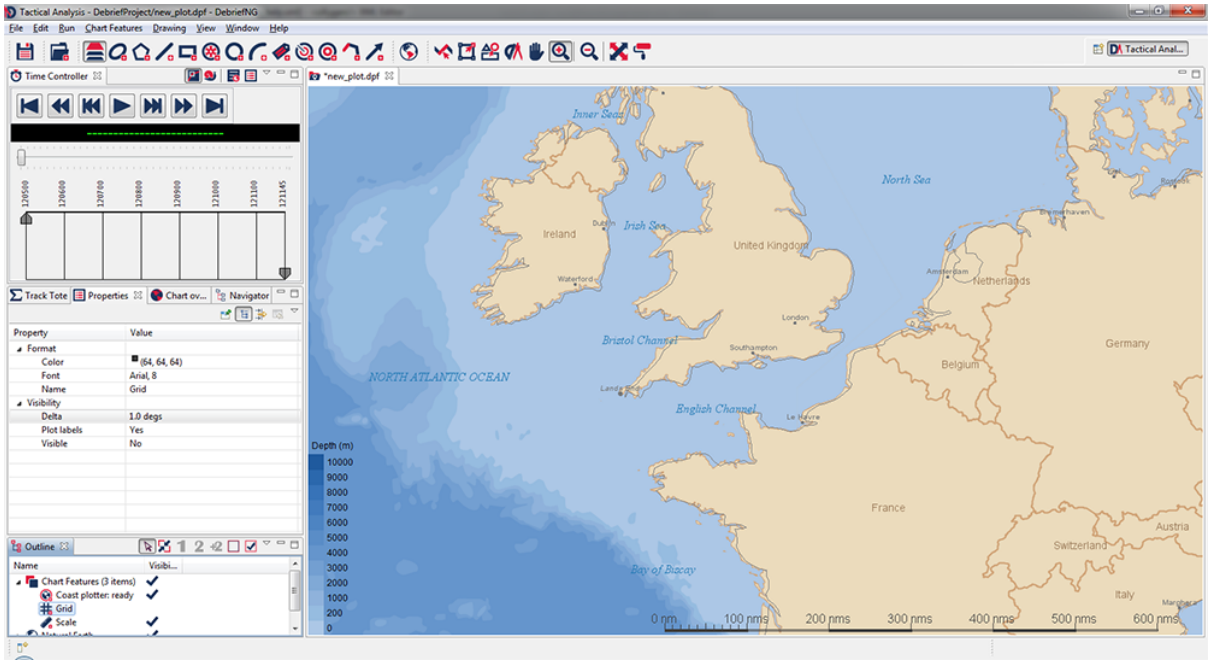
Zoom in a little more and you will start to see names of significant places and a more detailed coastline, from the 50M resolution dataset:

Figure B.6.3. Natural Earth in use (50M resolution)



Finally, once you're zoomed in close you will see bathymetric data, plus a lot more placenames:

Figure B.6.4. Natural Earth in use (10M resolution)



2. VPF data

2.1. Introduction

The *VPF* standard provides Debrief users with the ability to display the contents of a number of different vectored databases. VPF(TM) is a Registered Trademark of US National Imagery and Mapping Agency.

One of the databases, the Vector Map (VMap) Level O is an unclassified global database of many features, including coastlines, depth and elevation contours. This is the favoured VPF database for

use in Debrief. But note that Debrief makes no assumptions of or optimisations for the VMap data - any VPF data source can be used. Debrief can also import Digital Nautical Chart (DNC) databases. The Digital Nautical Chart is produced by the National Imagery and Mapping Agency (NIMA) and is an unclassified, vector-based digital database containing maritime significant features essential for safe marine navigation.

Here is an introduction to VPF from the US National Imagery and Mapping Agency's web site (http://www.nima.mil):

> The Vector Product Format (*VPF*) is a standard format, structure, and organization for large geographic databases that are based on a georelational data model and are intended for direct use. VPF is designed to be compatible with a wide variety of applications and products. VPF allows application software to read data directly from computer-readable media without prior conversion to an intermediate form. VPF uses tables and indexes that permit direct access by spatial location and thematic content and is designed to be used with any digital geographic data in vector format that can be represented using nodes, edges, and faces.
>
> 
>
> VPF defines the format of data objects, and the georelational data model provides a data organization within which software can manipulate the VPF data objects. A Product Specification corresponding to a specific database product determines the precise contents of feature tables and their relationships in the database. In this context, each separate product or application is defined by a Product Specification implemented by using VPF structures.
>
> VPF data is stored in a structure described in the Military Standard, Vector Product Format, MIL-STD-2407 . The Standard specifies the structure for directories, tables, table columns, table join relationships, and media exchange conventions for all VPF data. The data structure itself can be thought of as a template or skeleton within which the geospatial features and metadata are stored. While the Standard describes the structure, it does not describe the contents of a set of VPF data; this is the role of "VPF Product Specifications."

2.2. Introduction to VMAP Level 0

Vector Map (VMap) Level 0 is an updated and improved version of the National Imagery and Mapping Agency's (NIMA) Digital Chart of the World (DCW).

The primary source for the database is the 1:1,000,000 scale Operational Navigation Chart (ONC) series co-produced by the military mapping authorities of Australia, Canada, United Kingdom, and the United States. The complete database is available on a set of four CD-ROMs and contains more than 1,800 megabytes of vector data organized into 10 thematic layers. VMap Level 0 includes major road and rail networks, hydrologic drainage systems, utility networks (cross-country pipelines and communication lines), major airports, elevation contours, coastlines, international boundaries and populated places. VMap Level 0 includes an index of geographic names to aid in locating areas of interest. VMap Level 0 is accessible directly from the CD-ROM or can be transferred to a hard drive and used in many geographic information system (GIS) applications.

2.3. Obtaining VPF data

VPF data can be obtained from a number of different sources, although the main resource is the NIMA [http://www.nima.mil] web-site itself.

In Autumn 2015 VPF data was available from the US National Geospatial-Intelligence Agency (NGA), at these locations:

- `http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0noa.tar.gz` [http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0noa.tar.gz]
- `http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0soa.tar.gz` [http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0soa.tar.gz]
- `http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0eur.tar.gz`
- `http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0sas.tar.gz` [http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0sas.tar.gz]

In the UK, the Director General of Military Survey (DG Mil Survey) were able to provide the VMap database (charged to a Royal Navy UIN).

If/when you find other sources please provide feedback and this page will be updated.

Searching the Internet will undoubtedly provide other sources for the information - it's probably worth shopping around. Since the data is in the public domain agencies only have to charge a "handling fee".

2.4. Storing VPF data

Debrief can read VPF data directly from CD-Rom but copying it to your hard disk provides the following advantages:

- It runs many times (>10) quicker
- It allows you to view the contents of multiple CD-Roms simultaneously (the VMap level 0 data comes on 4 CDs so unless you have 4 CD-readers in your machine this is the only way to get global coverage)

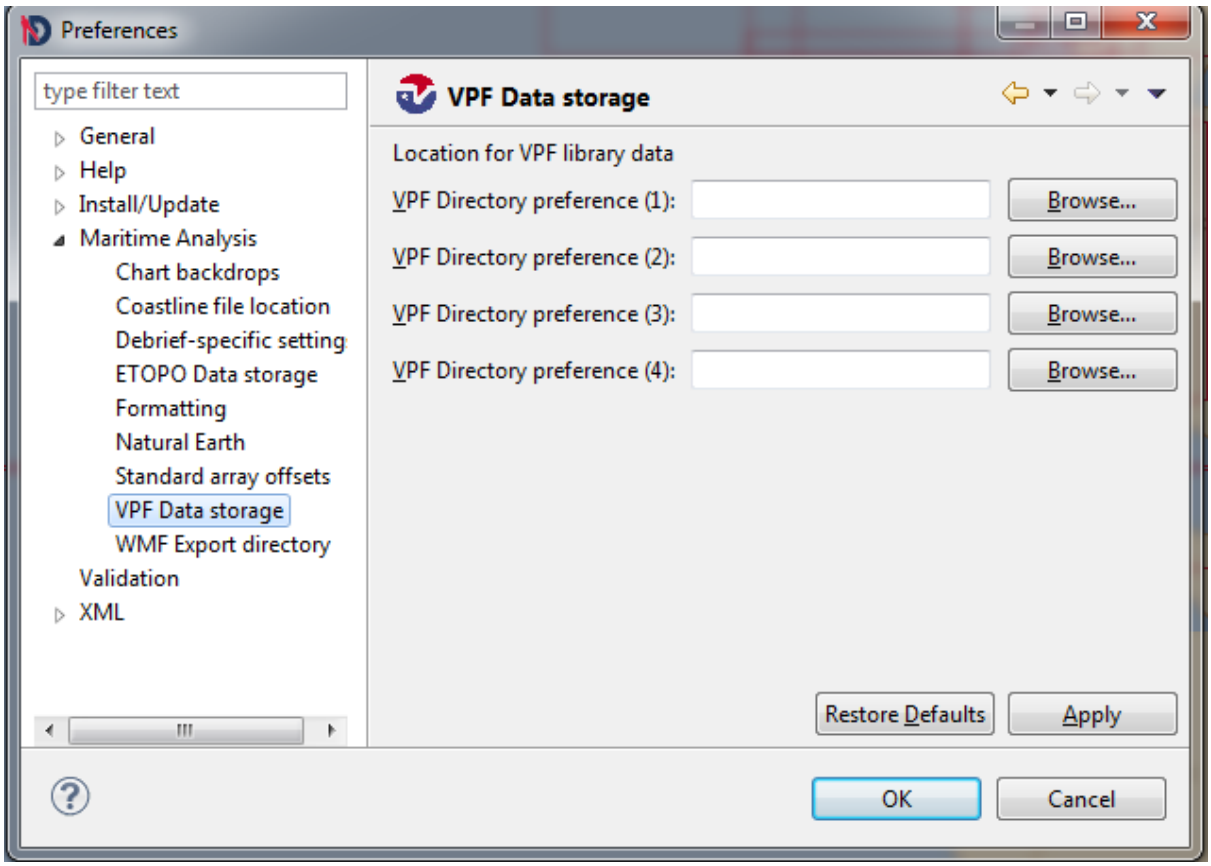
So, on the assumption that you do want to copy the data to your hard disk, here is a strategy for keeping the data tidy:

1. Create a folder in the top level of your hard disk called `VPF`
2. Now insert the first VPF CD-Rom and view it through your file manager (e.g. Windows Explorer). You need to find which directory contains the actual data, so have a look in each of the top level directories for a directory which contains the files `DHT` and `LAT`. Note, for VMap data this is a directory named `VMAPLV0`. When you've found this directory copy it across to the `VPF` directory you created on your hard disk.
3. So for VMap data, your hard disk you should have a directory named `VPF`, containing your first set of VMap data in a directory named `VMAPLV0`.
4. You will be copying in a number of data sets with this directory name, so rename `VMAPLV0` to the name of the current dataset (such as `NOAMER, SASAUS, EURNASIA or SOAMAFR`).
5. Next, swap your CD for the next CD in the series, copy the directory of interest across to the `VPF` directory on you hard disk, and rename it.
6. Once you've repeated this process for all of the CDs for the current database it's time to configure Debrief to load the data.

2.5. Configuring Debrief to read VPF data

Debrief determines where to find the *VPF* data using the VPF Data storage tab of the Preferences dialog from the Window menu. Indicate the four data-file locations using the file-browser buttons.

Figure B.6.5. VPF preferences



2.6. Thank heavens for Open Source

As many of you are probably aware, Debrief is an Open Source application, which means that anybody is free to copy, change, and re-use the Source Code for Debrief, provided they meet the terms of the Debrief license.

The VPF plotting libraries we're using in Debrief are taken from another Open Source application, OpenMap from BBN Technologies.

The OpenMap application can be found at http://www.openmap.net. Debrief makes no modifications to the OpenMap application code.

3. Viewing VPF data

3.1. Introduction

Debrief does not pay any attention to *VPF* data until it loads a plot which requires the data, or until the user requests that VPF layers be added.

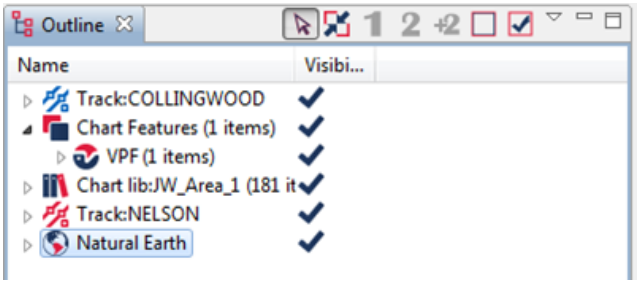
VPF layers are loaded into a plot using the "Create VPF Layers" button: 

Yes, it's a shrunken version of NIMA's VPF logo. If VPF layers have been loaded into Debrief, their details are stored in the plot file (i.e., whether they are switched on or off, and their colour). Note that the data itself is not stored in the plot file, just the names of the layers you're using (so there's no significant increase in file size).

3.2. Creating VPF layers

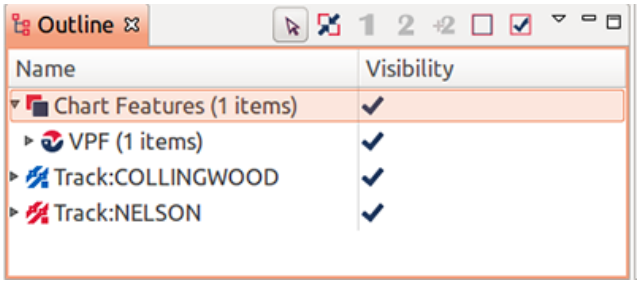
Once loaded into a plot configure the VPF data by switching to the Layer Manager. In there, open the Chart Features layer, to see the new VPF layer.

Figure B.6.6. VPF Data in Outline View



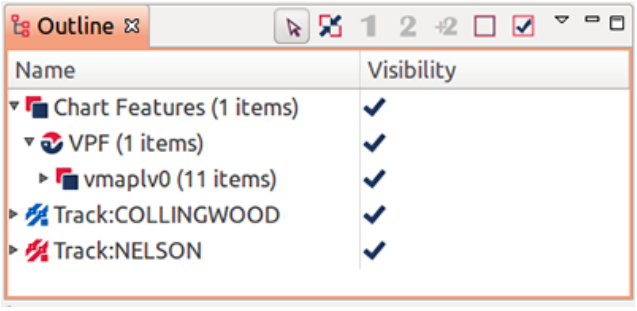
Click on the VPF layer to open it up, showing which VPF databases are currently loaded. In this example you can see that I have the VMap Level 0 (vmaplv0) data together with some Digital Nautical Chart (DNC) data. Each layer shows how many items are on that layer, and the empty check box shows that each layer is not currently visible.

Figure B.6.7. Layers within Chart Features



You are going to switch on coastline data first, so click open up the vmaplv0 database by clicking on the expand folder symbol. You will see the list of VMap themes listed. Later on, feel free to open them up and view the data they contain, but for now you will concentrate on the coastlines. The order of the themes may be different on your machine, this is of no concern.

Figure B.6.8. List of VPF layers



Next, open up the Boundaries theme to see which Boundaries can be plotted by VMap.

Figure B.6.9. Boundaries theme

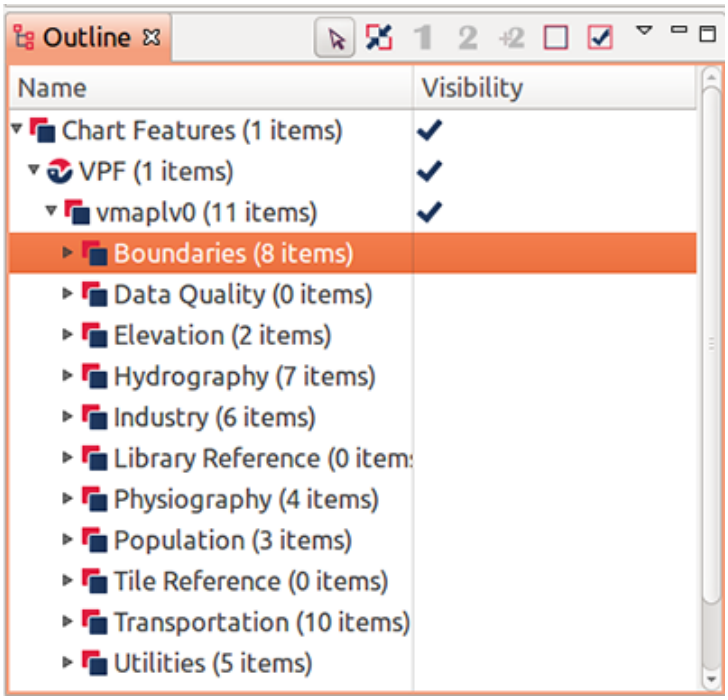
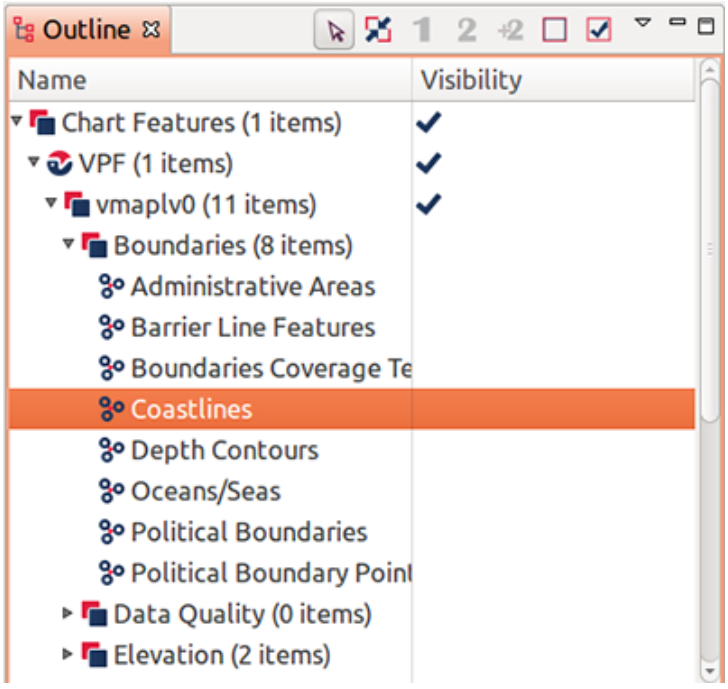


Figure B.6.10. Coastlines feature

Now you are going to configure the plotting of coastlines.



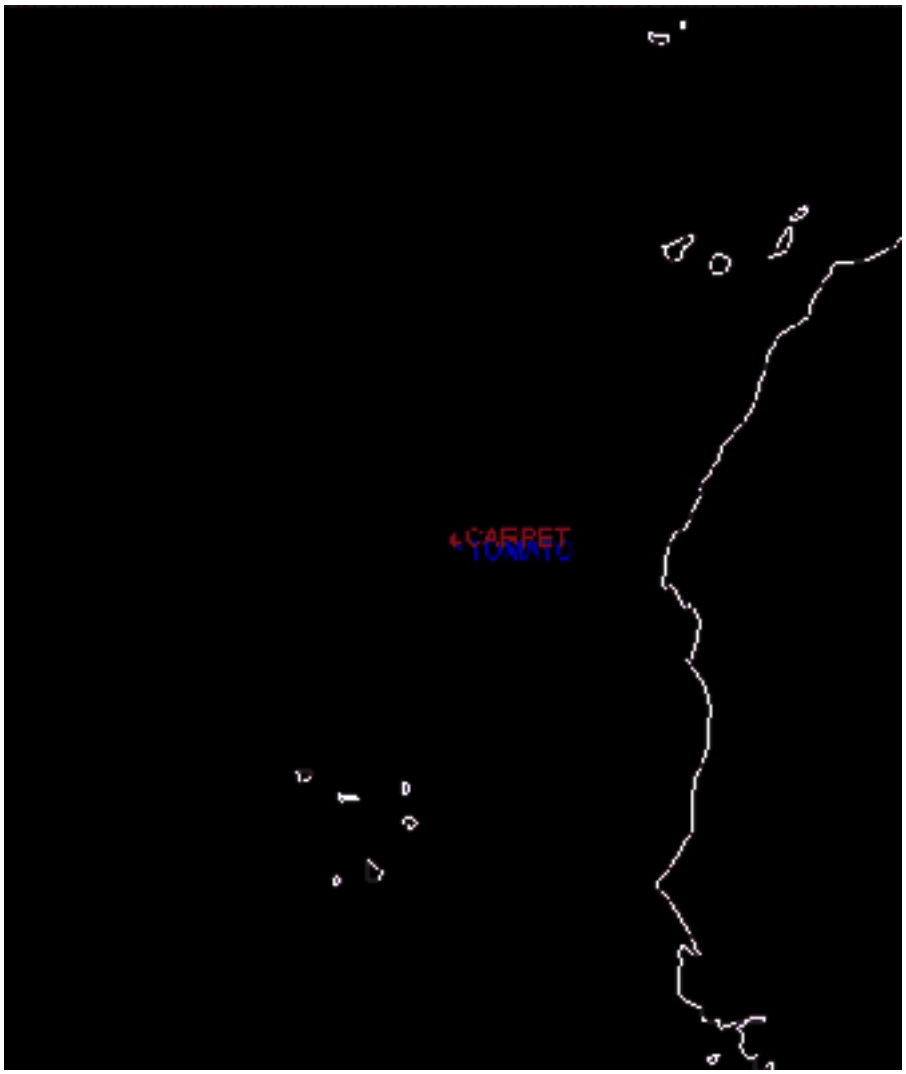
Right-click on the Coastlines entry, and from the Coastlines drop-down list, select Color, and then Light Grey. Note that White is the default colour for all VPF features.

Next, right-click on the coastlines entry again, and make the coastline Visible.

Whilst we've made the Coastlines visible, the layers above it aren't, so now you make its parents visible. Right-click on the Boundaries layer and make it visible. Next right-click on the vmaplv0 layer to make that too visible.

Nothing has yet appeared on the plot, but that is because you are not current over any coastline. Now zoom out 6 times until the plot looks like that below:

Figure B.6.11. First coastline



Now you can see your new coastline of Africa.

Select the Pan control from the Plot toolbar: 

Now drag the plot southwards to move the view up towards Europe. Keep dragging until you have a clear view of a more familiar coastline, that of the British Isles. From this view you will add depth contours.

Figure B.6.12. Coastline of British Isles

Right-click on the Depth Contours entry in the Boundaries layer and switch its colour to Grey. Also switch the Contours entry to Visible.

Depth contours will now appear:

Figure B.6.13. Coastline of British Isles with depth contours

Note

Note, the VMap CD-ROM gives the following description with relation to depth contours:

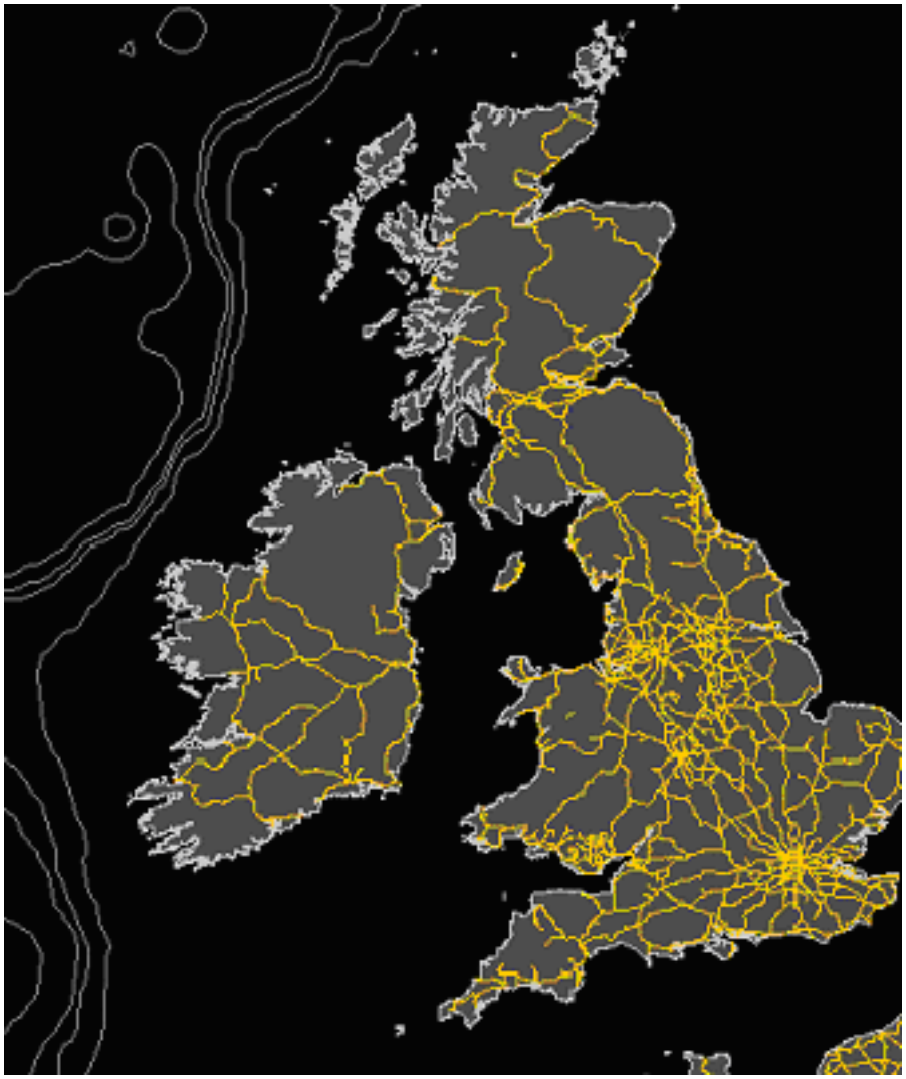
> Depth contours (BE015) were derived from digital bathymetric data provided by Defense Mapping Agency. Depths are expressed in the following intervals: 200, 600, 1000. 2000, 4000, 6000, and 8000 meters.

Feel free to zoom in on the data to see just how detailed the data gets (around Cowes on the Isle of Wight sections of coastline of only 200 yards length are clearly visible).

Also zoom out to view the whole of the British Isles and experiment with switching on features in other layers.

The plot below shows UK railways (from the Transportation Layer), and UK is shaded in by making the Administrative Areas (from the Boundaries Layer) visible.

Figure B.6.14. Railways of British Isles



Alternatively zoom and pan across to your home, see the accuracy of your part of the coastline.

3.3. Storing VPF settings

As explained earlier, when you save the *VPF* plot your VPF settings get saved with it. So have a go at saving your current view, exiting/re-starting Debrief then re-loading it. You should find yourself seeing the same view which you saved.

3.4. VPF best practice

There's no doubting the volume of data available through VPF, and the VMap Level 0 database in particular. The next level down from VMap Level 0 is VMap Level 1; which contains approximately 10 times more detailed information.

How much of it is of use is more subjective though, so it's best to lead your strategy for its use by your analysis requirements.

4. Configuring VPF defaults

4.1. The problem

Yes, VPF and VMap in particular provide a great volume of information which may help with analysis and will certainly improve the quality of "overview" images in reports.

With this great volume comes a management overhead however. You could have constrained Debrief to only use VMap data, and then only load a sub-set of it (coastlines & depth contours). Instead, however, it loads the full set.

How can you make it easier to use?

Read on.

Warning

You'll need to know how to use a text editor and have a rough familiarity with *XML* or HTML text.

4.2. The solution

How does it Debrief handle VPF data?

- When you save a plot with VPF data loaded, Debrief stores the names of all of the layers loaded, together with the colours and visibility of any features on them.
- When you re-load an existing session with VPF data, Debrief reads what layers the users wants and only loads those layers.

Accordingly, you could create a datafile containing only the layers you want, and drop this file into Debrief sessions. This should give us a simplified set of VPF layers.

When you drag/drop a *DPF* file into an existing session, the Layers in the *DPF* file are copied into the existing session, although the projection, Tote, and GUI parameters are ignored.

4.3. How to do it - 1

So what you do is open a new Debrief session and load the smallest REP file you have into it - (just to give your data an "origin").

Then add your VPF layers using the Create VPF Layers button on the Chart Features toolbar. Customise these layers so that the layers/features of interest are made visible and set to your desired colours.

Finally save the view to a plot file in an easily accessible location - call it `default_layers.dpf`

4.4. How to do it - 2

You are now going to edit this file to remove all unnecessary details except for your *layer*s.

Open a text editor (such as Notepad in MS Windows), and load your new .DPF file into it.

The contents of the file should be something like that shown below:

```
      <?xml version="1.0" encoding="UTF-8"?>

      <plot Created="Thu Jul 19 15:04:41 GMT+01:00 2001" Name="Debrief
 Plot">
      <session>
      <layers>
      <layer Name="Chart Features" Visible="TRUE">
      <vpf_database Visible="TRUE">
      <vpf_library Visible="TRUE" Name="vmaplv0">
      <vpf_coverage Visible="FALSE" Type="ind" Description="Industry">
      <vpf_feature Visible="FALSE" Type="extracta"
 Description="Extraction
        Areas">
```

```
      <colour Value="WHITE" />
      </vpf_feature>
      <vpf_feature Visible="FALSE" Type="storagep"
 Description="Storage Point
        Features">
      <colour Value="WHITE" />
      </vpf_feature>
      <vpf_feature Visible="FALSE" Type="indtxt" Description="Industry
 Coverage
        Text">
      <colour Value="WHITE" />
      </vpf_feature>
```

..... and so on, until

```
      <layer Name="Annotations" Visible="TRUE">
      <rectangle Label=" trial" LabelLocation="Left">
      <colour Value="YELLOW" />
      <fontcolour Value="YELLOW" />
      <font Family="Sans Serif" Size="12" Bold="FALSE"
 Italic="FALSE" />
      <tl>
      <shortLocation Lat="-9.2166417" Long="156.2783694"
 Depth="0.000" />
      </tl>
      </rectangle>
      </layer>
      </layers>
      <projection Type="Flat" Border="1.050" Relative="FALSE">
      <tl>
      <shortLocation Lat="60.6482349" Long="-15.0669609"
 Depth="0.000" />
      </tl>
      <br>
      <shortLocation Lat="47.9097909" Long="3.0063981" Depth="0.000" /
>
      </br>
      </projection>
      <gui>
      <tote />
      <component Type="Stepper">
      <property Name="AutoStep" Value="1000" />
      <property Name="Highlighter" Value="Default Highlight" />
      <property Name="StepLarge" Value="600000.000" />
      <property Name="CurrentTime" Value="691231 235959.999" />
      </component>
      </gui>
      </session>
      <details Text="Saved with Debrief version dated 19 Jul 01
 12:44" />
      </plot>
```

4.5. How to do it - 3

As you can see above, the file starts with an entry beginning with the characters <?xml. This line indicates that you are handling *XML* data.

Next you have a series of lines of data, which start with <plot> and end with </plot>. This format is similar to that found in HTML, and indicates that this file contains details of plot. Inside the <plot> you can see a <session>, which in turn contains a <layers> object, containing a series of <layer> items, followed by a <projection>, and a <gui>.

What you are going to do is strip out everything except the Layers object, and then thin this out.

So, move the cursor to the <layers> line. Delete everything before this (except for the <?xml version="1.0" encoding="UTF-8"?> line.

Now move down to the </layers> line, and delete everything from it to the end of the file.

As you look at the layers, you can see that each layer has a name (Chart Features is the first one in the example above). It is the Chart Features layer which you want to keep, so navigate down to the next Layer (which may have a track name), and delete that and all others (down to, but not including the </layers> line which marks the end of the data).

Within the Chart Features layer you can see an entry named <vpf_database> - this is the vpf data. Inside the vpf_database are <vpf_library> entries, one for each library loaded (probably just VMaplv0 in your instance).

Inside the <vpf_library> are a series of <vpf_coverage> entries. It is these which you will thin out. Work down through them deleting any you don't want. In your instance you only want to keep the boundaries data, so you will delete all other coverages. So, select blocks of text beginning with <vpf_coverage> and ending with </vpf_coverage> and delete those you don't want.

Finally, delete any <vpf_features> you don't want.

I've deleted all those I don't want, leaving the text below:

```
      <?xml version="1.0" encoding="UTF-8"?>
      <layers>
      <layer Name="Chart Features" Visible="TRUE">
      <vpf_database Visible="TRUE">
      <vpf_library Visible="TRUE" Name="vmaplv0">
      <vpf_coverage Visible="TRUE" Type="bnd"
 Description="Boundaries">
      <vpf_feature Visible="TRUE" Type="oceansea" Description="Oceans/
Seas">
      <colour Value="BLUE" />
      </vpf_feature>
      <vpf_feature Visible="FALSE" Type="polbndl"
 Description="Political
        Boundaries">
      <colour Value="WHITE" />
      </vpf_feature>
      <vpf_feature Visible="TRUE" Type="polbnda"
 Description="Administrative
        Areas">
      <colour Value="WHITE" />
      </vpf_feature>
      <vpf_feature Visible="FALSE" Type="depthl" Description="Depth
 Contours">
      <colour Value="WHITE" />
      </vpf_feature>
      <vpf_feature Visible="TRUE" Type="coastl"
 Description="Coastlines">
      <colour Value="CYAN" />
```

```
</vpf_feature>
</vpf_coverage>
</vpf_library>
</vpf_database>
</layer>
</layers>
```

4.6. At last!

Now, you can drag and drop this file into any Debrief session to instantly give you your "favourite" set of layers.

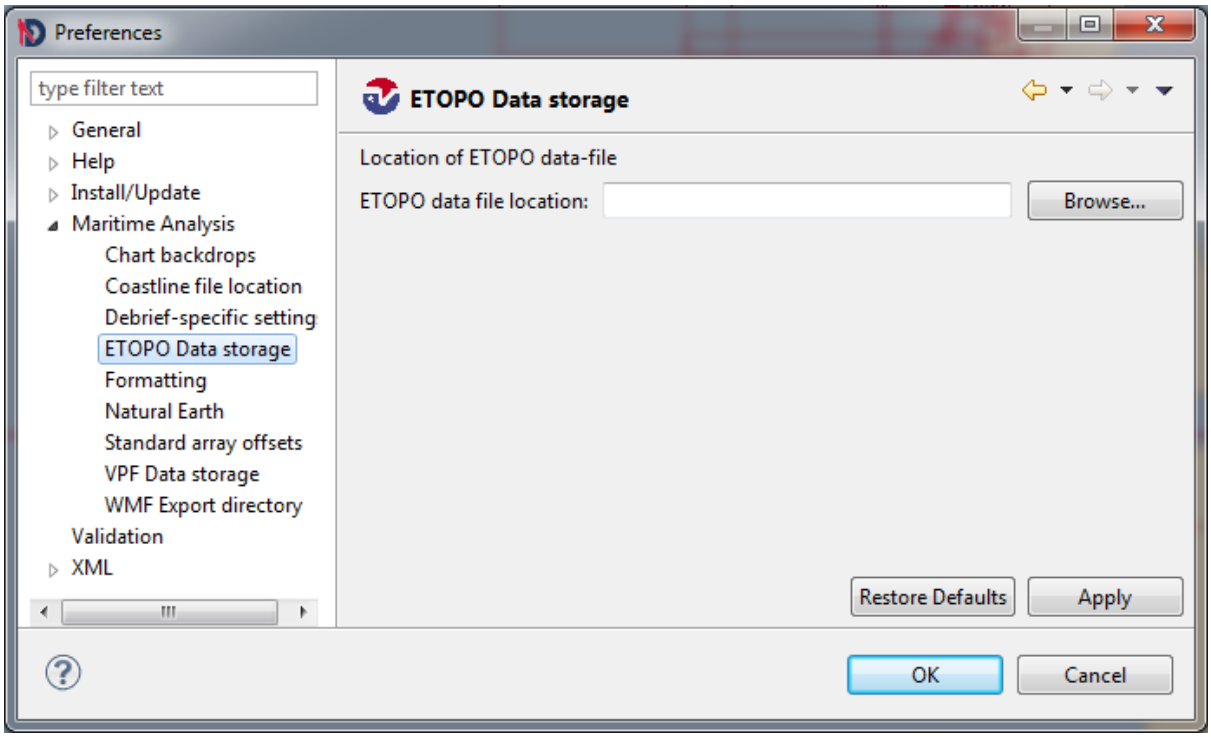
To test it, follow the steps described above (or copy the text I've just given you into a blank text file and save it as `default_layers.dpf`). Then create a new Debrief session, load one of your demo tracks (`boat1.rep`), then drag in `default_layers.dpf`. Zoom out, and have a look at the lovely, pre-formatted data.

5. ETOPO Data

5.1. Configuring Debrief to read ETOPO data

Debrief determines where to find the *ETOPO* data using the ETOPO Data storage tab of the Preferences dialog from the Window menu. Indicate the location of your ETOPO.RAW datafile using the file-browser button.

Figure B.6.15. ETOPO preferences



5.2. Background to ETOPO data

Debrief 2002 added the capability to view gridded bathymetric data, provided through the ETOPO dataset. The ETOPO dataset and its use is explained in more detail in Section 2.8, “ETOPO gridded bathy”.

Note

The 'ETOPO-5' data set is originally from the U.S. National Geophysical Data Center (NGDC) in Boulder, Colorado (USA), and represents the "best" available digital terrain values as integrated from existing five and ten-minute digital sources. The data set has elevation values spaced at every five-minute latitude/longitude crossing on the global grid (approx. nine km.-sq. spatial resolution, or 12 x 12 pixels/degree), and a one-meter contour interval. Bathymetric values are included in this data set, starting at approximately 10,000 meters below sea level, while the elevation values extend up to heights of approximately 8,000 meters above sea level. Some original sources of the data used include the U. S. Defense Mapping Agency for the conterminous USA, Japan and Western Europe; the Australian Bureau of Mineral Resources, and the New Zealand Department of Scientific and Industrial Research.

GRID has reformatted the original NGDC data file to place the origin at 180 degrees West longitude, instead of at 0 degrees Greenwich Meridian. The 'ETOPO-5' data file has 2160 records of data with a length of 8640 bytes each: the size of the data array is 2160 lines by 4320 elements, but this is a l6-bit or two bytes per element data file. The origin of the data file is at 90 degrees North latitude and 180 West longitude, and it extends to 90 degrees South latitude and 180 degrees East longitude. The data file comprises 18.66 Megabytes. The version of this data file at GRID has been discovered to contain two records (lines) of flawed data values; that is, portions of lines 2055 and 2056, beginning at the Weddell Sea north of Antarctica and continuing eastward. GRID is currently waiting for a response from the data supplier (NGDC) before attempting any replacement of what appear to be anomalous data values.

There are two useful references for the 'ETOPO-5' data set. These are: "Edwards, Margaret Helen, 1986. Digital Image Processing of Local and Global Bathymetric Data. Master's Thesis. Department of Earth and Planetary Sciences, Washington Univ., St. Louis, Missouri, USA, 106 p." and "Haxby, W. F. et al., 1983. Digital Images of Combined Oceanic and Continental Data Sets and their Use in Tectonic Studies. EOS Transactions of the American Physical Union, vol. 64, no. 52, pp. 995-1004."

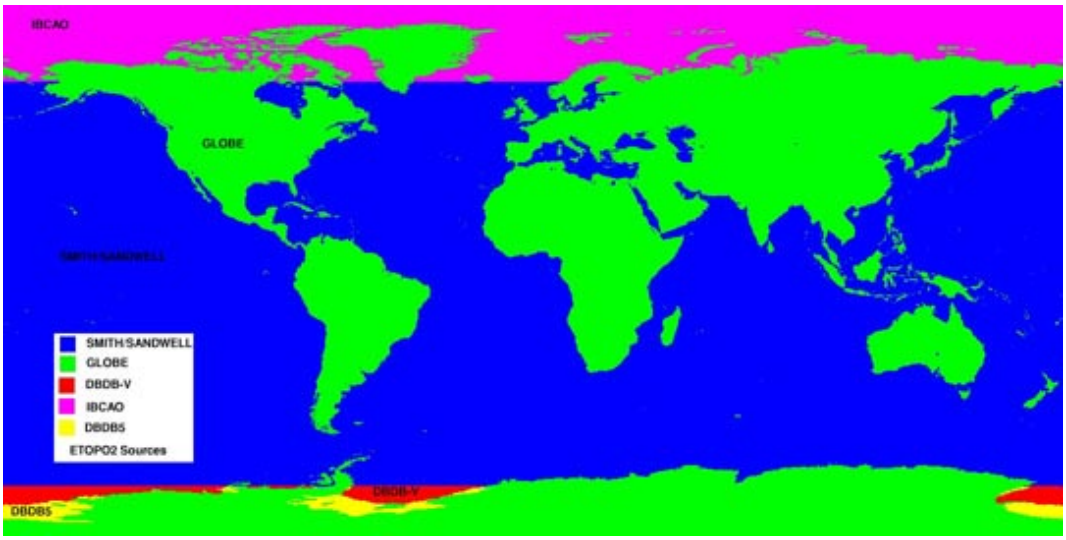
Note

In November 2002 the capability to read the US NOAA 'ETOPO-2' data set was been added to Debrief. The ETOPO-2 dataset may be purchased from the NOAA [http://www.ngdc.noaa.gov/mgg/gdas/gx_announce.Html], and is delivered on CD-ROM. The ETOPO-2 CD-ROM itself contains several datasets. The Dataset supported by Debrief is the `ETOPO2.RAW` file containing data in the *big-endian* format. This file must be copied into an `plugins\org.mwc.cmap.static_resources_1.0.0\data` sub-directory of the Debrief installation. This 2-minute dataset offers more than 6 times the detail of the ETOPO-5 dataset. This 2-minute data is derived from the following sources:

- The seafloor data between latitudes 64 North and 72 South is from the work of *Smith and Sandwell, 1997*. These data were obtained from satellite altimetry observations combined with shipboard echo-sounding measurements.
- Seafloor data southward of 72 South are from the US Naval Oceanographic (NAVOCEANO) DBDBV version 4.1 at 5 minute spacing.
- Seafloor data northward from 64 North are from the new International Bathymetric Chart of the Arctic Ocean (IBCAO) Version 1.
- Land data is from the GLOBE Project, an internationally designed, and independently peer-reviewed global digital elevation model (DEM).

These sources are summarised below:

Figure B.6.16. ETOPO2 data sources



This information has been taken from the NOAA Web-site [http://www.ngdc.noaa.gov/mgg/fliers/01mgg04.html].

The following options are provided for plotting ETOPO data:

Color	The color to plot the key itself.
Key location	This option determines where (and whether) to show the key for the depth data. Note that the ETOPO data will always appear behind other data, so it may be necessary to experiment with the key location.
Show land	Whether to plot land as land, or as very shallow water. Plotting the land as very shallow water is the favoured way of combining ETOPO data with VPF data. The VPF coastline data is of much higher resolution, differences being highlighted when ETOPO land is shown.
Visible	Whether to plot the ETOPO data.

The ETOPO-2 dataset provides a few more customizable attributes, which allow you to choose between enhanced performance over appearance in the plot:

Bathy Res	This is the size of each bathy square to be plotted in screen pixels. A resolution (Res) of 1 pixel provides the most detailed bathymetric plot, though it takes the longest time. Frequently it is possible to increase the bathy-res without any visible degradation in the data displayed - whilst providing performance benefits
Bathy Visible	This flag indicates whether the bathymetric plot should be shown or not (sometimes the set of contours are sufficient).
Contour Depths	Into this box provide a comma-separated list of depth contours (in metres) to plot. The values should expressed in ascending order as whole numbers. If Debrief encounters a problem whilst reading the values it will return to the last valid set of depths
Contour Grid Interval	The contour plotting algorithm is very processor-intensive, calculating the contours when zoomed out over a large area can take many minutes. The contour plotting algorithm inspects every depth datum in the data area whilst finding contours. The speed of calculation can be

	improved by directing the algorithm to skip a number of depth values before performing the next calculation. This may make the contours themselves very slightly more blocky, though still usable.
Contour Optimise Grid	Setting this switch instructs Debrief to automatically increase the Contour Grid Interval as you zoom from the plot. This prevents the exponential increase in time take to conduct the contouring, ensuring that no more than 10000 calculations are performed, and when zooming in it ensures that where applicable at least 2000 calculations are performed.
Contours Visible	This flag indicates whether the contours should be plotted or not - not showing contours substantially speeds up the redraw time.

Tip

The ETOPO-5 dataset uses a significant amount of memory on your PC, typically 30Mb, though this only gets loaded once per Debrief session, however many plots are loaded. Writing a WMF file with ETOPO data visible requires even more memory, and can cause Debrief to hang or crash. This problem can be overcome by following the advice described in Section 1.6, “Starting the program”.

The ETOPO-2 dataset is many times larger than ETOPO-5, thus is not read into memory but accessed on the fly. For this reason it does not consume as much memory, but does require a fast PC to produce acceptable screen updates.

Chapter B.7. Exercise planning

1. Introduction

In Spring 2012 Debrief received extensions in order to support exercise planning. An analyst is able to generate tracks within Debrief, replay those tracks to verify time/spatial constaints, and finally export the run-plan to MS Word via the clipboard.

The exercise planning tracks are generated in a new track-like entity. This new entity has an origin and a start date/time. These provide a quick way of changing the location and/or the start time for a set of vessel runs. The legs generated for a track represent straight line legs for a platform. Each leg is stored according to a demanded course and a leg-length. Leg-length is specified according to one of these calculation models:

Range-Speed	The time travelled along the leg is deterlimed by the time leg length (range) / speed
Time-Speed	The leg length is determined by the distance travelled in the specified time at the specified speed
Range-Time	The speed along the leg is determined by the speed necessary to travel the specified distance in the specified time.

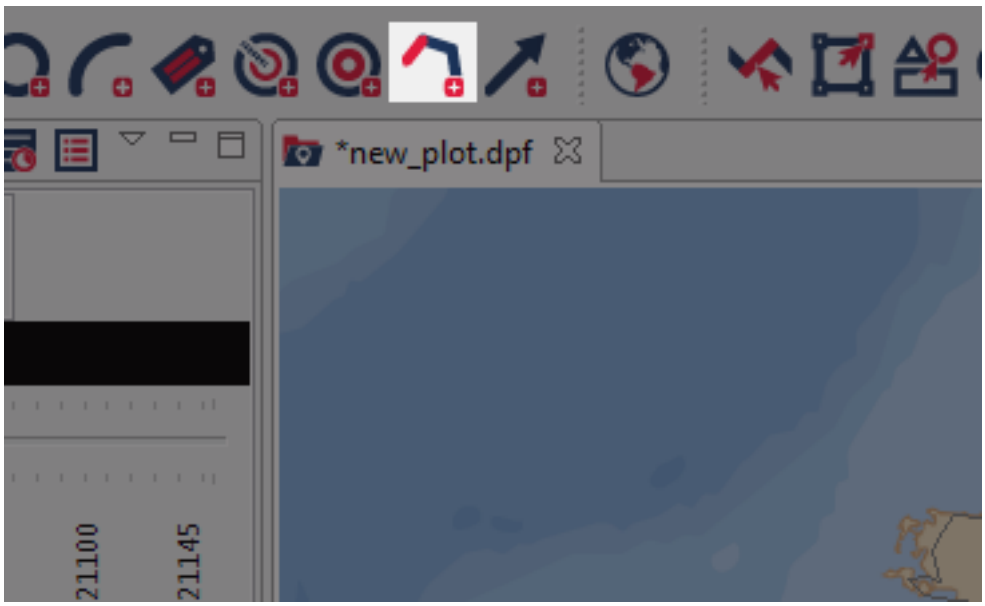
Thus, changing one of the two named parameters will result in the third parameter getting a calculated value.

2. Creating tracks

Newly created planning tracks are created in the centre of the current plot. So, start off by navigating to the part of the world where you roughly want your vessel tracks to be. You'll make more detailed changes to the track location once the track is created.

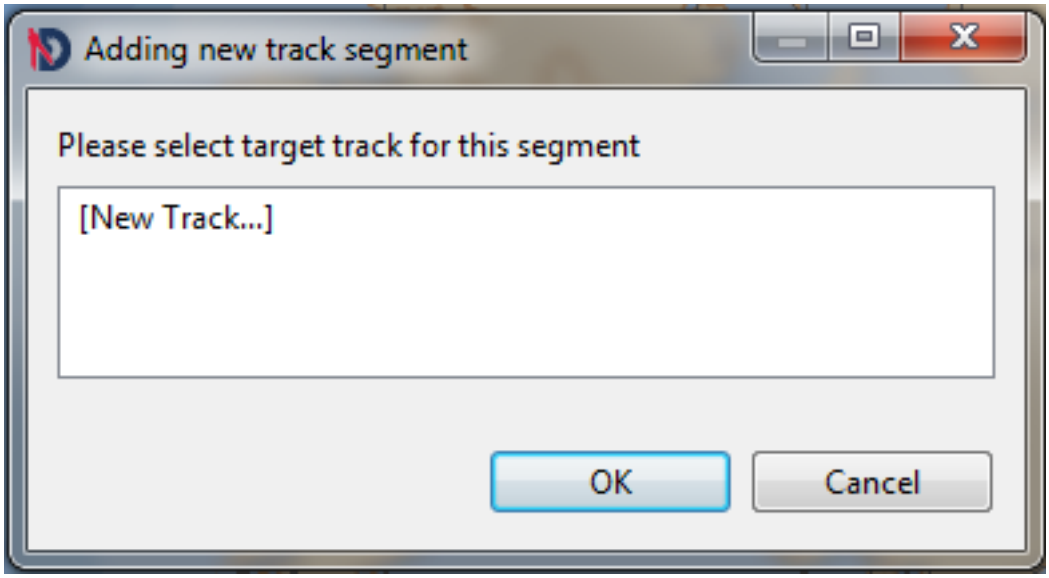
Next, you must select "Create Track Segment" from Debrief's Drawing Toolbar, or the Drawing menu.

Figure B.7.1. Create tracks toolbar button



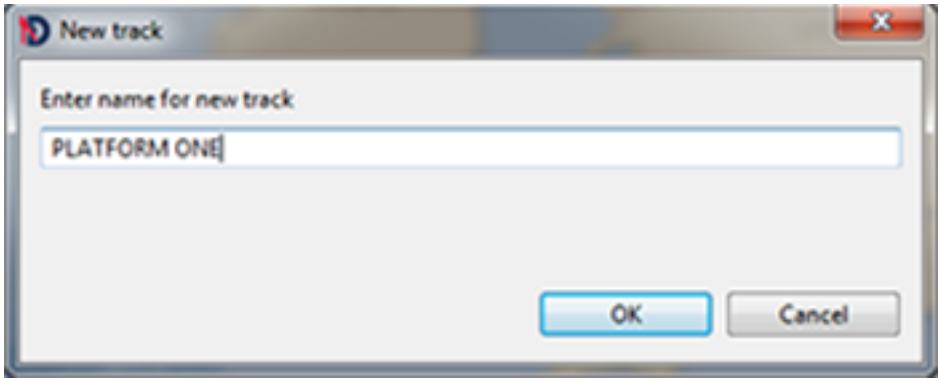
Next you'll be invited to select which parent track to add the track segment to. In this instance, we don't have an existing track, so select "[New Track...]" from the dialog. Clearly if you were extending an existing planning track you'd select it.

Figure B.7.2. Adding a new track segment



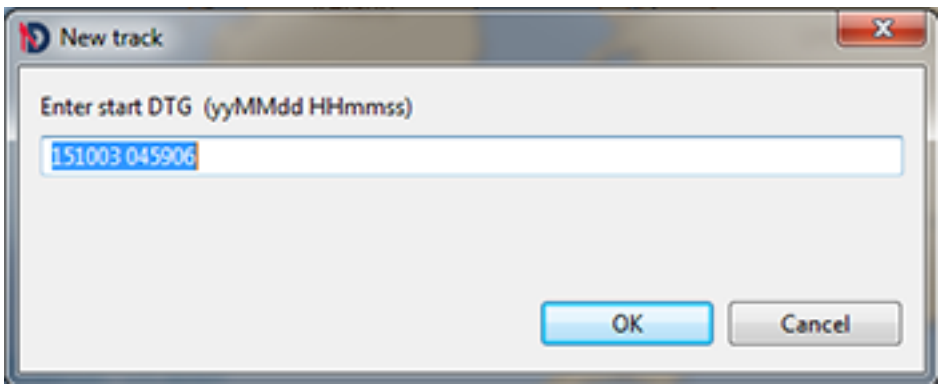
Since we're creating a new track we now see a couple of dialogs associated with the new track. First we must give the new track a name:

Figure B.7.3. Naming the new track



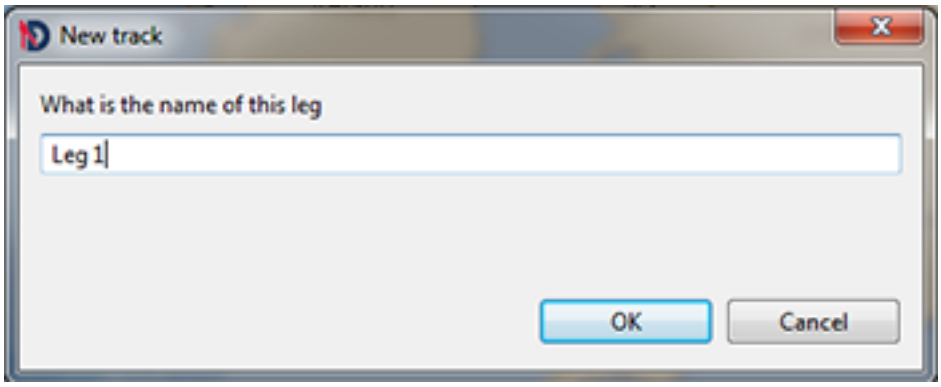
Next we must specify the starting date/time for the track:

Figure B.7.4. Specifying the track's DTG



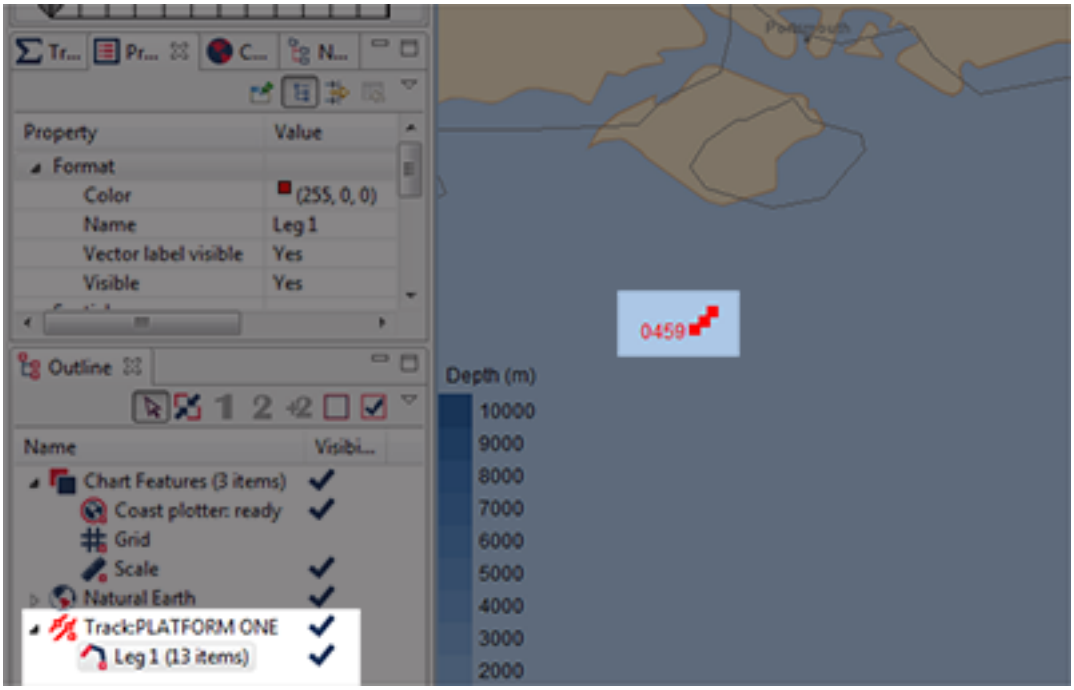
Once we've finished specifying the track, we must specify a name for this particular leg of the track:

Figure B.7.5. Naming the leg



Your track will now appear on the screen, and also in the layer manager.

Figure B.7.6. Track created

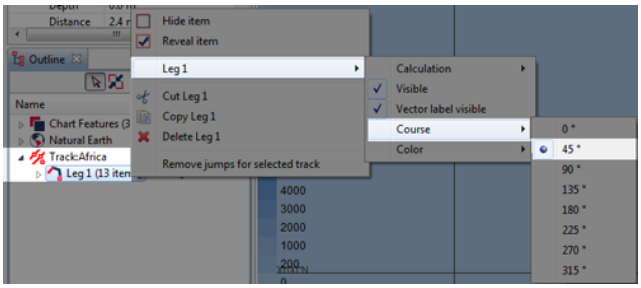


Once you've created a parent track, next time you can select it from the first dialog - after which you just need to specify a leg name,

3. Manipulating tracks

One you've created one or more track legs you can start editing them. There are four ways to edit a leg:

Properties Window	Select a leg in the Outline View, and you'll see the leg properties in the Properties window. Note the Calculation field, this indicates which two fields are used to specify the leg length. As you'll see, you can also amend the leg name, the course, and the depth for that leg.
Outline View	Right-clicking on a leg in the Outline View lets you quickly edit some of the leg parameters



Dragging objects on the plot

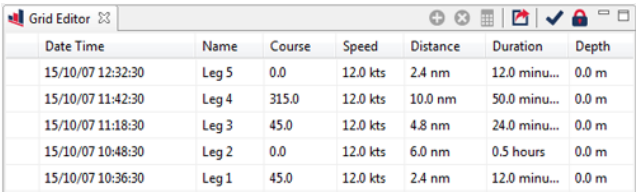
There are two ways of dragging to edit a planning track. You can select Drag Whole Feature  , and then drag the whole planning track to move it, or select Drag Component  and then drag an end of a track leg to rotate it.

Note

It isn't possible to drag to change the length of a leg - since it's unclear which of the two attributes in that segments calculation model the user intends to change.

Using the Grid editor

If you open the Grid Editor (See Section 8, “Using the Grid Editor”) you can view your set of legs in tabular form



Date Time	Name	Course	Speed	Distance	Duration	Depth
15/10/07 12:32:30	Leg 5	0.0	12.0 kts	2.4 nm	12.0 minu...	0.0 m
15/10/07 11:42:30	Leg 4	315.0	12.0 kts	10.0 nm	50.0 minu...	0.0 m
15/10/07 11:18:30	Leg 3	45.0	12.0 kts	4.8 nm	24.0 minu...	0.0 m
15/10/07 10:48:30	Leg 2	0.0	12.0 kts	6.0 nm	0.5 hours	0.0 m
15/10/07 10:36:30	Leg 1	45.0	12.0 kts	2.4 nm	12.0 minu...	0.0 m

Note also that the Grid Editor toolbar includes an Export to Clipboard function that will copy your legs to the clipboard in CSV format

4. XY plots of planning tracks

If you're a clever-clogs scientist doing fancy things like producing run-plans that have a predictable bearing rate, or range between the participants then this is just up your street. If you request an XY Plot of two planning tracks, you can then configure the graph to update itself as the tracks change.

Here's how you do it:

1. Select your two tracks in the Outline View
2. Right-click on one of the tracks
3. Select View XY Plot from the popup menu
4. From the popup dialog, select the calculation you're after
5. Then indicate which is the primary track (though it's irrelevant to many calcs)
6. Once the plot appears, check that the Auto-Sync toggle button in the graph toolbar is selected  .

Now, when you drag or otherwise manipulate your tracks the graph will update to show the new relationship between the tracks. Clearly you un-set the auto-sync toggle to shop updating with the plot.

Chapter B.8. Viewing narratives

1. Introduction to narrative data

1.1. Introduction

The term Narrative Data is used within Debrief to refer to time-stamped textual data recorded during an exercise.

Typically this data would be narrative data recorded to provide an overview of events within an exercise *serial*, but an equally valid use of the facility would be for a textual record of information exported from a recording device. An example of this could be control messages returned from a weapon, or readings taken from an onboard sensor.

1.2. Preparing the data

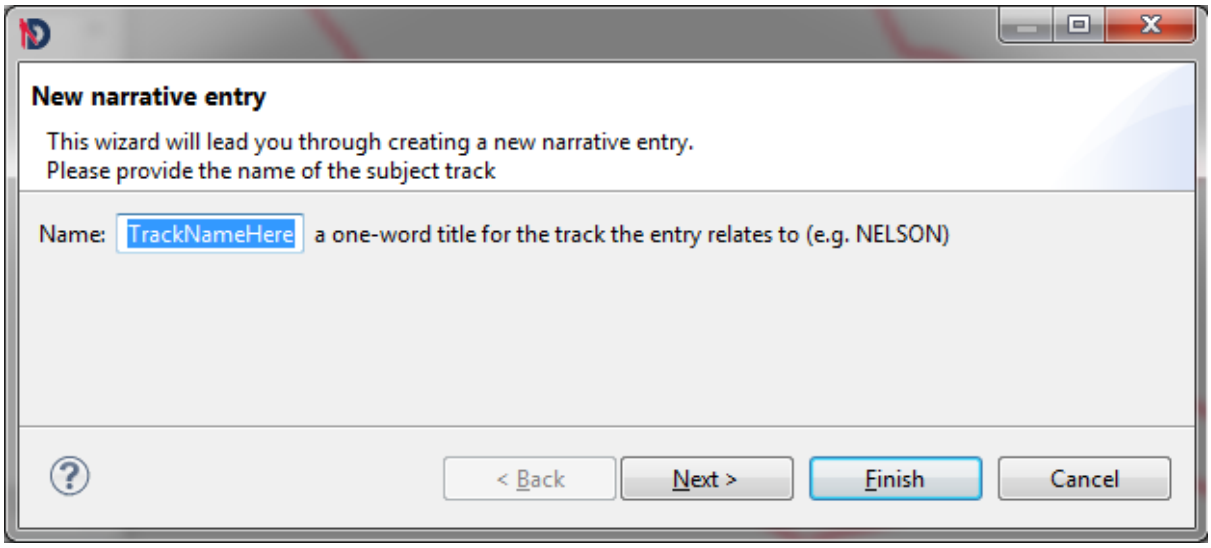
Narrative data is loaded into Debrief using the ;NARRATIVE and ;NARRATIVE2 replay file format entries as described at Section 1.3, “Annotation Data Intro”.

The narrative data can be located in a `.REP` file of its own, or together with other Debrief track and annotation data.

1.3. Loading the data

Narrative data is loaded into Debrief in the same way as other data, by dragging and dropping from a REP file formatted as above, or by using the *Generate New Narrative entry* wizard. Open the wizard by either right-clicking on the *Narratives* entry in the Outline View, or by right-clicking in blank area of the Outline View if this is the first narrative entry.

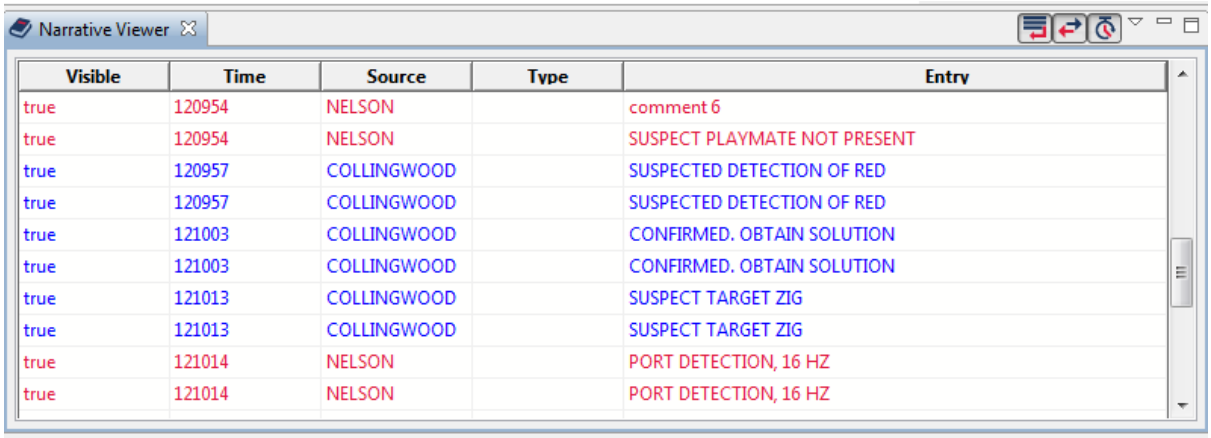
Figure B.8.1. New narrative entry wizard



1.4. Viewing the data

Once the data is loaded into Debrief, it is displayed in a list window - with one entry per line. When stepping through the data the "current" entry is highlighted, but the user is also able to double-click on an entry to move the Debrief step time to the time this entry was recorded.

Figure B.8.2. Viewing a narrative



Visible	Time	Source	Type	Entry
true	120954	NELSON		comment 6
true	120954	NELSON		SUSPECT PLAYMATE NOT PRESENT
true	120957	COLLINGWOOD		SUSPECTED DETECTION OF RED
true	120957	COLLINGWOOD		SUSPECTED DETECTION OF RED
true	121003	COLLINGWOOD		CONFIRMED. OBTAIN SOLUTION
true	121003	COLLINGWOOD		CONFIRMED. OBTAIN SOLUTION
true	121013	COLLINGWOOD		SUSPECT TARGET ZIG
true	121013	COLLINGWOOD		SUSPECT TARGET ZIG
true	121014	NELSON		PORT DETECTION, 16 HZ
true	121014	NELSON		PORT DETECTION, 16 HZ

Three settings affect how the Narrative Viewer integrates with the current plot time (as indicated on the Time Controller (see Section 2.1, “The Time Controller”).

Trim the narrative entries to the visible screen space

Highlight the narrative entry nearest to the current plot time.

Change the plot-time to that of a narrative entry when the user double-clicks on it.

Also, the narrative viewer is able to filter your data for you. Double-click on the Source or Type headings and a filter dialog will open.

Figure B.8.3. Filtering a narrative by Type

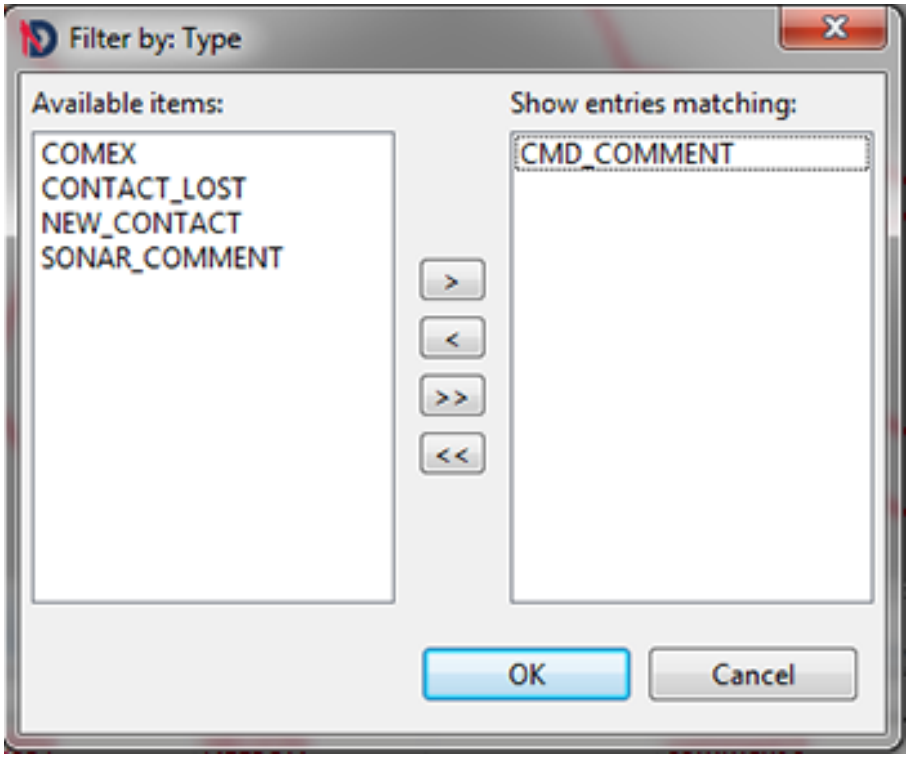


Figure B.8.4. Filtering a narrative by Source

In the examples shown, select which types of data you wish to see in the filtered narrative by either double-clicking on them or selecting one then clicking the right arrow. When you've selected which data you want to see, click OK.

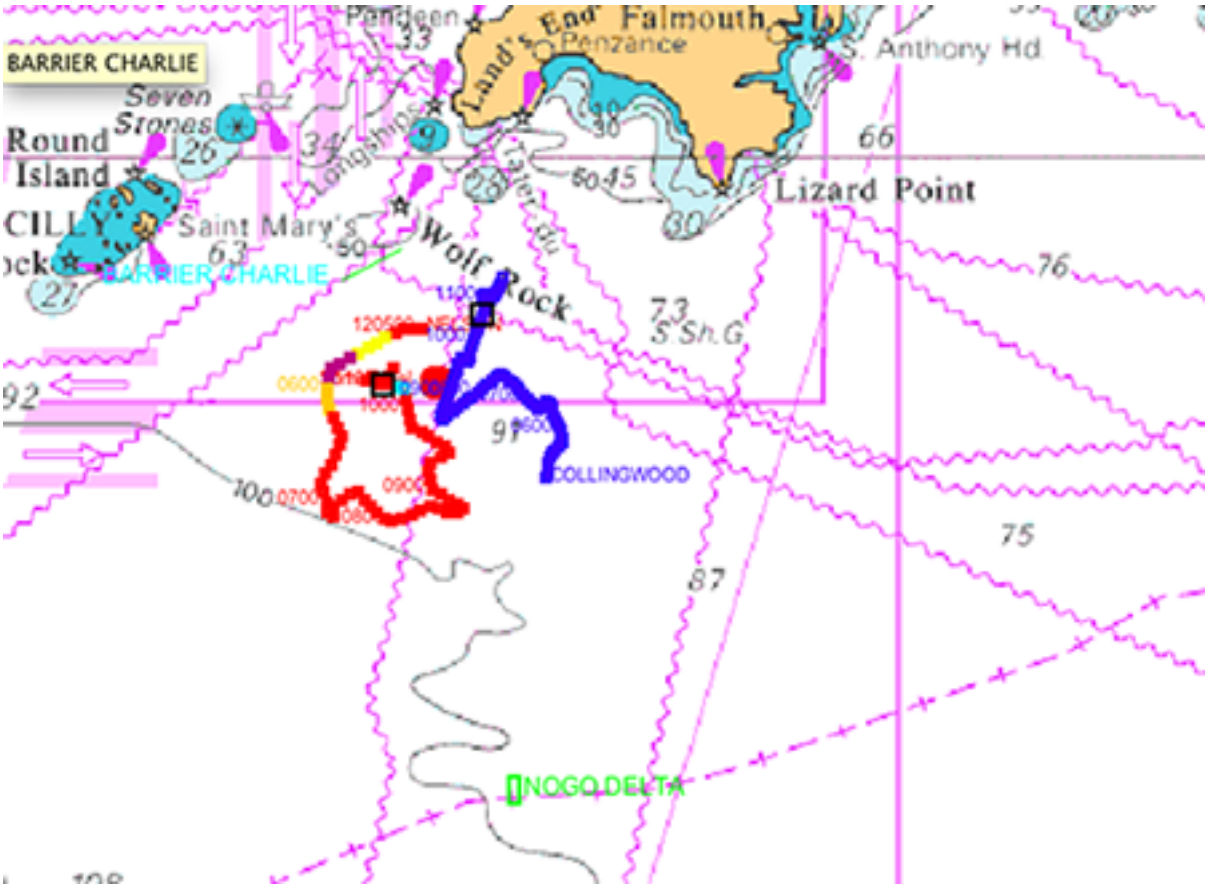
The drop-down menu for the narrative viewer also allows you to select how you want the date column formatted - helpful for very frequent or infrequent narrative entries.

Chapter B.9. Using chart backdrops

1. Introduction

VPF and ETOPO (see Chapter B.6, *External datasets*) can only add so much context to the analysis of a dataset. Nautical Chart data contains a large volume of additional infromation that can assist analysis. Debrief supports chart data distributed by UKHO, though the standards used also allow chart from other sources to be incorporated. The chart support in Debrief is provided through integration of GeoTools library (see Chapter D.3, *GeoTools in Debrief*).

Figure B.9.1. Chart data in Debrief



1.1. File types

Debrief supports the following types of chart data:

World Image file (`.tif`)	A raster chart image, with reference coordinates stored in a supporting .prj file
Esri Shapefiles (`.shp`)	A binary file adhering to the Esri open specification

Beyond specific file types, Debrief recognises chart folios distributed by UKHO. Where these folios are configured within Debrief users are able to view the spatial extents of available charts - loading them as necessary.

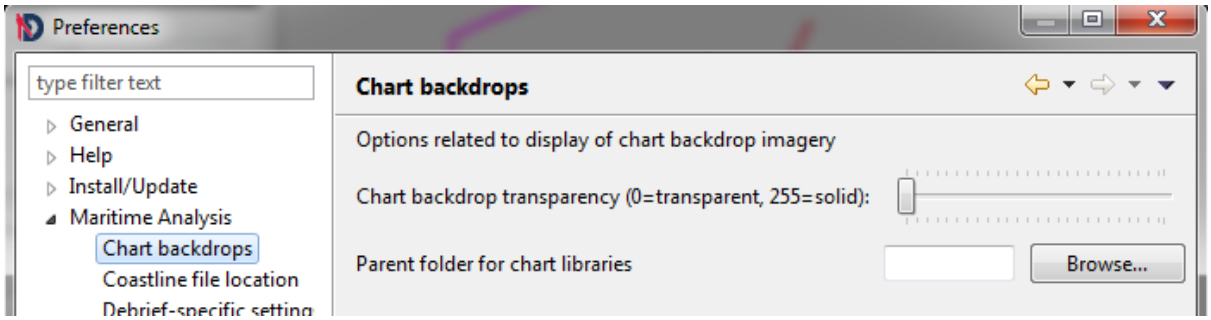
2. Loading data

Data is loaded by one of two methods: drag/drop of a specific chart file, or via a chart library.

Any shp or tif file can be dragged into Debrief - provided the prj sidecar files are present. When saving the plot Debrief will record the path to the datafile in the .dpf file in order that it can be reloaded at the next session.

Separately, Debrief can provide a folio of charts, though the data must be stored in a specific layout (see Section 3, “Storing Chart Folios”), and Debrief must be configured appropriately. Configure Debrief this by selecting the Chart Backdrops page under the Maritime Analysis section of the Window/Preferences dialog. In this page specify the parent folder sitting above any chart folio folders. Debrief will search the subfolders of this location to locate any chart folio files.

Figure B.9.2. Chart options

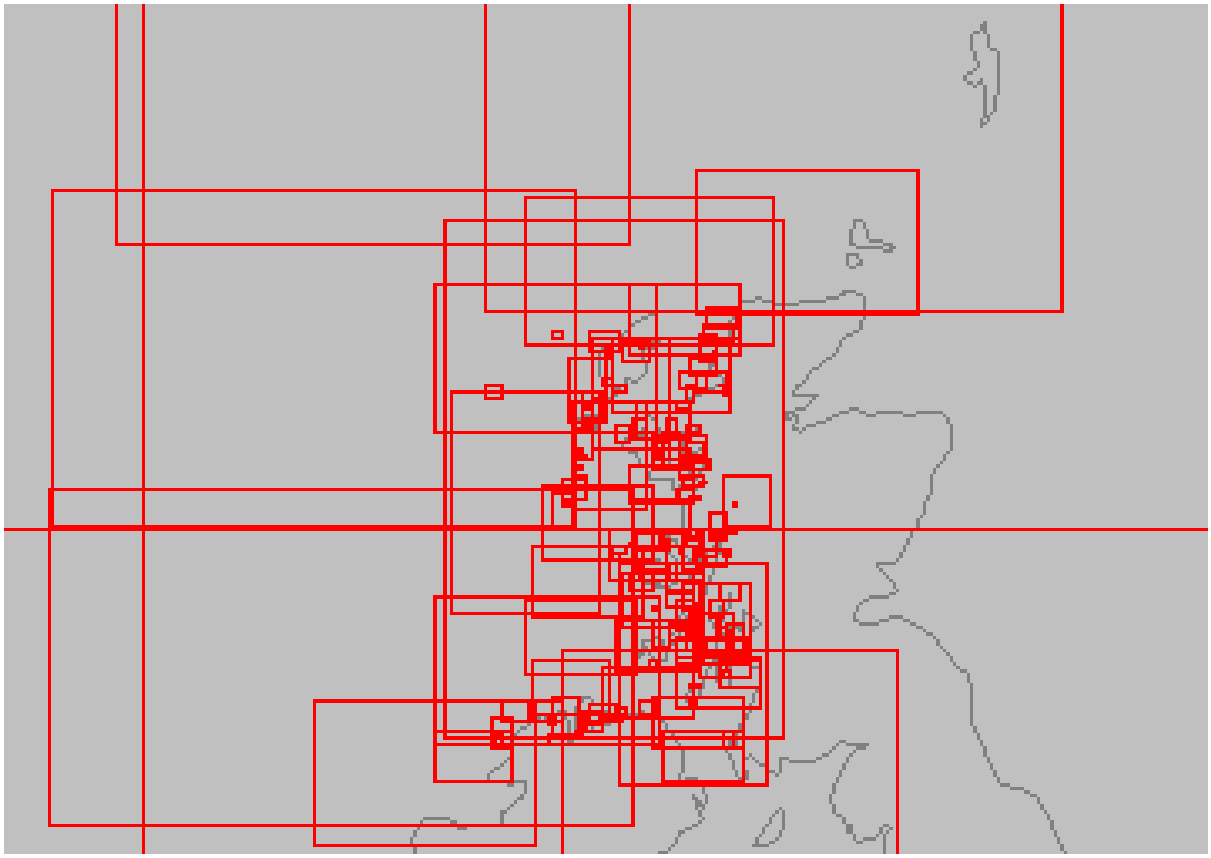


Note

As you saw, there is also a transparency attribute for charts. Use this to vary the transparency of the charts layers. For most analysis the presence of a chart backdrop can make vessel tracks difficult to distinguish. Making them semi-transparent fades them away a little - letting you focus on the tracks whilst still being aware of the backdrop chart data. Unfortunately, the transparency setting isn't observed when you copy the plot to the clipboard. Once pasted in MS Word, all charts are painted solid.

Once a parent folder is specified, when you click on Insert Chart Library from the Chart Features menu Debrief will scan the child folders for correctly named shapefiles and offer you a list of folios to choose from. Select one to load the folio.

Figure B.9.3. Chart folio

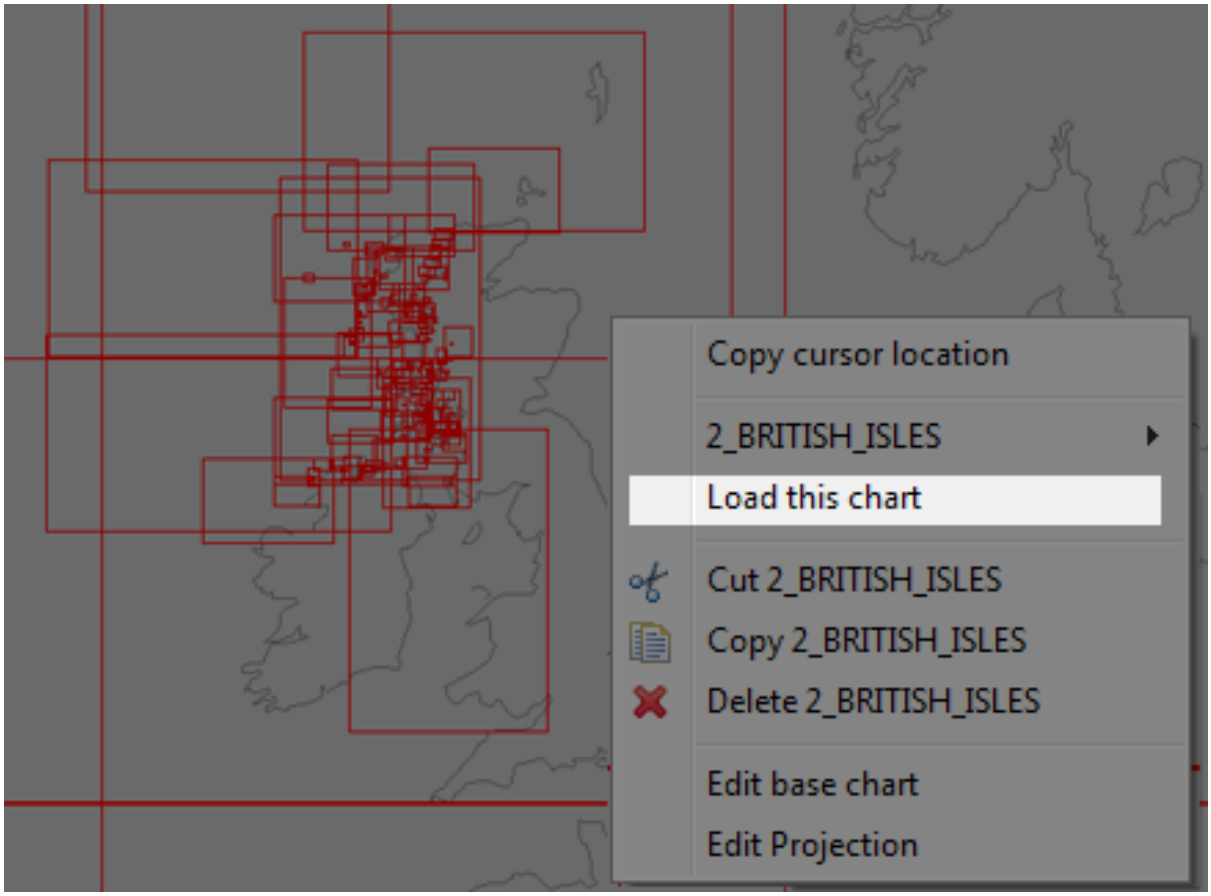


Note

If you right-click on the chart library in the Outline View you'll see the editable properties. These include the colour used for the rectangles plus whether the chart name is displayed against the chart rectangle.

The folio displays a rectangle for each available chart. The folio is a top-level layer in the Outline View that you can expand to view (and load) the child charts. Alternatively, right-click on a corner of one of the rectangles and select 'Load this chart' to load a specific chart.

Figure B.9.4. Loading a chart



The chart will then be loaded as a separate Debrief layer - and appear in the layer manager. As with any other layer, you can control it's visibility using the Outline View. Note, Debrief isn't going to store the chart image in the plot-file, it's just going to store the path to the Image - which will re-loaded next time you open the plot.

Chapter B.10. Analysing sensor data

1. Getting your data in

1.1. Introduction

Debrief uses the following terms:

Sensor	The *sensor* which recorded the data. Sensor data is grouped according to its sensor. This characteristic may be exploited by giving a unique sensor name to each track being recorded on a sensor - allowing tracks to be independently switched on and off.
Sensor Contact	This is an individual contact recorded on a sensor, a single bearing line reaching from the sensor location (origin) along the contact bearing to the contact range.

The sticky issue of whether to represent sensor data in absolute coordinates (where each line has its own origin) or whether to represent the data in relative coordinates (where the sensor origin is assumed to be the current ownship position) is managed through the ability to enter NULL fields for the sensor location.

Support for relative coordinates is provided for two reasons:

- Some *sensor* data sources may genuinely not contain positional data - allowing relative coordinates will ease the workload in these instances.
- In the submarine plot-lock process it is quite common to experiment with a number of track-shifts until the hull-mounted sensor bearing fans tie up with the other vessel. By keeping the sensor data separate to the vessel data, the existing sensor data-file can be dropped into Debrief with the updated vessel track-file, allowing the user to perform a visual qualitative check on the shift applied.

Note, it is not possible to define the sensor offset value in the REP file (see Section 1.4, “Sensor offset lengths editor”), it may only be defined from the Properties Window of Debrief NG. The value is stored safely when the Debrief plot is saved, however.

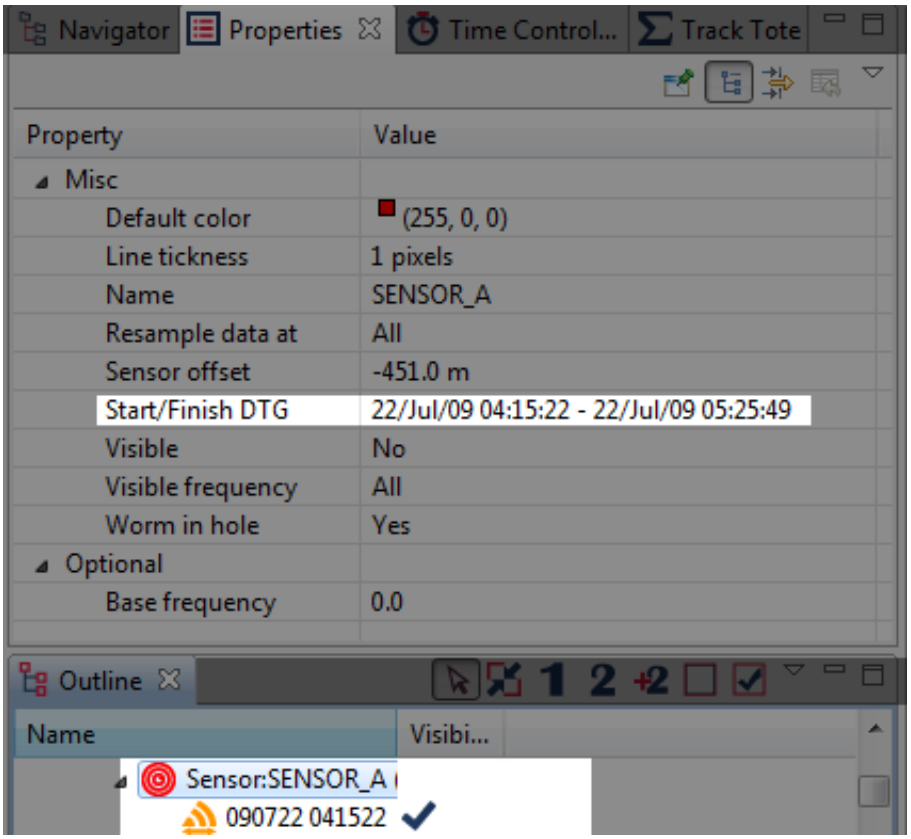
When Sensor Contact data is being plotted for a Sensor using relative coordinates, Debrief reflects a Sensor Offset distance for that sensor. The sensor offset distance denotes the horizontal distance between the centre of the centre and the attack datum of the host platform (+ve forwards, -ve backwards - so towed array data would be -ve). See the next section for more detail on the support for a shared library of array lengths. The location of the start of the sensor contact bearing line is calculated using the current value of course for the host platform and the sensor offset value.

Note

Sensor data is typically added to a plot by dragging/dropping it from the Navigator view. If the dragged in datafile only contains cuts from a single sensor a dialog pops up give you chance to override the sensor name and color. If the file contains data from more than one sensor then they just quietly slip in.

1.2. Viewing Sensor Data Time Period

When viewing the properties view, you will see a Start/Stop DTG for the current sensor data time period. This entry is read only, but provides an at a glance method of viewing range set for the sensor data.



1.3. Worm in the hole

When examining a Sensor in the properties view, you will see that in addition to Sensor Offset, there is a 'Worm in the hole' (see Section 2, “Worm in the hole algorithm”). This is typically used for towed arrays, whereby the start of the sensor bearing line is located at the Sensor Offset distance back along the host platform track, instead of in a straight line behind the host.

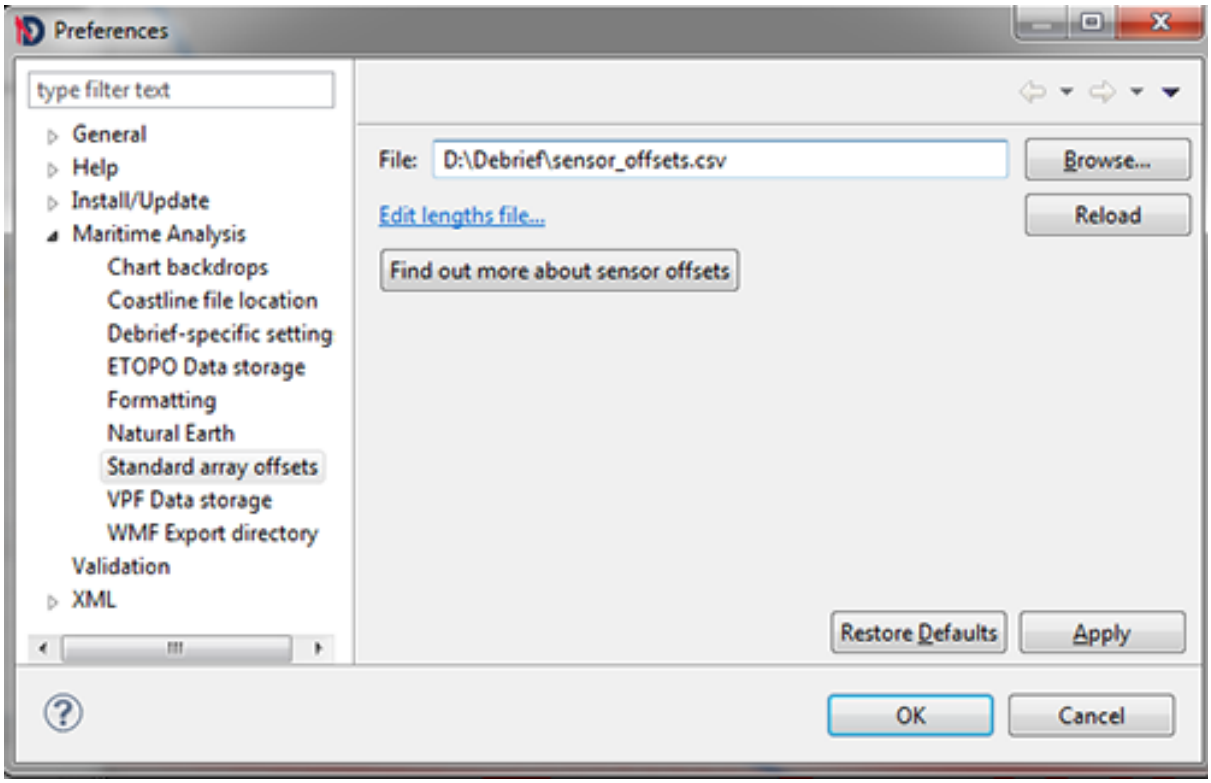
1.4. Sensor offset lengths editor

Since Autumn 09 Debrief has supported a library of sensor offsets. These offsets are used to populate a drop-down list of lengths, used in support of specifying sensor offsets.

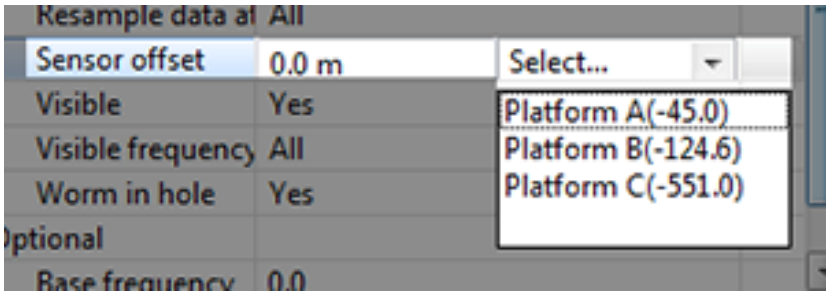
Somewhere on your network create a csv-formatted text file that stores two columns of data (as shown below). The first column is the platform/sensor name, the second column is that combination's array offset in metres (with -ve figures at the stern of the host platform). The file should end in a csv suffix, and is formatted as follows:

```
Sensor Name,Length (m)
Platform A,-45
Platform B,-124.6
Platform C,-551
```

Then open the Debrief preferences page, and specify the offset file location using the Standard Array Offsets page



Once the standard offsets file has been specified, you are provided with a drop-down list of sensor offset distances when you view sensor data in the properties window:



Note

Note, if you haven't got a sensor array offsets file assigned, you can still enter a value by hand. But, you do not need to specify the metres units. For an array offset of -400m, just enter -400.

1.5. Preparing Sensor Data

Sensor data is loaded into Debrief in REP files, just like any other Debrief data. The line format is one of:

```
;SENSOR: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS  H
 BBB.B
 RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
 bearing (degs), range(yds) [or the single world NULL], sensor name,
 label (to end of line)
```

or

```
;SENSOR2: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
 BBB.B
 CCC.C FFF.F RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
 bearing (degs)  [or the single word NULL], ambigous bearing (degs)
 [or the
 single word NULL], frequency(Hz) [or the single word NULL],
 range(yds) [or
 the single word NULL], sensor name, label (to end of line)
```

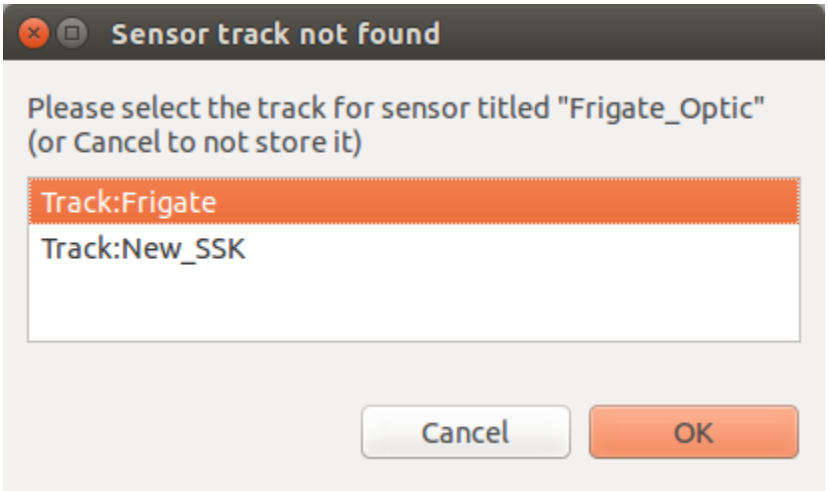
or

```
;SENSOR3: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
 BBB.B  CCC.C
          FFF.F GGG.G RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
bearing (degs) [or the single word NULL], bearing accuracy (degs)
[or the single word NULL], frequency(Hz) [or the single word NULL],
frequency accuracy (Hz) [or the single word NULL], range(yds)
[or the single word NULL], sensor name, label (to end of line)
```

As you can see, unlike most other Debrief line formats, this format allows for NULL fields. Where the sensor latitude and longitude values are replaced by the single word NULL, Debrief plots this sensor contact using a relative origin. NULL values may also be provided for the ambiguous sensor bearing and/or detection frequency.

Note

It's quite common to produce a DSF file where the track name doesn't exactly match the tracks already loaded. So, if, when importing a REP file the track name doesn't match any existing tracks, Debrief will invite you to choose the track to add the sensor to.



1.6. Relative Data

When (as described above) Debrief plots data using a relative origin, it follows the following procedure:

- The first time the sensor contact line is plotted, it examines its parent track to find the vessel position nearest to (or greater than) the sensor contact DTG.
- This position is then offset by a vector produced from the sensor offset value and vessel's current course.
- The sensor contact then calculates the position of its far end relative to this origin

2. Analysing your data

2.1. Introduction

When first loaded, *sensor* data is not made visible, since with any reasonable volume of sensor data the plot quickly becomes illegible. Sensor data is switched on and off individually by accessing the sensor, via its Track, from the Outline View.

It is once in *snail trail* mode that sensor data is most easily analysed. When in snail mode the Snail display mode performs the following processing:

- For each Track being plotted, the display mode looks to see if it contains any Sensor data.
- It then examines each list of Sensor data to see if it's visible. If it is visible, it plots the current sensor contact (nearest to the *Tote* time), followed by the sensor contacts as disappearing contacts running back through the length indicated in the TrailLength parameter in the properties window.

Generate track from Active Sensor Data. If your sensor data has both range and bearing, you have all the data you need to generate a target track. If you right-click on a Sensor, or on a block of sensor cuts in the Outline View then Debrief will inspect them to see if they are suitable for generating a target track. Specifically it will check that they all have a Range value, a Bearing value, and no ambiguous bearing. If the selected data matches these constraints you will be invited to Generate Track from Active Sensor Data, and Debrief will generate a new track, named according to the sensor that produced it.

2.2. Managing high volumes of sensor data

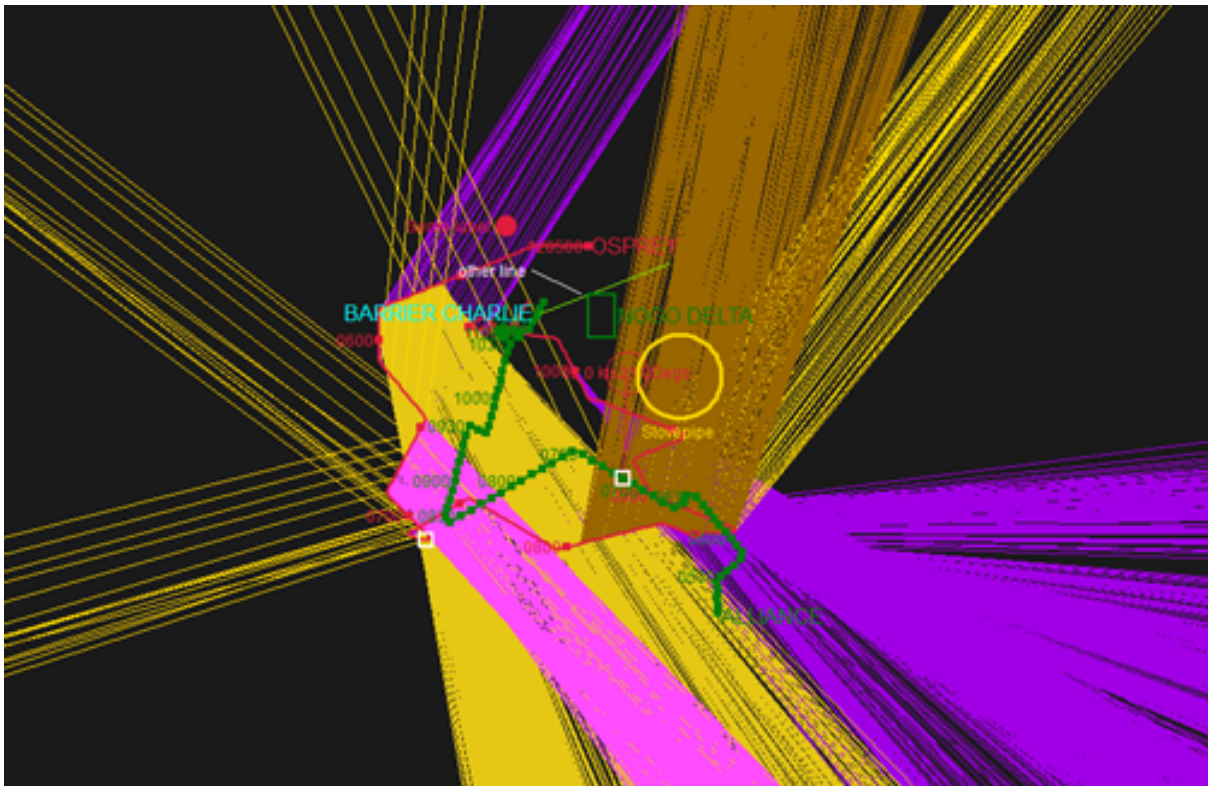
2.2.1. Introduction

Modern command systems produce high volumes of sensor data, and a command system that just uses a 3-digit track counter can easily go 'around' the clock when allocating track numbers to contacts.

Debrief provides capabilities to both ease the challenge of deciding which sensor data is related to a specific target, and to automatically split a single sensor track in multiple tracks when they clearly relate to different targets.

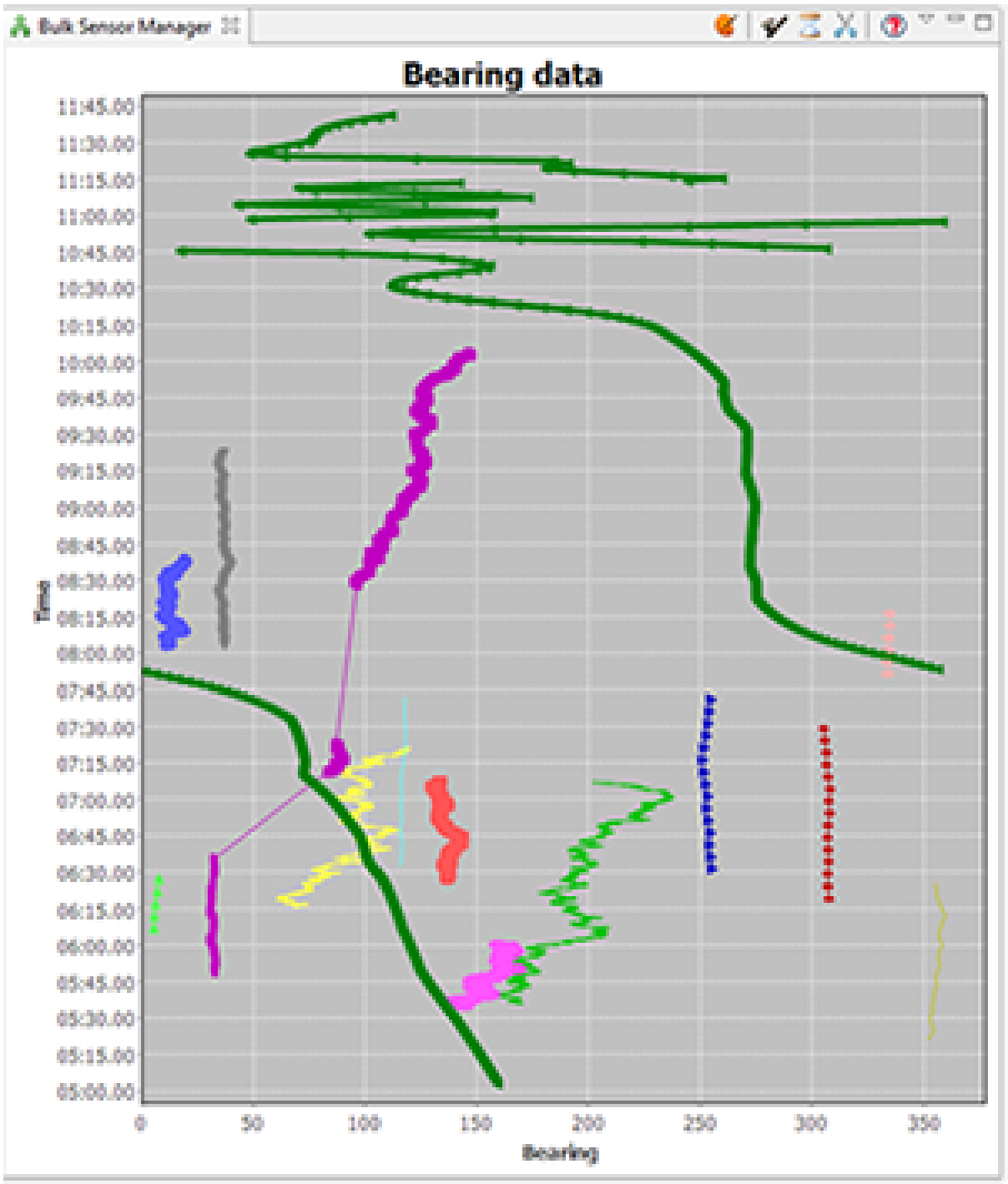
2.2.2. Filter sensor data of interest

With lots of sensor data it can be increasingly difficult to determine which tracks should be made visible - the plot below shows just 12 visible tracks - it's possible to have many hundreds of tracks..



Cumbersome sensor data

The new Bulk Sensor Manager offers an opportunity to determine which sensor data has a direction near to one of the other tracks currently loaded in Debrief. Before opening the Bulk Sensor Manager it's worth telling Debrief which is your 'ownship' track, and which are the tracks of interest. Do this by marking tracks as primary and secondary in the Track Tote (see Section 1, “Assigning tracks as primary and secondary”). Once you have a primary track set, select Bulk Sensor Manager from the Window/Show View menu. You'll see a plot like the one below:



As you'll see, the view shows multiple series of track data. Currently visible tracks are displayed in bold lines (the thick green line in the screenshot) - so, you'll typically be trying to determine which blocks of sensor data are close to/similar to target tracks. Sensor data is shown with symbols if it's visible in the plot (see the pink line near the centre-left of the image). Clicking on a block of sensor data in the view also selects that sensor data in the Outline View (though on occasion you need to expand the Sensors folder in the Layer Manager). As blocks of sensor data are selected, they are shown in black in the view. Items can be multiple-selected with the <ctrl> key. Once multiple blocks of sensor data are selected in the Outline View you can right-click on them and select 'Merge sensors into *xxxx*'

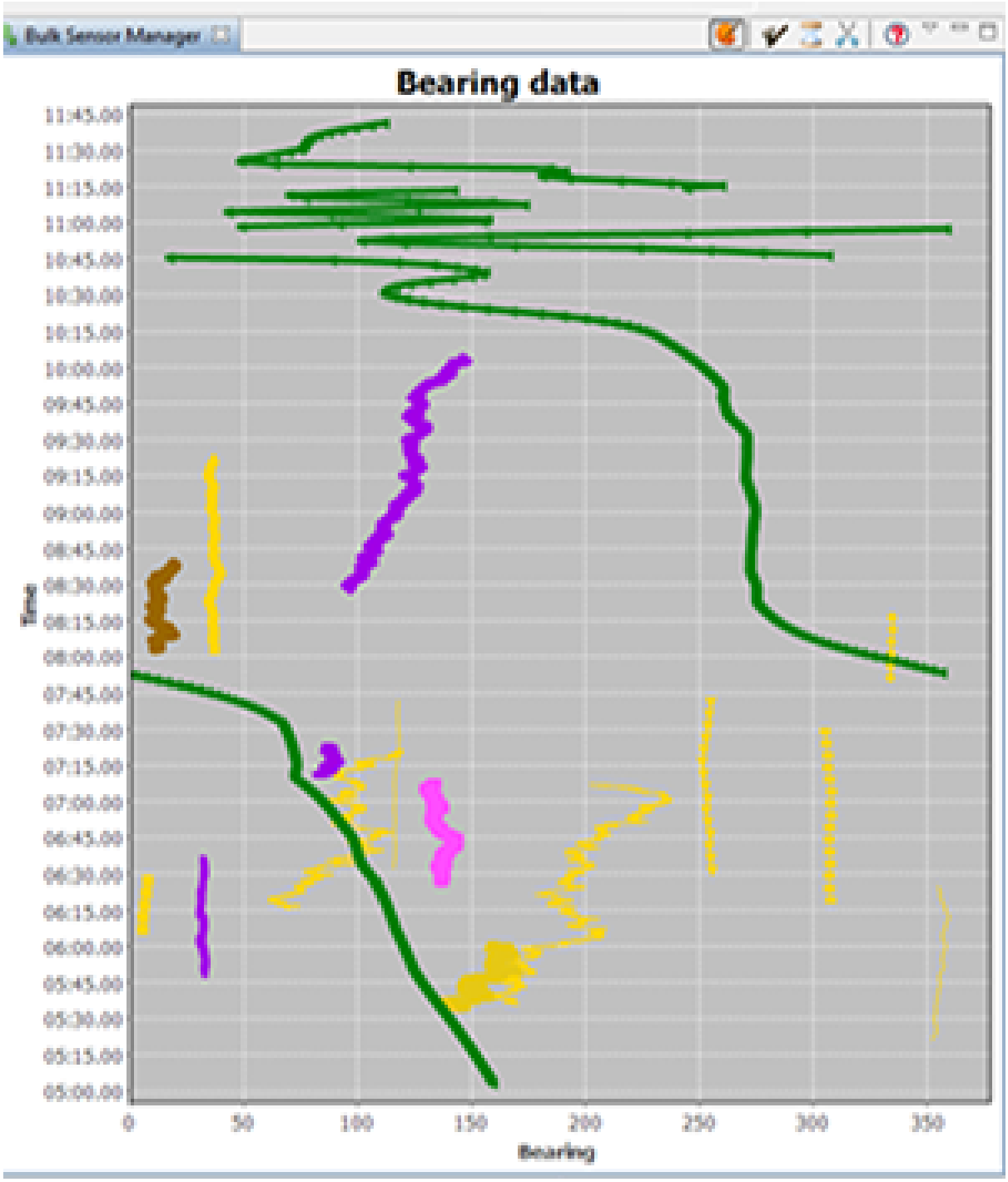
Also note that whilst by default comes up with a unique color/symbol combination for each block of sensor data, you can click on the 'Show original colours' palette icon at the top of the view. This will switch the Bulk Sensor Manager to using the sensor data color as configured via the sensor manager.

2.2.3. Automatically split sensor tracks

For data from a command system that frequently 'wraps around' the track counter, when the data is read in Debrief will mistakenly assume that the data is all for the same contact: so the data will show as a continuous series to the left of the chart, then there'll be a jump over to were that track number is used again against another contact on the right hand side.

Debrief offers a tool to automatically split sensor data - the scissors icon at the top of the Bulk Sensor Manager offers 'Auto-split sensor segments'. It will pass through all of the sensor tracks in the primary track, and if there is a jump of more than 15 seconds Debrief splits the successive data into a new tracks. The original and new tracks are then renamed with a "_1", "_2" suffix.

In the above screenshot you can see that the purple dataset actually covers 3 periods of sensor data, roughly: 0545-0640, 0710-0720 and 0830-1010. If we click on the Axe toolbar button ( , Auto-split sensor segments), then we will see the tracks separated as in the following screenshot.



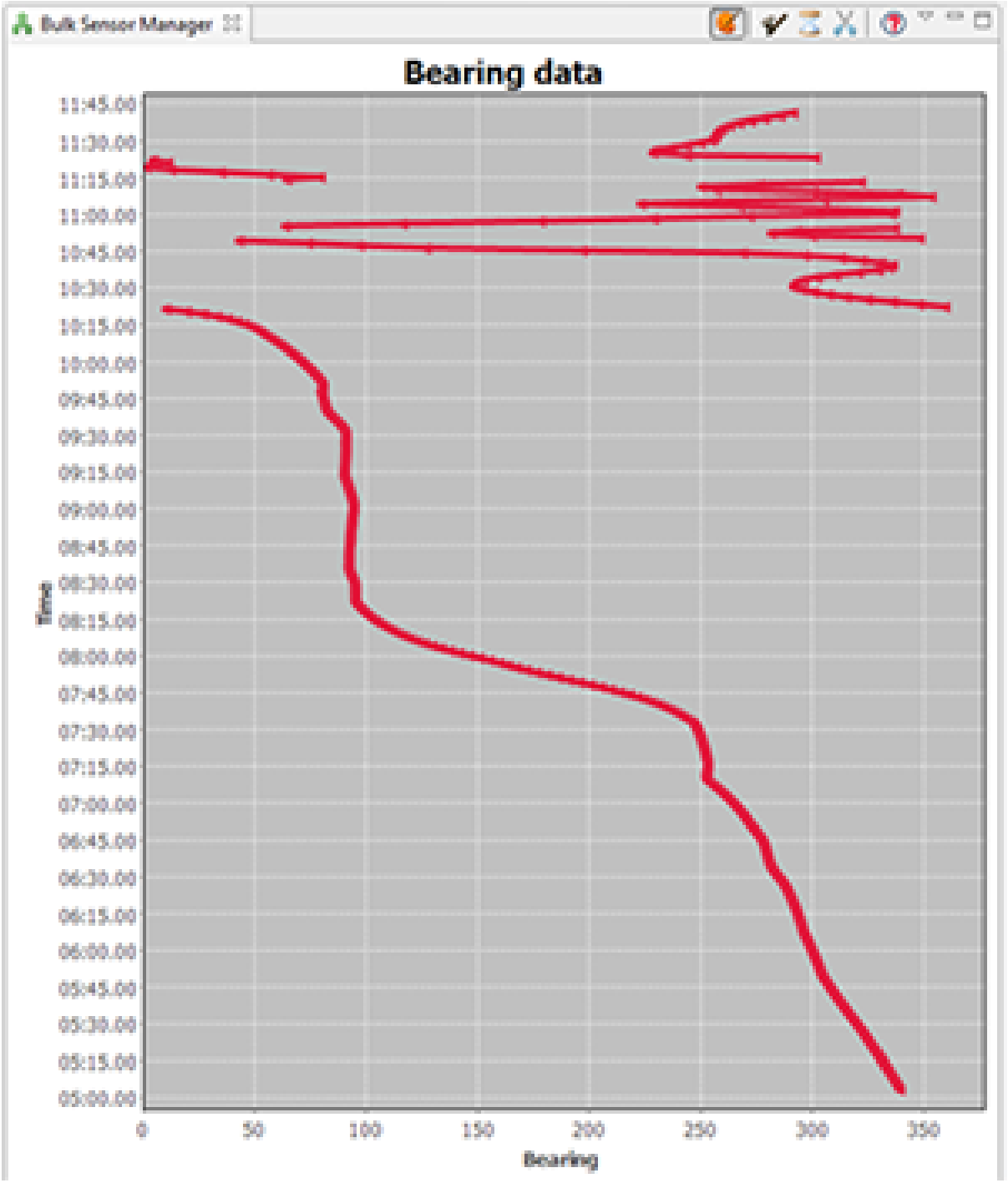
2.2.4. Trim to track period

Sometimes the data extraction process results in sensor datasets being loaded/provided that cover time periods outside the range of the parent track. These unnecessarily slow down Debrief - so you may choose to delete them. The Trim to track period() operation will delete any blocks of sensor data that completely fall outside the time period of the primary track.

2.2.5. Remove sensor data unrelated to Secondary Tracks

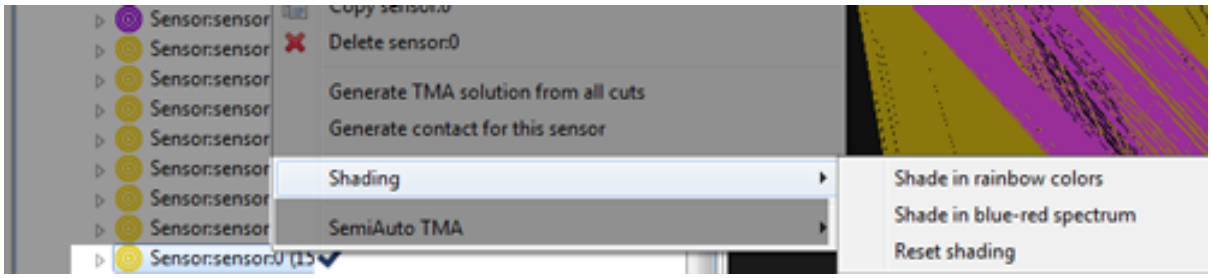
Sensor data loaded from a command system may contain many, many tracks. Many of the tracks will be of contacts unrelated to the platforms in question. It's quite easy for Debrief to determine sensor data that isn't related to the loaded secondary tracks. By clicking on the Remove sensor data

unrelated to Secondary Tracks () button, Debrief will compare the bearing from the primary track to each secondary track. If a block of sensor data doesn't have a single cut that is within 45 degrees of any secondary track, it will be deleted. Running this operation on the above sample dataset gives this greatly reduced sensor dataset:

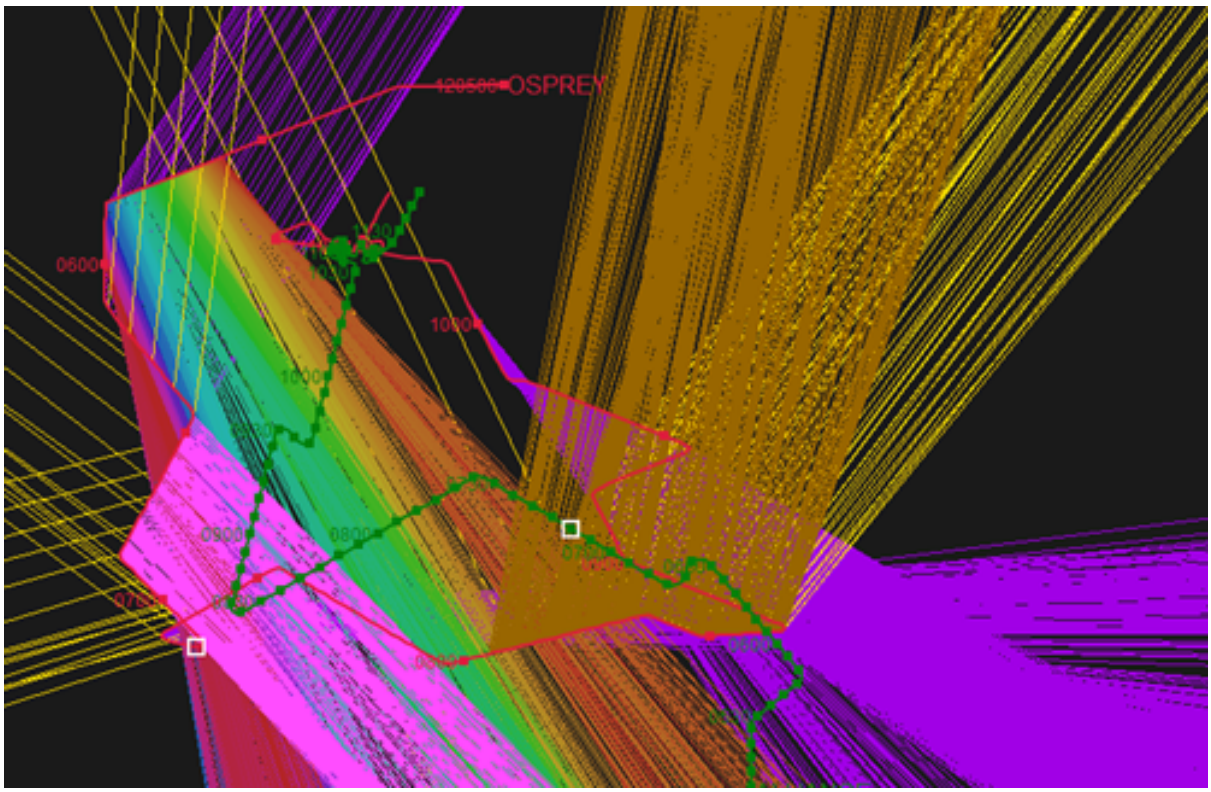


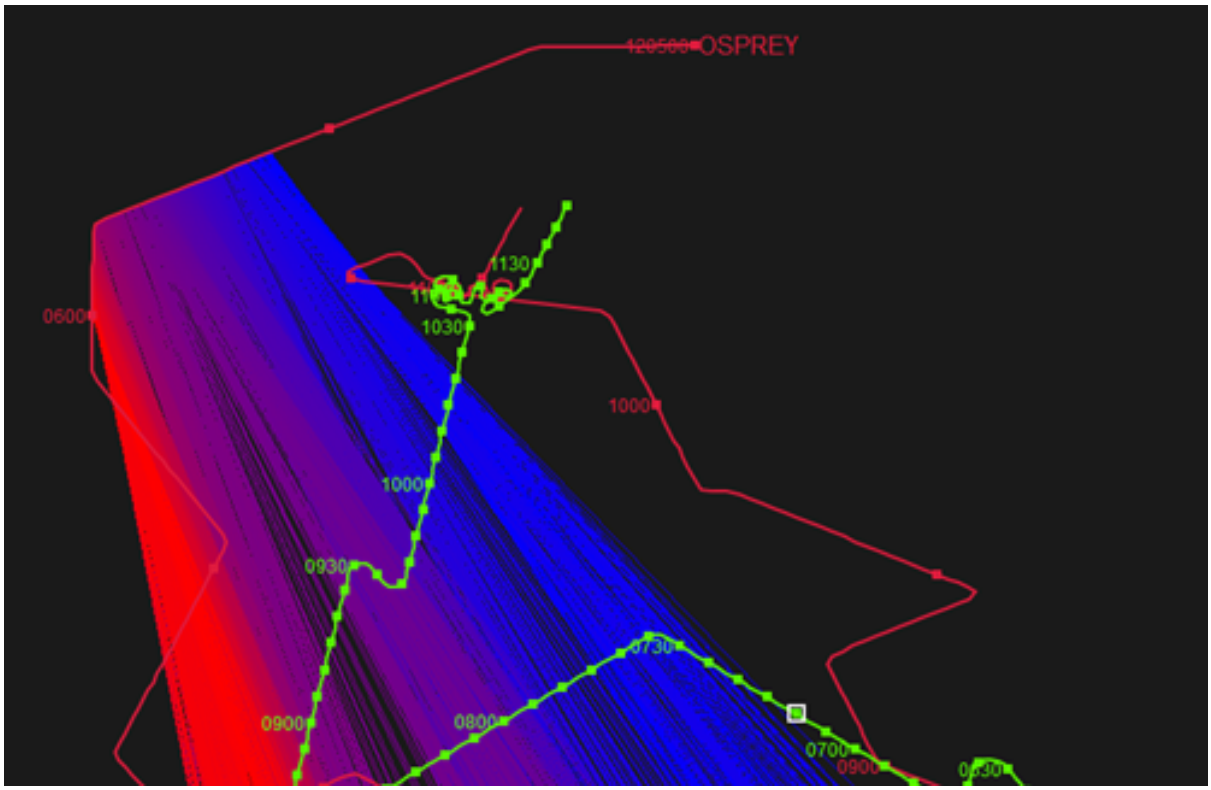
2.2.6. Automatically shading bulk sensor data

When very long sets of sensor data are exported to a report, it can be difficult for a reader to distinguish newer from old cuts - particularly when they are overlapping. Debrief provides support for this via the two Shade operations:



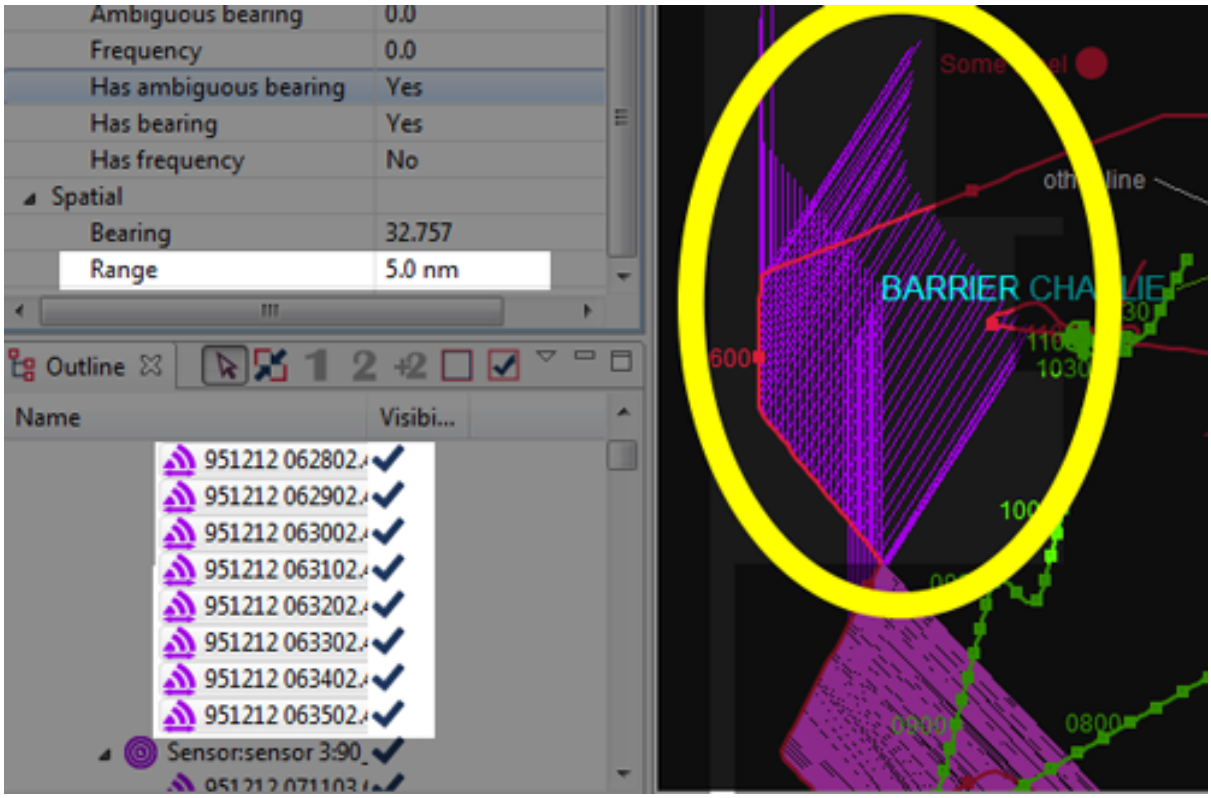
These two operations shade the selected block of sensor data (or whole sensor) according to two operations - as shown below.





2.2.7. Trim length of sensor lines

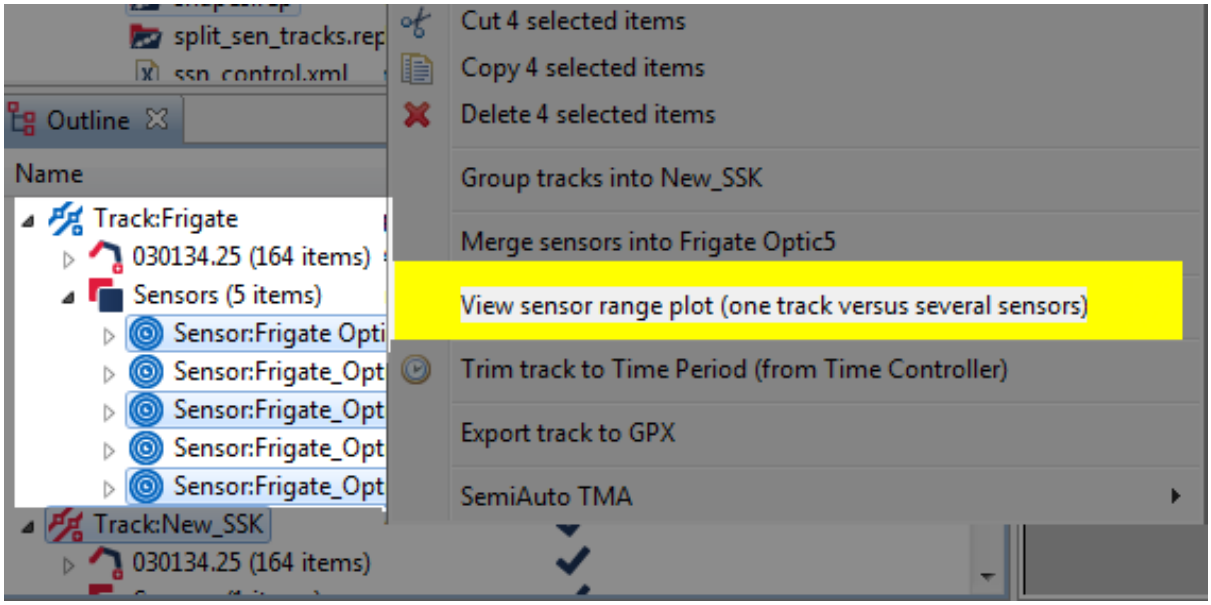
As you'll see in the earlier *Cumbersome sensor plot*, a lot of sensor data can make a plot difficult to interpret. If sensor data isn't provided with a range attribute, then sensor lines are drawn out to infinity (or the edge to the viewport - whichever is nearer...). On a large area plot this may given sensor lines of 10s of nautical miles long. If in truth the detections are only in the thousands of yards, you may wish to give the sensor data a range. To do this, start by multi-selecting a block of sensor data in the Outline View. Next, open the properties window - you'll see the elements have zero range. Just insert an 'indicative' value for range. The sensor lines on the plot will now be clipped to that range - greatly reducing plot clutter.



Trimmed sensor data

2.3. View plot of sensor range to target

Whilst Debrief makes it easy to view a plot of the range between two vehicles, on occasion you may wish to view a time graph of the range from the array centre to the target. This graph can contain the distance from one or more sensors to a particular target. Open the graph by multi-selecting (<ctrl-click>) a series of sensor wrappers and a single target track (alternately you could select one track plus the 'Sensors' item of another track if you wish to view a range plot that includes all the sensors). If such a combination of objects have been selected in the Outline View, when you right-click on them you are offered the 'View sensor range plot' menu item:



Popup menu with a track plus multiple sensor selected

Note, the Sensor Range Plot reflects the sensor offset length, plus whether the 'Worm in the hole' (see Section 1.3, “Worm in the hole”) property is set for each sensor.

Tip

On occasion you may wish to view a sensor range plot of a sensor for which you haven't loaded sensor data. That is, you've loaded & displayed hull-mounted sonar data, but you wish to know the range from the centre of the array to the target. Do this as follows:

1. Right-click on the ownship track, select 'Add new sensor'
2. Name the new sensor using the wizard, and specify its default color
3. Now select the new sensor in the Outline View - in order that you can edit its properties. In the property window set the correct sensor offset distance (remember it's in metres, and specify negative values to represent a trailing array), and mark that it should use a 'worm in the hole' algorithm
4. Now, back to the Outline View - multi-select the new sensor, and the target track. Right-click on them and select View sensor range plot.

In this mode of use, the algorithm generates a range for each calculated position on the target track. In the normal mode (where we have sensor data), a range is calculated for each point at which there's a sensor cut (using interpolation to determine the position of the target track).

Note

Debrief is able to produce a plot of multiple sensors against on track, or the range from one sensor to multiple tracks - but it can't calculate the ranges for multiple sensors against multiple tracks. That would be just ridiculus. Your brain would explode, honest.

2.4. Managing ambiguous data

Whilst hull-mounted sensor typically produce a single bearing to their contact, towed-arrays typically produce ambiguous bearings. They are aware of the relative bearing to the contact, but are now aware of whether it is to the Port or Starboard of the host platform.

The Debrief sensor format (DSF) handles ambiguous data by allowing two bearings to be read in, and once that block of sensor data is made visible Debrief plots both sensor bearings.

In order to analyse the sensor data, or use the sensor data in other analysis, the analyst must decide which is the actual bearing (to be kept), and which is the ambiguous bearing (to be removed).

Once a decision has been made on which of the two bearings to keep, open that block of sensor data in the Outline View. Then, select the relevant sensor cuts (using <shift> and <control> as necessary to multi-select). Once selected, right-click on one of the items and select Keep port bearing or Keep starboard bearing.

2.5. Exploiting doppler calculations

To support the display frequency residuals (as used in Section 2.3, “Dragging tracks”), Debrief contains a set of doppler frequency calculations (see Section 3.1, “Frequency algorithms”). Should you require this data in a third-party application it is possible to export this calculated data. Do this export as follows:

1. You must have a Debrief plot open that contains ownship plus target tracks
2. Ownship must be marked (Section 1, “Assigning tracks as primary and secondary”) as primary track (see Section 1.1, “Tote area”), and the target track must be the only secondary track

3. The ownship track must contain sensor data

4. The target track must have its base frequency assigned.

5. Using the Time Controller view (see Section 2.1, “The Time Controller”), select the time period for which data is to be exported

6. Select Export Doppler Shift data

The data is collated using the following algorithm:

1. Loop through the visible blocks of sensor data for the ownship track

2. Find all target track segments that overlap with this block of sensor data

3. Loop through this block of sensor data

4. For each sensor cut, find the position on the target track nearest to this time

5. Perform a doppler calculation for this sensor cut/target position

The data will now be written to file in csv format: first the base frequency then a series of time-stamped measured and predicted frequencies.

2.6. Multi tonal frequency analysis

There exists the situation in frequency analysis where multiple tonals are held, but the analyst would rather use the highest freq tonals. Here's a strategy for how to manage it,

First, load the bearing sensor data. Edit the sensor cuts (probably in the grid editor, Section 8, “Using the Grid Editor”) to include the highest freq data held, potentially colouring the blocks of cuts according to the base tonal that they relate to. Go through these blocks of cuts creating a solution for each block - then setting the base frequency for that solution to the respective value.

You can now switch individual solution blocks on and off, ensuring the stacked dots (Section 1.3, “Track shifting”)will be working with the correct data values.

2.7. Multipath analysis

Should you have access to multipath sensor data of a target vessel from a host platform, Debrief is able to assist in determining the target depth. In overview, Debrief allows you to compare the measured time interval with one calculated from the relevant dispositions of the two tracks (plus measured sound speeds for that location).

2.7.1. Datasets

To perform this analysis you require the following datasets:

1. Ownship and Target tracks

2. A SVP (Sound Velocity Profile) file for the location (using the file format defined in Section 5.2, “SVP file”). The maximum depth of the SVP is interpreted as the depth of the water column: the depth slider is uses this value as its maximum, and it is used as the maximum permissible value in the optimisation algorithm.

3. A file of measured time intervals (using the file format defined in Section 5.3, “Time delays file”)

2.7.2. Time delay algorithms

The indirect path length is calculated as follows:

$$L_d = \sqrt{H_d^2 + (z_s - z_r)^2}$$

$$L_{s(horiz)} = \frac{H_d * z_s}{z_s + z_r}$$

$$L_s = \sqrt{L_{s,horiz}^2 + z_s^2}$$

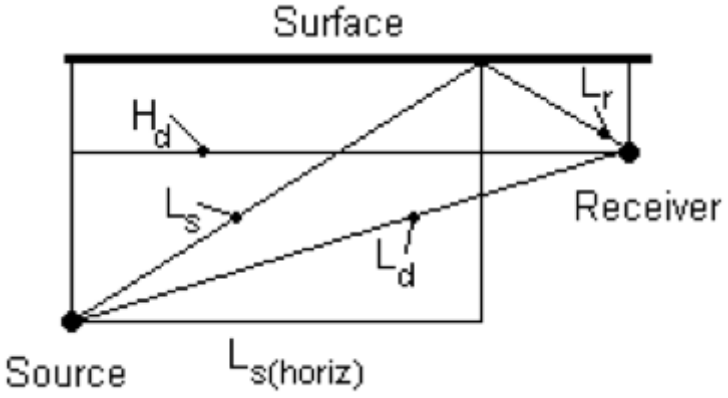
$$L_r = \sqrt{(H_d - L_{s,horiz})^2 + z_r^2}$$

Algorithm used to determine indirect path length

Where:

L_d	Direct path length
$L_{s,horiz}$	Horizontal range to the point where the surface path intersects the surface (see diagram below)
L_s	Path length from source to surface
L_r	Path length from surface to receiver
H_d	Horizontal range
Z_s	Source depth
Z_r	Receiver depth

These lengths are the calculated as:



Comparison of acoustic paths

Finally the path lengths are divided by the respective sounds speeds to obtain the path travel times, and this the overall time delay:

$$Time\_delay = \left(\frac{L_s}{c_s} + \frac{L_r}{c_r} \right) - \frac{L_d}{c_d}$$

Calculation of time delay

2.7.3. Sound Speed algorithms

The sounds paths shown in the illustration above may travel through depths that have varying sound speeds. We calculate an average sound speed for the depth profile by calculating a weighted mean for that depth range:

Source depth = 55 m

Receiver depth = 30 m

Sound Speed Profile:

Depth (m)	Speed (m/s)
0	1500
30	1505
40	1510
60	1515

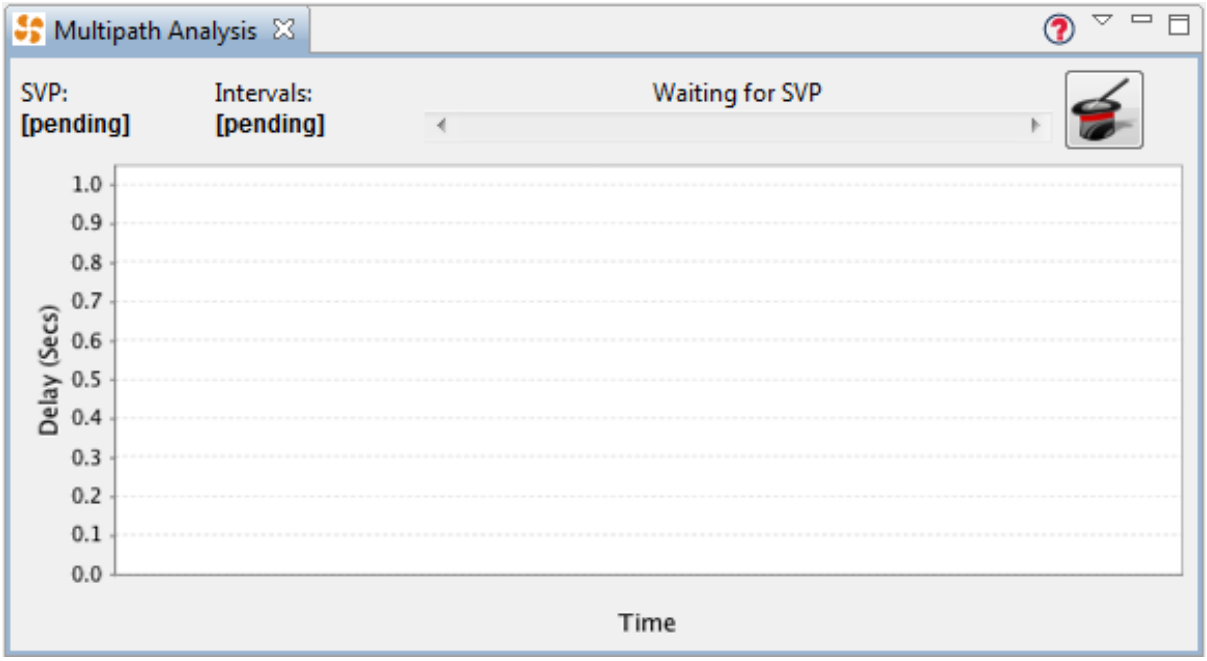
Average sound speed between 0 m and 30 m can be ignored as the sound never goes into that region. Average sound speed between 30 m and 40 m is 1507.5 m/s. The sound speed at 55 m is 1513.75 m/s (through linear interpolation). Average sound speed between 40 m and 55 m is 1511.875 m/s. Using the depth range in metres as its weighting value, the overall average sound speed is ((1511.875*15)+(1507.5*10))/25 = 1510.125 m/s

2.7.4. Optimisation Algorithm

Debrief is able to use Dr Michael Thomas Flanagan's (at www.ee.ucl.ac.uk/~mflanaga, [note: external link]) Nelder Mead simplex optimisation algorithm to determine the optimum target depth to minimise the least-squares error between the calculated and measured curves.

2.7.5. User interface

Perform multi-path analysis using the Multipath analysis view. Open this view by selecting it from the Window/Show View menu.

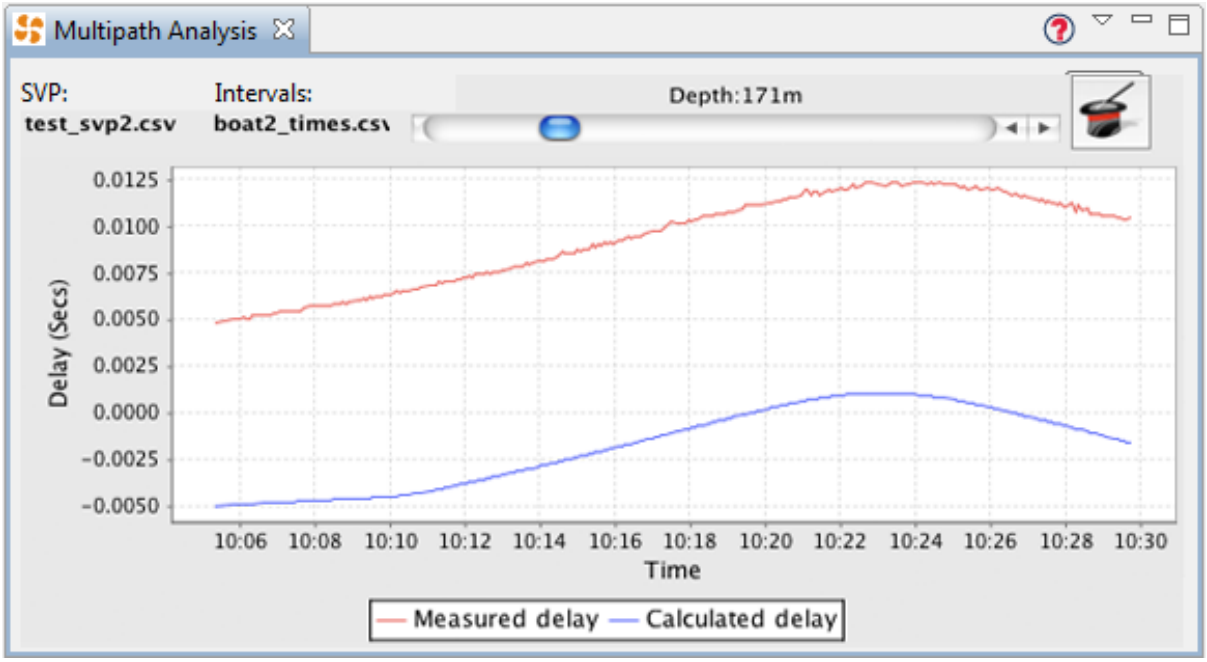


Initial view of multipath analysis view

As you'll see from the above screenshot, the panel is laid out with a series of controls above an xy plot. The controls at the top of the view indicate the files being used for SVP and time-delays, a slider control is provided to let you trial target depths, plus there's a *magic* button that runs an optimisation algorithm - generating a *best-guess* of target-depth by minimising the time-delay error.

So, start off by dragging in your SVP and time-delay files - dropping them over the respective *[pending]* label. Next, ensure your 'ownship' track is marked as the current primary track, and your target track is the secondary track in the Debrief Track Tote (as explained in Section 1.2, “Assigning tracks”).

Once you've loaded the data and assigned the tracks, the slider control will become enabled and you can drag it to trial new target depths. The xy plot will update to show a comparison of the calculated time delays against the measured delay. You'll determine the target depth estimate by using the slider to place the blue (calculated) line as close as possible to the red (measured) line.



Multipath analysis view with data loaded

An alternative to working out the estimated target depth by eye/experimentation is to click on the magic top-hat icon. A Nelder and Mead simplex optimisation algorithm [http://www.ee.ucl.ac.uk/~mflanaga/java/Minimisation.html] [note: external link] will now run - with the slider set to the result on completion. Should you be interested in the performance of this iterative optimisation algorithm, a summary of each step is placed as an information message in the error log (viewable by selecting Window/Show View/Error Log). The log will show the 'score' for each target depth trialled - normally only around 10-20 depths are necessary to reach an optimum.

Note, during development, on occasion the optimisation algorithm failed to produce a realistic target depth. On each occasion this was a symptom of there not being a valid result in the available target range (0 to 1000m) - which itself was symptomatic of invalid data files: dropping the data-files into the wrong 'slot' or mistakenly using units different to those specified in the reference (see Section 5.3, “Time delays file”).

Chapter B.11. Management of TMA and TUA solutions

1. TUA data

1.1. Introduction

TMA algorithms are used to produce an estimate of range by analysing a sequences of sensor contacts which only contain bearing (and optionally range). The output of the algorithm is typically a series of estimated target locations with optional estimates for course, speed and depth. Uncertainty in bearing and range may be indicated by representing the target location as an ellipsoidal Target Uncertainty Area (TUA).

TUA contact data is always related to one of the currently loaded tracks, and may be represented either as an absolute location (at the centre of the ellipse) or as a range and bearing from the nearest point on that loaded track. The strategy for use of absolute versus relative data is described earlier in Section 1.5, “Preparing Sensor Data”.

1.2. Loading TUA data in bulk

TUA data is loaded into Debrief in REP files, just like any other Debrief data. The line format is:

```
;TMA_POS: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
TT...TT OOO.O  XXXX YYYY  CCC SSS DDD xx.xx
;; date, time, ownship name, symbology, tma lat, tma long, track name,
ellipse orientation (deg from north),  maxima (yds), minima (yds),
course, speed, depth (m), label string

;TMA_RB: YYMMDD HHMMSS.SSS AAAAAA @@ BBB.B RRR.R TT...TT OOO.O
XXXX YYYY  CCC SSS DDD xx.xx
;; date, time, ownship name, symbology, bearing (deg), range (yds),
track name, ellipse orientation (deg from north),  maxima (yds),
 minima (yds), course, speed, depth (m), label string
```

Note

There are two annotation format to represent TUA solutions (TMA_POS and TMA_RB). TMA_POS is used to define a TMA solution at a particular location, and TMA_RB is used to define a TMA solution at a specific range and bearing from the current ownship location at that specific DTG. Where a solution ellipse is not known the orientation, maxima and minima values can be represented by a single NULL value.

1.3. Loading small amounts of TUA data

An alternate route to loading TUA data is to use the 'Generate TUA Ellipse' wizard. Right-click on a Track or the Solutions layer inside a track, and select the respective menu item. The time for the ellipse is taken from the Time Controller, though you may type in any time. Follow the steps, and when the wizard closes you will have created your TUA ellipse. Clearly, if you have lots of TUAs to load, you're better off by loading the data into Debrief via a data-files as described above.

1.4. Relative data

When (as described above) Debrief plots data using a range and bearing, it follows the following procedure:

- The first time the TUA solution is plotted, it examines its parent track to find the vessel position nearest to (or greater than) the TUA solution DTG.
- The TUA solution then calculates the position of its centre relative to this origin

1.5. Analysing TUA data

When first loaded, TUA data is not made visible, since with any reasonable volume of TUA data the plot quickly becomes illegible. TUA data is switched on and off individually by accessing the whole TUA track, via its Track, from the Outline View.

It is once in *snail trail* mode that TUA data is most easily analysed. When in snail mode the Snail display mode performs the following processing:

- For each Track being plotted, the display mode looks to see if it contains any TUA data.
- It then examines each list of TUA data to see if it's visible. If it is visible, it plots the current TUA solution (nearest to the *Tote* time), followed by the TUA solution as disappearing solutions running back through the length indicated in the TrailLength parameter in the properties window.

2. TMA Management

2.1. Introduction

In addition to being able to display TUAs from third-party TMA implementations, Debrief can be used to derive fresh TMA solutions. These solutions are not presented in the form of TUAs, but in the form of analysis legs: a time-limited track segments comprising a series of points at constant course and speed. Traditionally these segments are created by analysis of a combination of sonar-derived bearings and frequencies.

These TMA segments are represented as track segments within Debrief for which the course and speed of the whole track (solution) are editable via the Properties Window.

The workflow for creating and manipulating TMA segments is covered in the *Single Sided Reconstruction in Debrief* tutorial in the Debrief online help.

2.2. Generating track segments

Debrief allows you to generate a segment of track data from a block of sensor data - replicating the process conducted on warships. This track segment adjusts dynamically to reflect the sensor location changing (such as if the sensor offset is edited - see Section 1.4, “Sensor offset lengths editor”). In the absence of sensor data it is possible to generate a segment of target track from a set of ownship positions. Data to support these operations is collected via a wizard, as described in the above tutorial.

2.3. Dragging tracks

Once TMA segments are present on the Debrief plot they are manipulated using the Drag Track Segment toolbar button  . Once this button is pressed Debrief automatically displays the Bearing Residuals view, which displays bearing residual data - subject to the constraints listed earlier in Figure B.2.2, “Stacked Dots view” (with the TMA Segment present as the secondary track). If your data contains frequency data then you can open the Frequency Residuals view, which display actual, and calculated frequencies (based on the selected solution). Read more about the frequency algorithms in Section 3.1, “Frequency algorithms”.

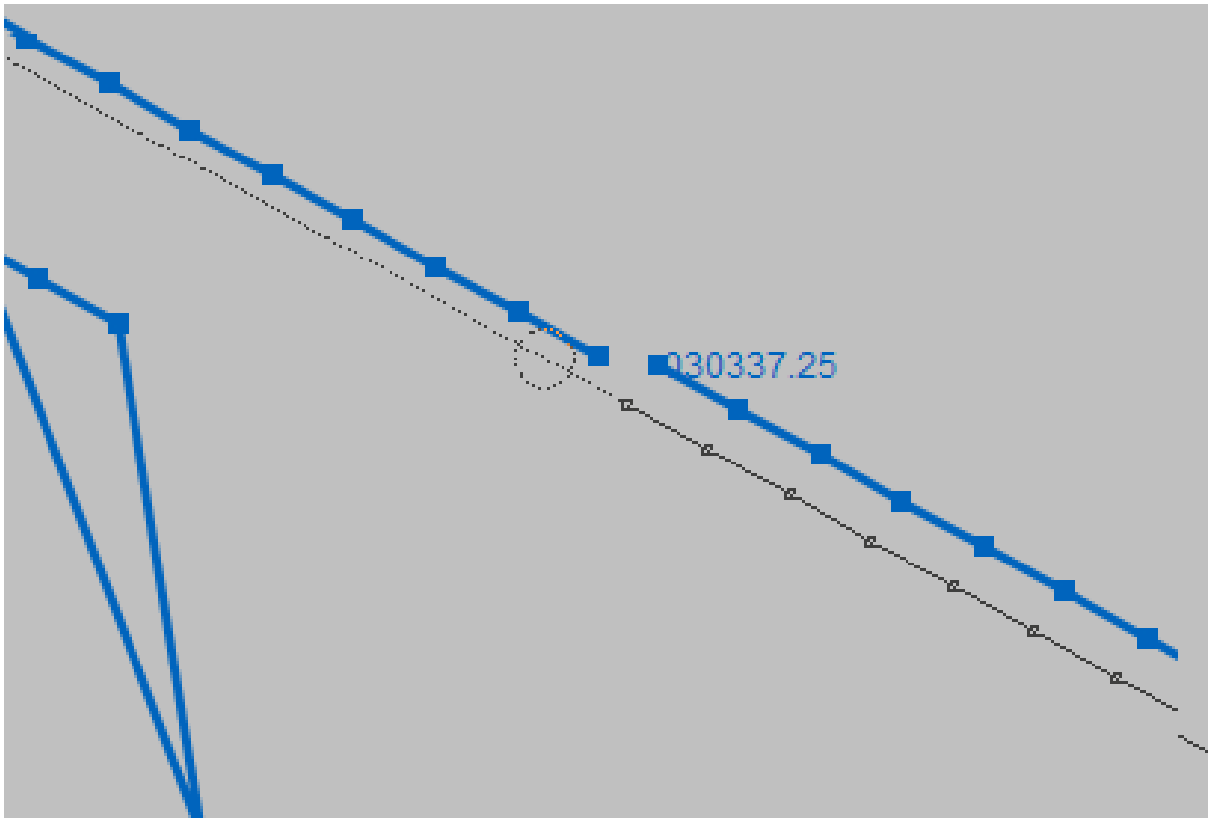
Once the plot is in 'Drag track segment' mode a series of track-dragging modes are enabled - with the current mode selected via the control buttons on the Stacked Dots plot. The control buttons provide the following modes:

Translate	Drag the whole track around, changing range and bearing from the source, but maintaining target course and speed
Rotate	Drag one end of the track, maintaining the track length (target speed) but changing the target course
Stretch	Drag one end of the track, maintaining the target course but changing the track length (target speed). When the centre-point of the track is dragged, the track moves in and out stretching/contracting as necessary to adhere to the start/end points on the host platform bearing fan.
Shear	Drag one end of the track, changing both the target course and speed

Note

Note, only the Translate and Rotate operations are available for all track sections. Only the Stretch and Shear operations are suitable for application to straight-line TMA Segments Thus, the hand cursor will only turn green over straight-line TMA Segment hot spots.

When in Translate or Rotate mode, the drag-highlight is extended beyond the first/last point in the track to assist with alignment. A target-circle is also plotted on this extended stalk. This target is plotted at a distance equal to the distance back to the end-but-one point. Thus, if you have equi-spaced track data this target can be used to align the track sections longitudinally and transversely. In the following diagram the lower-right track segment is being dragged inline with the upper-left track segment. Place the circle/cross hair over the end point of the fixed track to align the tracks in both dimensions.



Note, when the plot is in one of these modes, the operations apply to the ends of the track - so the operation is started by dragging one end of the track (once the 'selected' hover highlight is shown). In

addition to the specific operations performed at the ends of the target track, all modes support picking up and dragging the track when it is picked up at its centre In the Stretch mode the track shifts as described above, in all the other modes it is translated.

2.4. Combining tracks

Collectively, TMA segments are combined into target tracks using the Group and Merge operations described in Section 7, “Grooming track data”.

3. Semi-Automated TMA generation

Many Debrief users have high-performance workstations that have sufficient processing power to produce an initial best-guess at a solution for target tracks based on bearing data. In 2012 talks started regarding the delivery of such a capability to analyst usesrs. Significantly, it was perceived that an automated approach would benefit from multi-leg engagements, engagements that are typically challenging using the existing Debrief manual TMA capabilities.

Following exploratory discussions with Atlas Elektronik UK regarding the FRITMA tool it was determined by the relevant stakeholders that a custom-built TMA algortihm would be a viable solution for the requirement. This new Debrief feature was to be called Semi Automatic Track Construction (SATC)

Note

Further detail regarding the strategies and algorithms used in SATC are found later in this document, in Chapter D.5, *Semi Automated Track Construction (SATC)*

3.1. Bearings Only TMA

The concept of Bearings Only Target Motion Analysis, or Bearings Only Tracking (BOT) is an established field of mathematics. An easy intro to this is covered in this users online guidance [http://mpet.freeservers.com/dc_tma1.html] for the 688i game. This system utilises a variant on that concept. Whilst BOT focuses on the production of a near-real time estimate of the current state (location, course, speed) of the subject vehicle, this system performs the process time-late, aiming to produce a whole segment of vehicle track. The time-late nature of the undertaking (possibly several weeks later) means that an analyst has the opportunity to research other information about the vehicle track to contribute to the algorithm. This other information may be through acoustic analysis, supplementary documentation, or third-party sensors.

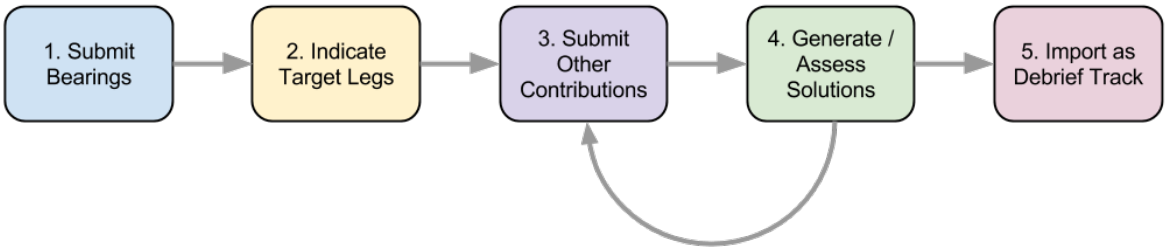
3.2. Glossary

Hard Constraint	A set of parameters that any target solution must meet - for example an analyst may know the range of speeds of a class of vessel sufficient to say "any solution track for this vessel must be within 2 and 9 knots".
Estimate	A subjective contribution from an analyst, such as "I believe the vessel is travelling at 4 knots". The SATC will consider values other than the estimate, but solutions that match the estimate will be favoured.
Contribution	A piece of information that is used in development of target solutions. This may be a measurement (such as range, bearing or frequency) data, an analyst forecast (a prediction, such as speed range), or one of a hidden set of analysis contributions that deduce further constraints based on other contributions.
Scenario	The collection of information and knowledge that is used to produced target solutions
Leg	A period of time for which it is known that the target vessel is either on steady course/speed (Straight Leg) or is manoeuvring (Altering Leg)

Precision	An indication of the level of fidelity to be considered by the SATC algorithm. For the LOW, MEDIUM, HIGH precisions the algorithm is allowed to run for 5, 15 and 30 seconds resp. For these precisions, the number of solutions considered for each processing cycle are 45, 70 and 120 resp.
Solution	A single permutation of target track, comprised of a series of legs - both altering and straight.

3.3. Overview of SATC

A series of meetings with Dr Iain McKenna (BAE Systems) led to a strategy where informed analysts would be able to submit a series of contributions representing knowlege they held regarding a vessel engagement. This flow of knowledge and interaction with the algorithm is as recorded below



SATC Sequence

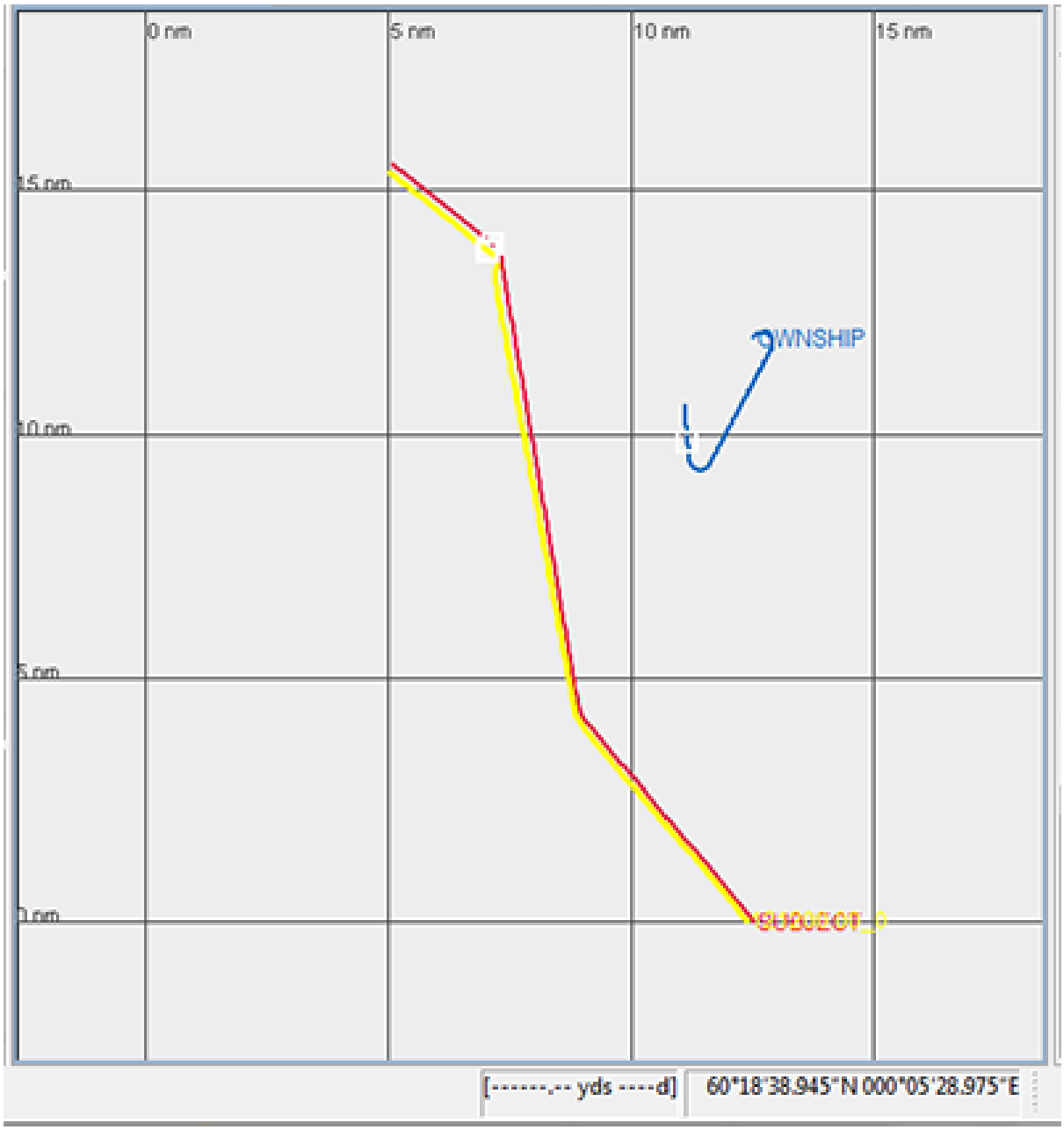
1. Submit Bearings	Bearing data lies at the core of the SATC algorthm, proving one of the most useful constraints on potential target track. The solution generation process is started by submitting a block of bearings to a solution.
2. Indicate Target Legs	The second-most important contribution is the knowledge of time periods where the target is believed to be travelling in a straight line. Again, this greatly reduces the number of candidate solutions that need to be considered.
3. Submit Other Contributions	Once the first two contributions are in place, the analyst is then able to submit his knowledge/hypotheses regarding target course, speed, or the range to the target. These forecasts can relate to the whole target solution, particular legs, or even discrete periods of time between those legs.
4. Generate/Assess Solutions	The analyst can then trigger the generation of a solution. The solution is optimal in regard to the contributions that the analyst has submitted. If the solution does't match the analysts expectation, or if it just doesn't look right, then the analyst can submit further contributions, as appropriate.
5. Import as Debrief Track	Ultimately, if/when the analyst is happy with the solution produced he can import the solution into Debrief, either as a collection of legs (to be used in the existing manual TMA process), or directly as a conventional Debrief track.

Internally, the SATC process runs in two stages. The first considers hard-constraints that the analyst has recorded for the engagement. These constraints are intelligently fused into a consolidated body of constraints, one set for each potential target location (one location is produced for each bearing measurement). This phase is able to run very quickly, since it just deals with hard facts. It is possible to view the set of constraints changing live on the Debrief plot whilst the hard-constraints are manipulated.

The second stage is where candidate solutions are produced. This stage can be quite time-consuming, so it is only run on analyst request. The stage produces an imaginary grid over the potential start and end positions for each straight leg. The algortihm then looks at the possible straight courses between combinations of these points. An optimisation algorithm reduces the number of permutations that need to be considered. The optimisation algorithm also discards solutions that, whilst mathematically optimal, do not represent achievable successive courses (due to simple speed/time calculations).

SATC Process

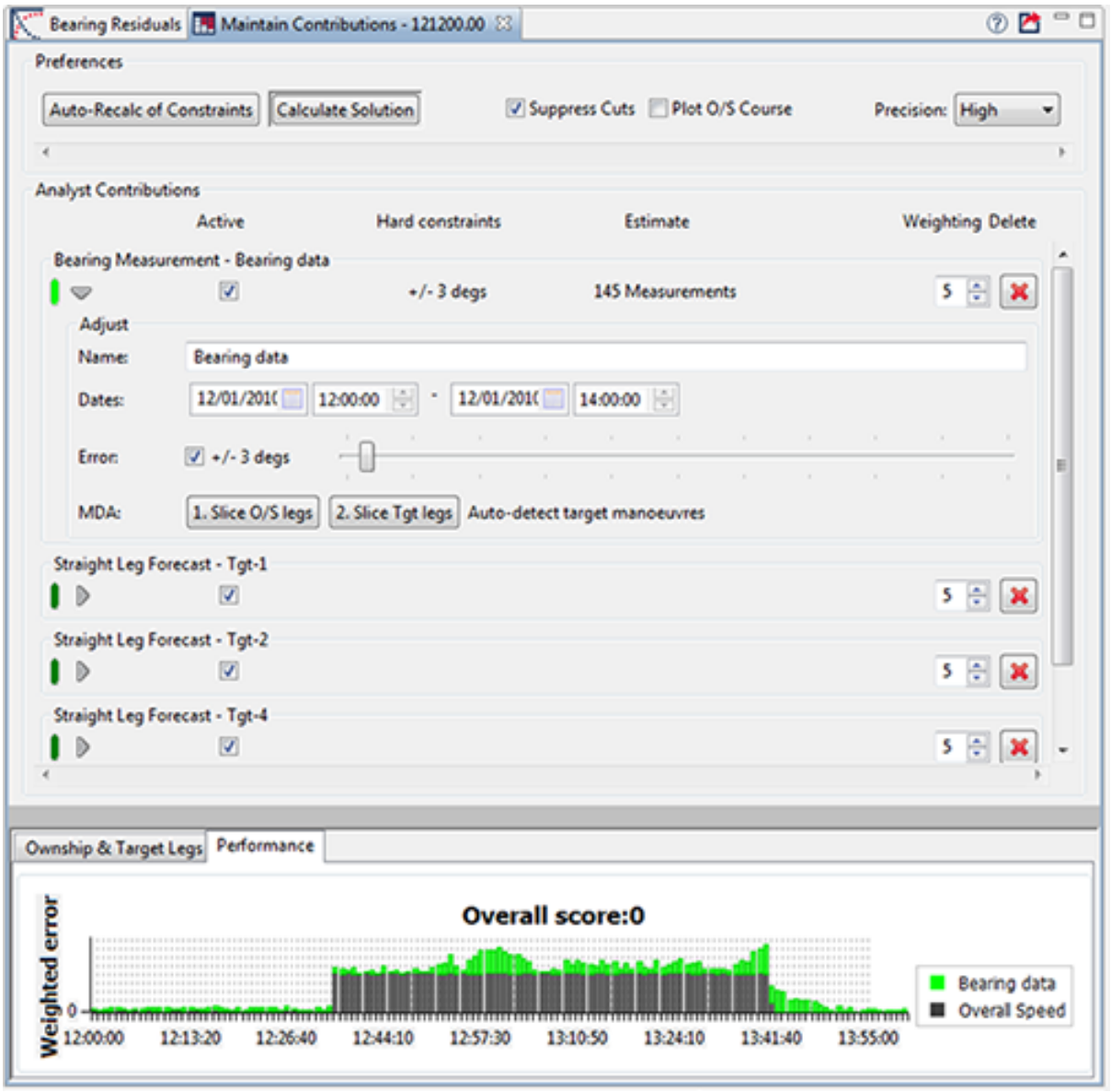
The following screenshot shows a TMA scenario using simulated data. The ownship track is shown in Blue, the truth target track is shown in Red, and the generated solution is shown in Yellow. The range error between truth and generated track is shown in the Range vs Time Plot at the foot of the image. The scenario is particularly challenging, given that there are more target manoeuvres than ownship manoeuvres, the target is quite distant, and the bearing rates are relatively low.



Sample solution

3.4. User Interface

SATC scenarios are managed from within the Maintain Contributions view within Debrief. This view opens automatically when a new scenario is created.



Maintain Contributions View
The following sections describe the three parts of the Maintain Contributions View

Preferences. This panel provides overall control of the scenario.

Auto-Recalc of Constraints	This toggle button tells SATC whether it should recalculate the constraints each time a contribution is changed
Calculate Solution	Actually calculating the solution can take up to several minutes, so the overall solution is only calculated when this button is pressed
Suppress Cuts	Whether to allow Debrief to Suppress the least significant cuts. Greater precision settings allow suppression of fewer cuts.
Precision	Used to indicate which of the three precision levels are required in solution generation.

Analyst Contributions. This contains a list of the contributions provided by the analyst. For each contribution, this data is present:

Name	Each contribution is presented inside it's own screen object. The contribution name is provided at the top-left of this block

Hard Constraints	Some contributions have hard constraints. In the earlier screenshot you can see that the Speed Forecast has hard constraints of 5-15 knots. No solution will go outside these bounds.
Estimate	Where an estimate is provided, it is shown here. For the Speed Forecast in the earlier screenshot, you can see that an estimate of 13 knots has been provided. So, solutions will be favoured if they have a target speed nearer to 13 knots.
Weight	The weighting is a way of the analyst tuning the TMA process. Higher weightings figure more strongly in the generated solutions.
Delete	Use this to delete a contribution. It will also disappear from the Layer Manager.

Performance. The performance graph is an indication of progress of the SATC algorithm, as a series of states along the whole solution. For each timed state object the graph shows one or more bars. A bar is shown for each contribution that is capable of calculating solution error - with some error scores shown per state (such as for Bearing Measurement Contribution) or an error shown for a whole leg (such as for an estimated leg speed). The error shown is normalised to a zero-one value within the allowable min/max range (or bearing error for bearing data), then scaled according to the weighting value for that contribution.

Analysis support. In the toolbar for the Maintain Contributions view is a button to export a summary of the scenario to the clipboard in CSV format. This format includes a series of x,y, date-time, course, speed ownship state measurements, followed by a series of time-stamped bearings. The X/Y and date-time have been made relative to an arbitrary location to hide any potentially sensitive data. Access to the data eases the task of externally analysing the scenario.

Chapter B.12. Support for DIS Protocol

1. Introduction

One group of users employ Debrief for the analysis of scenario results from a headless simulation engine. The simulation engine produces results data in a range of output formats. Historically the users had to convert an existing output format to Debrief `.rep` format, but in the late 2000s direct support for Debrief was added.

In 2015 this body of users recognised the utility of using Debrief as a live simulation monitor, to bring these benefits:

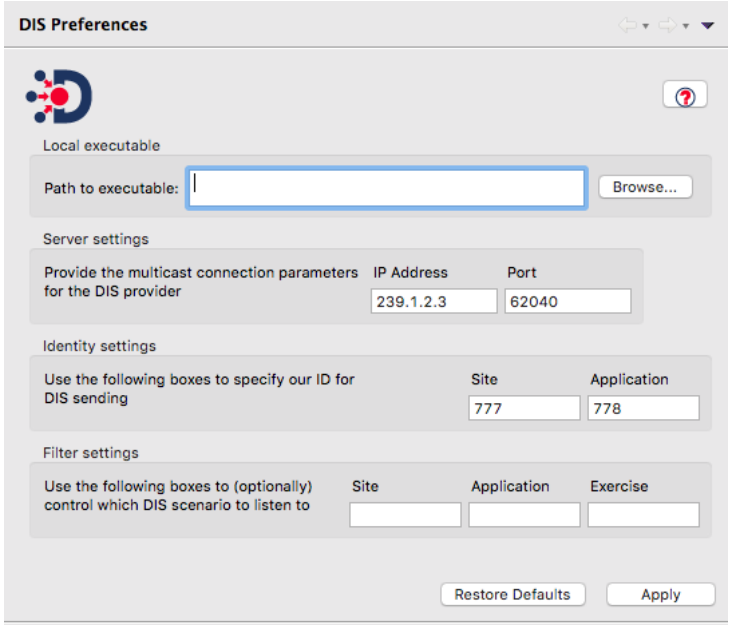
- Verify scenario participants behave as predicted before committing to lengthy Monte Carlo block of simulation runs
- Develop the behaviour of one or more scenario participants in a programming-like CODE-RUN-DEBUG cycle

The *DIS* standard was adopted for network communications between Debrief and the headless simulation engine: `[https://en.wikipedia.org/wiki/Distributed_Interactive_Simulation]`.

You can find System documentation for the DIS integration later in this document (Chapter D.6, *System Documentation for DIS integration*)

2. Configuring DIS

Access the DIS preferences page either from the Window / Preferences / Maritime Analysis DIS preferences page, or by clicking on the Server prefs link on the DIS Listener View.



2.1. Network configuration

Debrief uses DIS to talk to simulators running on the current machine, or on another machine on the network. This network communications requires some configuration - to tell Debrief what to listen to. Two specific values are used:

IP Address — The 4-block Network address for the multicast group being used for DIS messaging

Port — The number of the port on that address which is being used for DIS messaging

2.2. Configuring the launch of an external simulator

Some DIS simulators run as complex combinations of services & applications. Other simulators are contained in simple standalone executables. It is possible to launch such an executable directly from Debrief

Path to executable — The application (or script) used to trigger a simulation run

2.3. Giving Debrief an identity

In an environment where a networked simulation has multiple participants, it can become necessary to specify identify particular participants (so their messages could be, for example, ignored). Two particular values are used for this configuration

Site — This value is unique to the site (location) where Debrief is being run from. The value isn't derived from a formal table, but is by local agreement

Application — This value is unique to the Debrief application.

2.4. Filtering DIS traffic

It is possible for multiple DIS simulations to simultaneously run on a single network. Common-sense would suggest that these multiple simulations should use unique network address and port, but this isn't mandatory. If multiple simulations are being run on the same network address and port, it is possible to control which simulations Debrief listens to:

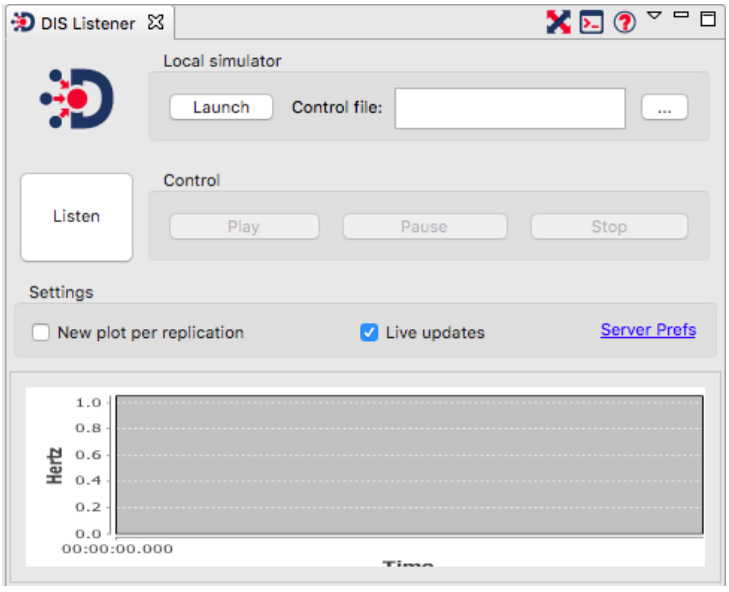
Site — Provision of this optional parameter restricts Debrief to handling messages from the specified site

Application — Provision of this optional parameter restricts Debrief to handling messages from the specified application

Exercise — Provision of this optional parameter restricts Debrief to handling messages from the specified DIS exercise

3. Monitoring a DIS simulation

Interaction with DIS scenarios is performed from the DIS Listener View.



At the top of the panel are two small icons. The first has a toggle behaviour, and when ticked Debrief will auto-resize at each step, to ensure all loaded data remains visible. The second icon opens the Console view (to keep track of messages returned by the launched simulator - see more at Section 4.3, “Track the simulator using the Console”), and the final one accesses the help for Debrief's DIS integration.

3.1. Running a local simulator

If you have specified the path to an executable in the Server preferences (see Section 2.2, “Configuring the launch of an external simulator”), then you can use the Local simulator box to (optionally) specify the location of the input file to be passed to the executable. Click on the ... button to open a file browser.

Once a control file has been specified, you can click on the Launch button to start the simulator executable, passing the input file to it. After launching Debrief will automatically start listening on the correct network port.

Tip

A quick way of setting the input file is to drag one from the Navigator view oneo the path. This will update the input file path to the one dragged in.

An even quicker way of running a different input file is to right-click on a `*.inp` file in the Navigator view and select Run in simulator. This will launch the simulator, passing it that control file. Do this if you have lots of different input files in your workspace.

3.2. Interacting with a DIS simulation

The large listen button is used to toggle whether Debrief is listening for DIS messages.

If Debrief receives sufficient network status messages from the simulator, Debrief will enable the Play, Pause, Stop. After Stop has been pressed, and new DIS messages received will be put into a new plot (existing data will be wiped).

3.3. Other settings

Two other minor options are provided. If New plot per replication is ticked, when Debrief recognises that data for an new simulation run is being received, Debrief will open a fresh plot to contain the data.

Note

If the current plot is based on a `.REP` input file, Debrief will base the fresh plot on that same input file. This practice can be used to design a set of background annotations (such as exercise area, scale, bathy backdrop) - which gets used for all plots.

If the simulator is running very quickly, and it looks like the UI is getting swamped with updates, the Live updates checkbox can be cleared, which will let Debrief continue to receive and store DIS messages, but Debrief will not bother updating the user interface (the Outline view or the Plot Editor).

Lastly, the Server prefs link can be used to open the preferences page for Debrief DIS communications.

3.4. Logging DIS messages

On the DIS Listener view's drop-down menu is a Log Messages toggle button. When this is selected, Debrief will log all messages to the Debrief system log. This log can be accessed via the Window / Show View / Error Log menu option. The `.log` file is chunked into 1Mb files, and is stored in the hidden `.DebriefNG/.metadata` folder within your home directory. To remind the user about this logging (with its associated performance and disk usage penalty), the Scenario Complete popup dialog includes a reminder that logging is active, when relevant.

3.5. Performance graph

At the foot of the DIS Listener View is a performance graph. When a simulation is being listened to, this graph updates once per second. It shows two lines. One is the frequency at which simulation messages are being received, and the second is the frequency of screen updates. As more data is received, and shown on the plot, the frequency of screen updates will slow down.

Inspection of this graph can give the analyst an indication of whether the simulator is running, how quickly it is running, and how Debrief is handling the volume of loaded data.

4. DIS Tips & Tricks

4.1. Specifying format parameters in advance

In early 2016 Debrief received the ability to provide format instructions in a REP file, as in the following example.

```
;FORMAT_FIX: 10_min_sym SYMBOL NULL NULL TRUE 600000
;FORMAT_FIX: 30_min_lab LABEL NULL NULL TRUE 1800000
;FORMAT_FIX: 15_sec_arr ARROW NELSON NULL TRUE 900000
;FORMAT_TRACK_NAME_AT_END: name_at_end
;FORMAT_LAYER_HIDE: hide_dis_6  DIS_6
```

Use of these instructions will save effort re-applying formatting to each new model run.

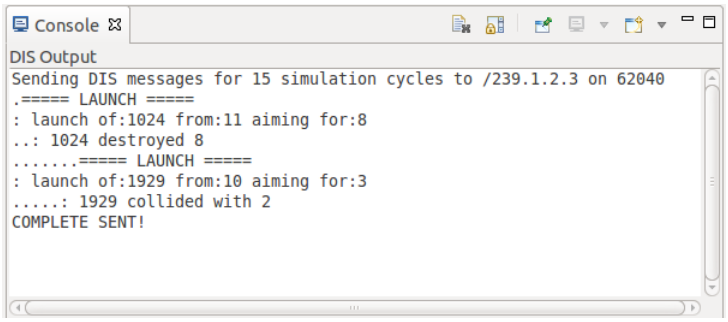
4.2. Running Debrief's DIS simulator

To aid development of the new DIS integration capability, a very simple simulator was developed that was capable of sending out messages in DIS format. This tool has now been included in Debrief, to help users get started with (or just play) with Debrief's DIS integration.

The simple simulator can be started using the script file that is contained in the `Sample Data / Other formats` folder. Just assign the `sender.sh` or `sender.bat` as the executable in the Server preferences page, and use `control.inp` as the simulation input file.

4.3. Track the simulator using the Console

If you simulator outputs message to the command line, you may be used to always running the simulator from a command prompt - so that you can track any error / status message returned. But, you can use the Show Console button in the DIS Listener View to open the console, as shown below.



Part C. Maintainer's Guide

This section will give you all you need to look after a collection of Debrief installations, from fault-finding through to optimising the installations used.

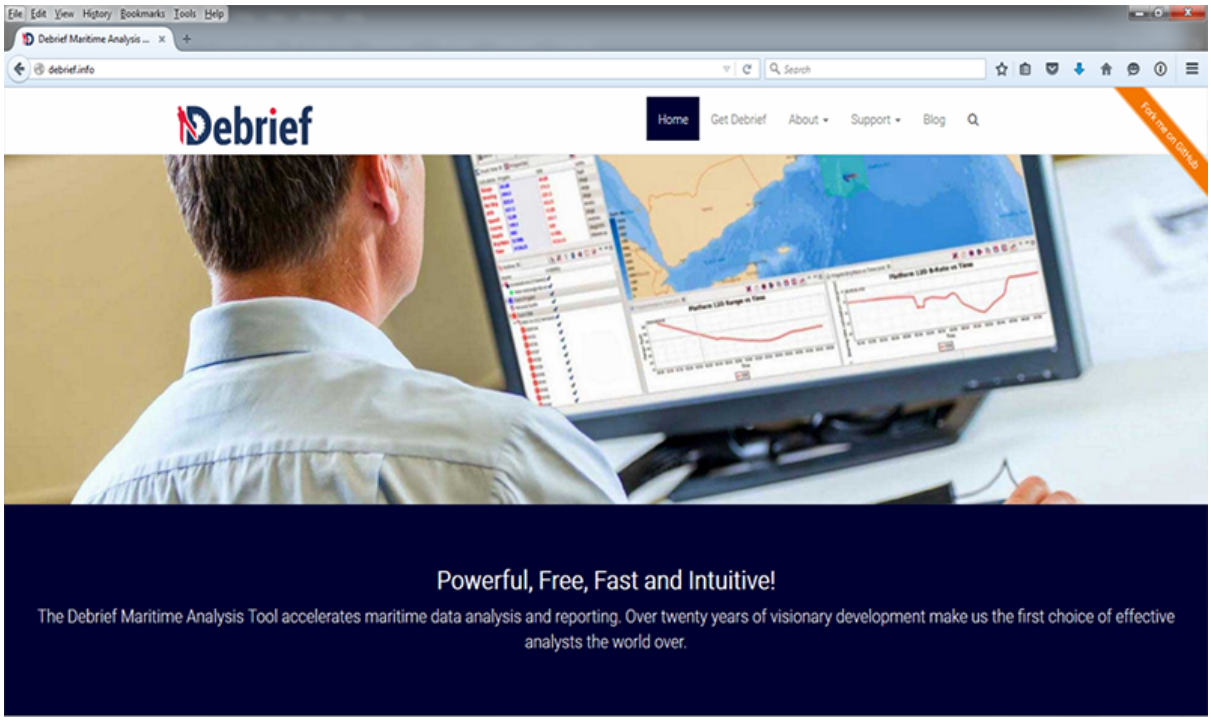
Chapter C.1. Participating in Debrief development

1. Debrief online

Whilst the use of Debrief originated at the *MWC*, it now has a very distributed community of users. Communication across this distributed user-group is enabled via the Internet through three mechanisms: the Debrief home page, the SourceForge project management page, and the Debrief news list. These mechanisms are described in this chapter.

The Debrief web-site, found at http://www.debrief.info provides an introduction to Debrief, together with access to this tutorial and a *way-in* to getting started with Debrief.

Figure C.1.1. Debrief home page



The SourceForge.net page, used to support the project management of Debrief, is described later in Section 5, “Providing feedback”, and the Debrief news list is described in the next section, Section 2, “Keeping up to date”.

2. Keeping up to date

It's easy to keep up to date with what's happening in Debrief. You can find out about new bugs, software changes, and formal software releases using one of these methods:

2.1. Twitter

Twitter fans just have to follow the @debrieftool [https://twitter.com/debrieftool] account.

2.2. Email

Alternatively, you can navigate to Debrief's *GitHub* site (https://github.com/debrief/debrief), and then either *Star* the page to hear high level announcements, or *Watch* the page to hear all the gory detail. Learn more about the options here [https://help.github.com/articles/watching-repositories].

3. Debrief at the Maritime Warfare Centre

This section provides guidance on maintenance of Debrief installations for users on the Maritime Warfare Centre network.

3.1. Installing Debrief

The IT Support department should be approached to conduct fresh Debrief installations. The most up-to-date Debrief installer is located at `//File-Share/Operational Analysis/collaborative/application/debrief/InstallDebrief.exe`.

3.2. Keeping your Debrief installation up to date

The development of the Debrief application is ongoing throughout the year. Occasionally particular users have new requirements, or bugs are found. Normally, only users working in related areas are informed about new features or minor bugs that only occur when performing that particular type of analysis. Where new features relate to users across MWC, or where a significant bug is fixed, e-mails are sent out to the Debrief user mailing list (see Section 2, “Keeping up to date”).

> **Debrief Users mailing list.** On the MWC network mailing lists have been set up to allow e-mails to quickly be addressed to a group of users. One such list is the *Debrief Users* list. When you start using Debrief it is worth contacting IT Support to have your name added to the list - then you know you will be informed of any significant changes to Debrief.

Don't worry about always having the newest version of Debrief. Most software changes only apply to a small portion of the software and don't justify the upgrade effort. If, however, a Debrief fix applies specifically to your work, you should follow these steps to upgrade your Debrief installation:

Searching for software updates

1. Select Help/Software Updates/Find and install.
2. Select Search for updates of the currently installed features, then Next.
3. Follow the instructions provided to view what updates are available, select any relevant ones (though you can't really go wrong in retrieving all of them), and install them. Easy.

Now, Debrief may now have found any updates. Hey, maybe there weren't any. But maybe Debrief's not looking in the right place. If no updates are found, follow these steps to ensure the MWC download server is configured:

1. Select Help/Software Updates/Find and install.
2. Select Search for new features to install
3. You will see a list of download sites. If there isn't one with a name you associate with MWC, select New Local Site....
4. A file dialog will open. From it, enter `Network Neighbourhood` , and navigate to `//Needles/Operational Analysis/collaborative/applications/Debrief/Updates`. Then press OK and name the local site as **`MWC Updates`**. Accept this. Now, from the list of update sites, ensure **`MWC Updates`** is ticked, but not the others (since they rely on an internet connection).
5. Now you should be able to perform the search for software updates again (see Searching for software updates).

4. Debrief across the Internet

This section provides guidance on maintenance of Debrief installations for users with world wide web (Internet) access.

4.1. Installing Debrief

Debrief can be downloaded via the downloads page at the Debrief site (http://sf.net/projects/debrief). Be warned, because the full install is getting on for 150 Mb in size. If you aren't able to perform such a download, or you need a formal copy for your IS/IT department please send an e-mail request to the project manager (see Table 1, “List of acknowledged Debrief users”), and a CD-based copy will be forwarded to you via snail mail.

4.2. Keeping your Debrief installation up to date

The development of the Debrief application is ongoing throughout the year. Occasionally particular users have new requirements, or bugs are found. Normally, only users working in related areas are informed about new features or minor bugs that only occur when performing that particular type of analysis. Where new features relate to a wide body of users , or where a significant bug is fixed, e-mails are sent out to the Debrief user mailing list.

Don't worry about always having the newest version of Debrief. Most software changes only apply to a small portion of the software and don't justify the upgrade effort. If, however, a Debrief fix applies specifically to your work, you should follow these steps to upgrade your Debrief installation:

Searching for software updates

1. Select Help/Software Updates/Find and install.

2. Select Search for updates of the currently installed features, then Next.

3. Follow the instructions provided to view what updates are available, select any relevant ones (though you can't really go wrong in retrieving all of them), and install them. Easy.

Now, Debrief may now have found any updates. Hey, maybe there weren't any. But maybe Debrief's not looking in the right place. If no updates are found, follow these steps to ensure the Debrief update server is configured:

1. Select Help/Software Updates/Find and install.

2. Select Search for new features to install

3. You will see a list of download sites. If there isn't one with a name you associate with Debrief, select New Remote Site....

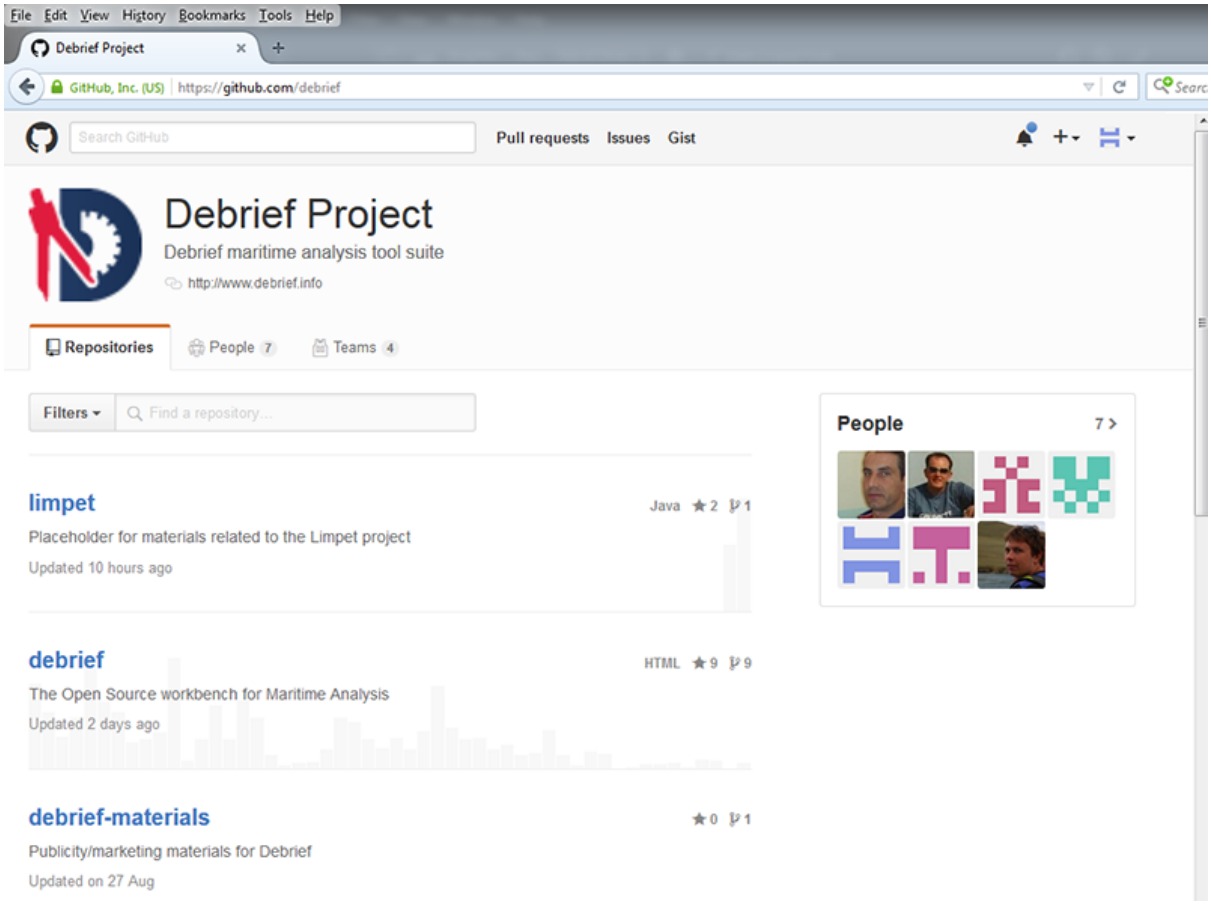
4. A file dialog will open. In it, enter `Debrief update site` for the Name, and `http://debrief.sourceforge.net/eclipse/` for the URL. Now, from the list of update sites, ensure **`Debrief update site`** is ticked, but not the others (since they rely on an internet connection).

5. Now you should be able to perform the search for software updates again (see Searching for software updates).

5. Providing feedback

In addition to the Debrief web-site, the Debrief project makes use of SourceForge and *GitHub*, online development web-sites that provide us with bug-tracking, file download support, and news groups. Since 2014, Github (https://github.com/debrief/debrief [https://github.com]) has taken over the bug-reporting and feature requests, with SourceForge (http://sf.net [https://sf.net]) just responsible for downloads.

Figure C.1.2. Debrief's home page at GitHub



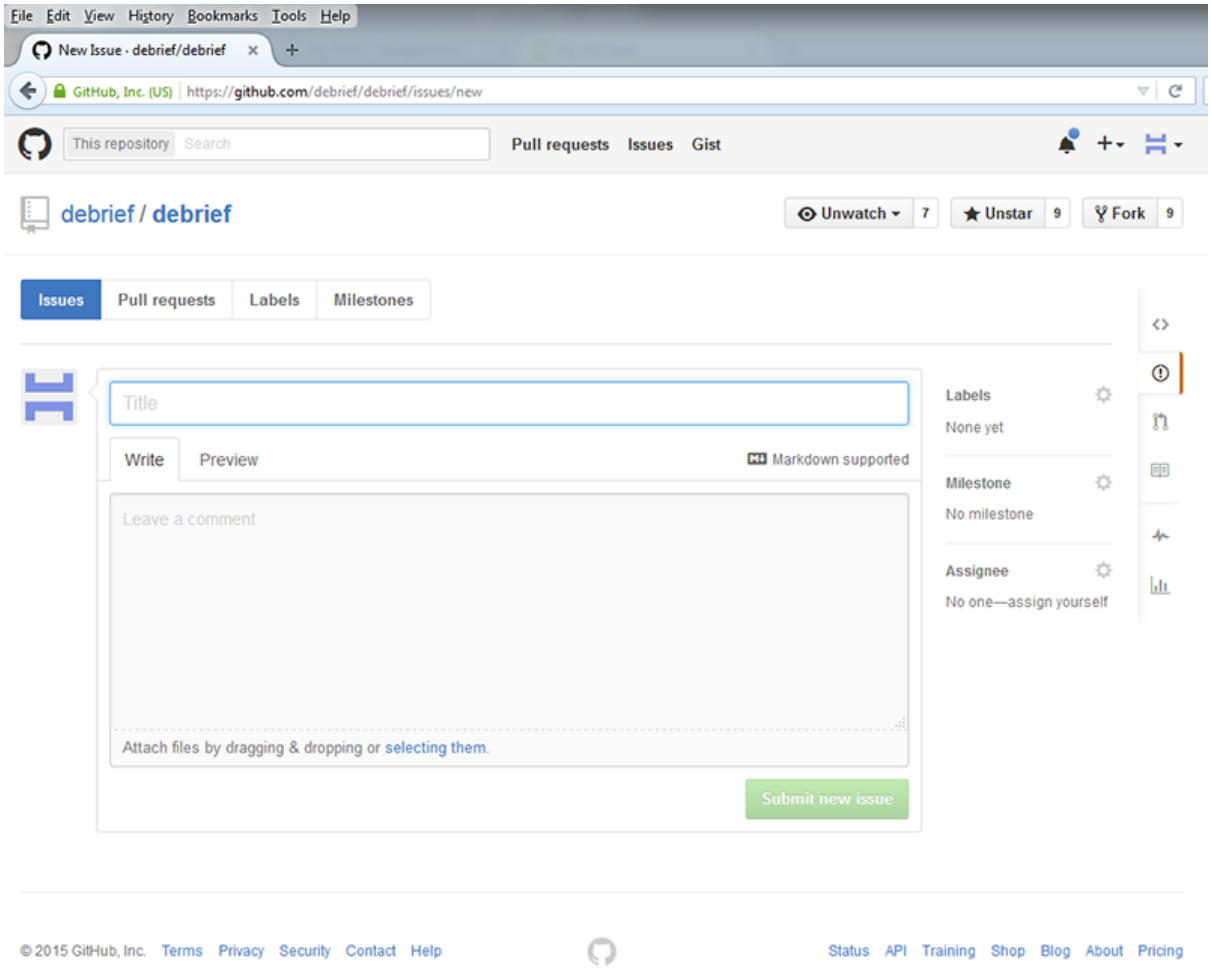
From the screenshot of the GitHub home page for Debrief you can see a list of the services it provides, ranging from discussion forums, through bug and feature-request trackers

5.1. Reporting bugs

To report a bug, or a feature which you believe is behaving incorrectly, you must first register at www.GitHub.com [https://github.com]. Once registered, you can navigate to the Debrief Issues [https://github.com/debrief/debrief/issues] page, click on the 'New Issue' button, and submit a request for a bug-fix or new features. The items are then tracked, and provided an e-mail address was provided, the submitter is kept informed of the progress of the submission.

The following screenshot shows the form used to submit bugs. Don't worry if you're not quite sure if the bug is actually intended behaviour, or if you're unsure whether a proposed new feature would 'make the cut'. Once your request is emailed it's recorded on the system and an informed debate/ discussion can follow. This ensures requests don't fall through the cracks.

Figure C.1.3. GitHub Issue Reporting page



Once you have submitted the report, you should receive an e-mail confirmation of the bug-report, followed by an e-mail from the Project Manager (see Table 1, “List of acknowledged Debrief users”), normally containing a time-frame for resolution of the bug.

5.2. Requesting new features

The procedure for requesting new features is much the same as for Section 5.1, “Reporting bugs”, described above. The time-frame for resolution of feature-requests is normally larger than for bugs, but if maintenance effort is being performed in a related area of Debrief, and the Project Sponsor (see Table 1, “List of acknowledged Debrief users”) is supportive, then the new feature should get added quite quickly.

Chapter C.2. Debrief maintainer's guide

1. Installation guidance

1.1. System requirements

Table C.2.1. Requirements for Debrief

Requirement	Minimum	Recommended
Processor	800 MHz	1400 MHz
Memory	256 Mb	1024 Mb
Java Virtual Machine	JVM 1.5	As new as possible. JVM 1.5 includes significant performance improvements, particularly in the list processing used extensively within Debrief
XML Libraries	SAX and DOM support (through the enclosed jaxp.jar and parser.jar libraries)	
OpenMap libraries (used for plotting NIMA's Vector Map data)	4.2.1 (through the enclosed openmap.jar library)	

1.2. Implementing 'open with' support for DebriefNG

Since Winter 2010, DebriefNG can be opened by on a Debrief data file (rep (Section 1, “Replay file format”) or DPF (Section 2, “Debrief file format”)). But, first DebriefNG must be associated with files of that file-type, via Windows Explorer. Do this as follows[1]:

1. Open Windows Explorer
2. Navigate to a Debrief REP file
3. Right-click on the file and select 'Open With'
4. Now browse to the DebriefNG.exe executable, typically contained in the `c:\DebriefNG`. You may also wish to select 'Always use this application'.
5. Now repeat this process for a Debrief `dpf` file , you can select 'Always use this application' - since dpf files aren' used by many applications. Debrief did use the `xml` suffix, but that related to lots of files - so in Oct 2014 we adopted `.dpf`.

1.3. Directory structure

Debrief installs itself into a `Debrief NG` folder in the top level of your C:\ drive.. Within the Debrief NG folder you will find the DebriefNG executable in the top-level, together with some additional `dll` files used for optimised graphics (`gdiplus.dll`), and copy to clipboard (`JavaClipboard.dll`).

The `configuration` folder contains details of Debrief's initial settings together with the locations of update download sites.

[1]Note, across the range of versions of MS Windows a variety of user interface phrases exist. The steps in this process contain the text that will loosely match that which you'll see

The `features` and `plugins` folders include details of what Eclipse plugins are present (where each feature is actually a group of plugins), together with the plugins themselves. The `workspace` folder is a default workspace provided for new users. It contains current work projects together with a large volume of metadata representing change history, screen layout and user preferences.

1.4. Multi-user Debrief installation

The conventional way of installing Debrief is for each user to have their own copy, maintaining their own sets of plugins and updates (as above). There is an alternate installation scenario, however - for all users to share a common Debrief installation. An advantage of this scenario is that software updates only need to be performed once, at the central location. By default, however, shared installations are troublesome for Debrief - since all users will be looking at a single set of settings - so administrator participation is necessary.

Whilst a number of theoretical solutions exist for this problem, the most reliable solution to emerge is to force Debrief NG to use the workspace in the users own login directory (since each user has a unique login directory). This also overcomes the challenge of new users having to select a workspace before they know what a workspace is.

Thus, the Debrief shortcut (.ini file) has been modified to explicitly indicate that the workspace is in the a folder in the user login area:

```
-data
@user.home/.DebriefNG
```

The '.' character before the DebriefNG folder name follows a convention that is normally sufficient to make the operating system make the folder hidden by default.

Tip

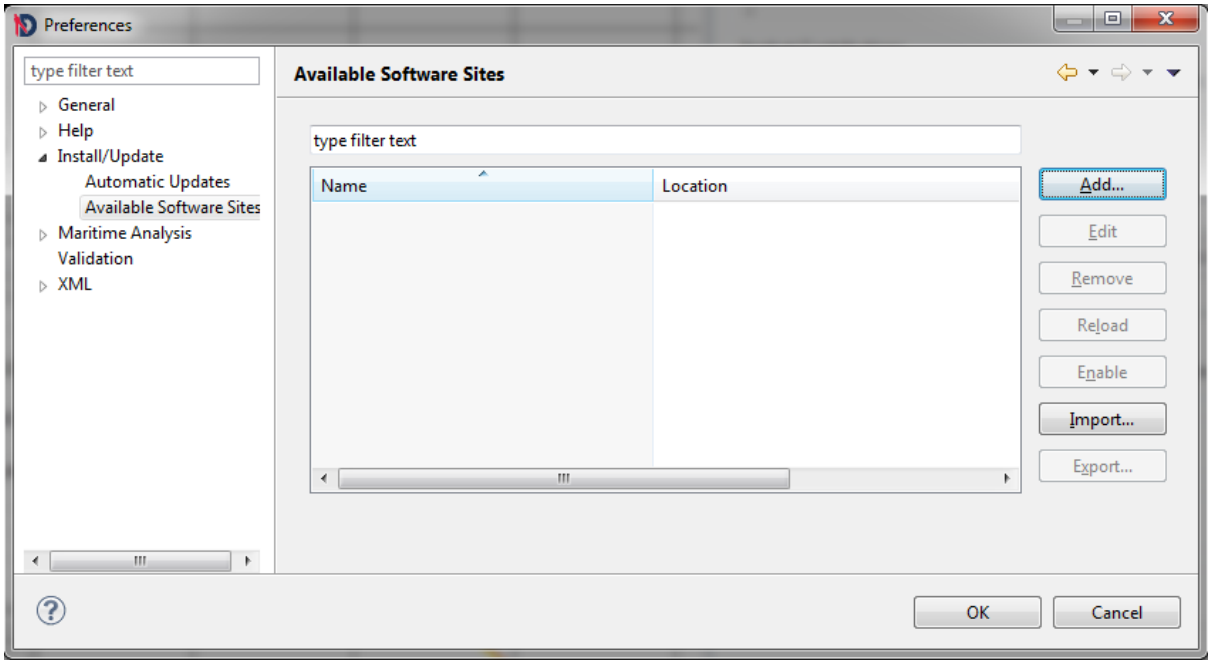
When users are upgrading from older versions of DebriefNG, the application will ignore the last workspace location, and load a new, blank workspace from the new user directory. If the user now closes Debrief the workspace data can be copied from the old location into the new location, which will get read when the application re-opens.

1.5. Centralised software updates

In the Autumn 2009 updates Debrief adopted the Eclipse p2 *update* mechanism. With this update came support for centralised updates. Centralised updates allow updated Debrief components to be placed at a central location on shared network storage, with individual Debrief installations automatically updating from it.

Start by locating a central shared folder, and creating a *Repository* folder in it. This is where your Debrief maintainer will unzip Debrief updates. Next, you must tell the individual Debrief installations where the repository is. Do this from the Preferences Install/Update Available Software Sites dialog:

Figure C.2.1. Repository location dialog



Then, to update a Debrief installation select *Check for updates* from the *Help* menu. Note, the *Install/Updates* section of Debrief preferences also includes options for automatic software updates.

1.6. Starting the program

The shortcut placed on the Start|Program Files|Debrief NG menu actually calls the `DebriefNG.exe` file found in the startup directory. This application triggers the startup of Debrief, passing to it the parameters contained in the `DebriefNG.ini` file. This file contains options to select the initial perspective, the memory allocation, and additional files to include on the classpath.

1.7. Use of English locale

Note that Debrief fails when on an machine set to a non-English locale, such as French(Canada). The application cannot read data-files correctly, nor can it populate screen editors. The workaround to this problem is to explicitly direct Debrief to startup using an English locale. Do this by launching Debrief with the "-nl en" command parameter. This can be performed either by modifying the shortcut used to start Debrief, or by adding the command parameter to the `Debrief.ini` configuration file (though this will have to be re-performed after each Debrief update).

2. Fault-diagnosis instructions

2.1. How-to

With the adoption of the Eclipse framework, Debrief enjoys the benefits of the Eclipse configuration and logging engine. Debrief uses this for information, warning, and error reporting.

Details of an individual Debrief NG installation can be obtained by selecting About Debrief NG from the Help menu. From the dialog that opens, select Installation Details. This will open a text dialog providing the following information:

- Date stamp
- System properties (about the Java and OS environments)
- Plug-in registry (plug-ins and fragments - ids, versions, and names)

- Update manager log
- Current error log contents

In addition to these configuration details is a button to View Error Log. The Error Log is a continuous stream of comments recorded by Debrief - typically comments that are not of value to a typical user under normal usage. The log itself is called `.log` and is located in the `workspace/metadata` folder within your Workspace (initially, this is within your Debrief NG installation directory).

Th error log may be examined for hints to the problem occurring. The error trace may be viewed in a text editor, and may be forwarded back to the Project Manager (see Table 1, “List of acknowledged Debrief users”) via post/e-mail/fax as applicable. The file does not contain any details of the data being edited, but the file may still be inspected before transmission

If these error messages do not provide any insight to the problem, you are left with the normal diagnosis steps as follows:

- Try to run another Java application on the machine, to check that Java is not corrupted
- If Java is ok then,
  - Check that Debrief can open and process the "sample" files included in the installation: boat1.rep and boat2.rep.
  - If Debrief can process these files, but not the current ones then
    - Try to re-install the Debrief application from 'save' disks to check that the sources are not corrupted
    - If at this point the error is still occurring then you will have to resort to reading as much of this Help guidance as practical, and then contacting the Project Manager (see Table 1, “List of acknowledged Debrief users”).

3. Storing Chart Folios

As well as supporting the drag/drop (see Section 2, “Loading data”) of chart data directly into a Debrief plot, Debrief supports the concept of a Chart Folio. Such a chart folio must adhere to specific layout conditions in order for Debrief to understand it.

Folder	The chart folio is contained in a single folder, with the folder named according to the geographic area covered (since this is the name offered to the user)
Shapefile	The folder must contain a shapefile named `rasterExtents_ARCS_Export.shp` . This shapefile must contain a series of rectangle geometries - one for each chart in the folio. Each rectangle must have a property named `the_geom` that contains the coordinates, and one called `Name` that contains the filename of the respective chart.
Chart images	The folder must also contain a series of Tif images with supporting .prj sidecar files. (Note: the folios received from UKHO in Autumn 2011 do not contain sidecar fiels. Debrief generates the missing sidecar files as necessary. This is acceptable for UKHO charts, which follow a particular standard - one cannot assume it non UKHO charts follow the same standard).

4. Debrief properties

4.1. Introduction

Whilst Debrief used to use a dedicated settings file, the settings have now been incorporated into the general framework of the application - with context sensitive help available where applicable.

5. Master template for export scenario to PowerPoint

5.1. Introduction

In Summer 2018 the ability to export an engagement/scenario to MS PowerPoint was introduced. In the past analysis had captured engagement to video, then included this video in a presentation. Hey, in the far distant past export video was a capability build into Debrief - though it as dropped since Debrief wasn't able to compress the videos, and they were huge.

The new strategy involves capturing the screen coordinates of scenario participants while Debrief is playing through an engagement, then injecting them as an animation into a specially formatted donor PowerPoint file.

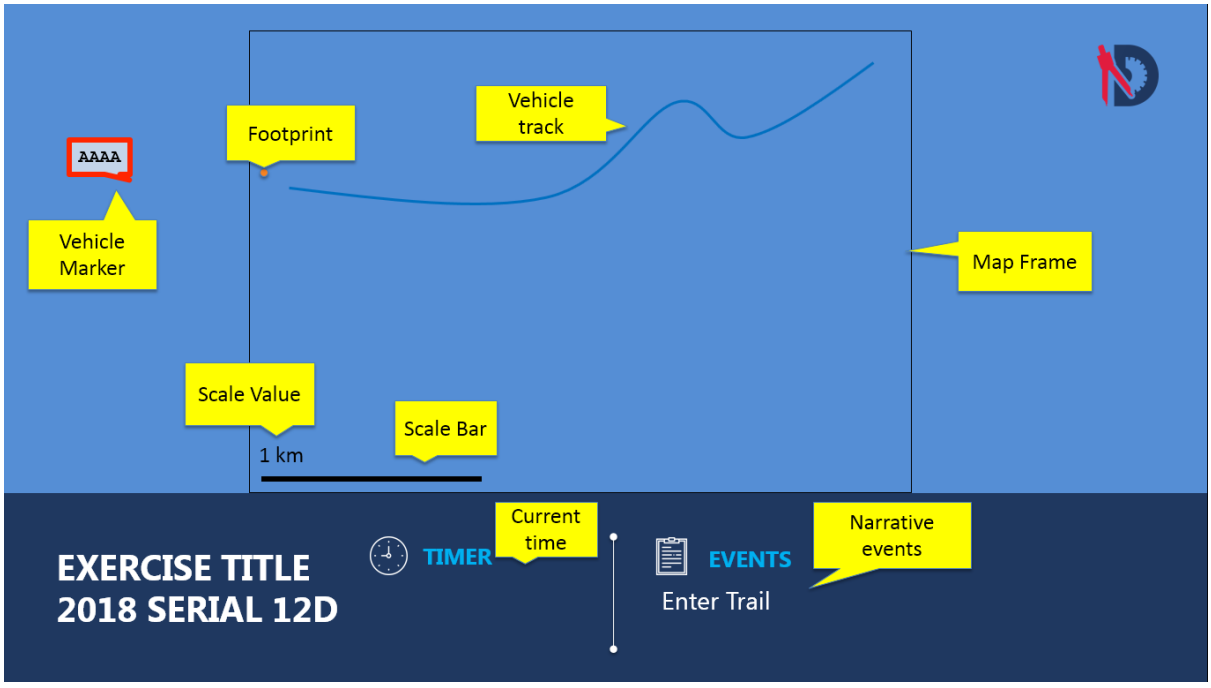
This section will introduce the donor template file, and how to collate/maintain it.

The donor file includes the following elements (named as specified), used as in the image below it:

Designated elements in template file

Map Frame	Frame into which the chart is plotted. (Rectangle shape named `map`)
Vehicle Marker	Marker to indicate current platform location (with platform name). (Callout named `marker`)
Vehicle track	Marker that is used to build up historic platform track. (Line named `track`.)
Narrative Events	List of visible narrative/diary events, populated as time progresses. (Line named `narrative`.)
Current Time	Current scenario time. (Text element named `time`.)
Footprint	Footprint to indicate the steps of the past plataform's positions. (Object named `footprint`)
Scale Value	Text value that indicates that distance. (Object named `ScaleValue`)
Scale Bar	This will consist of a rectangle, with width sized according to a particular distance. (Object named `ScaleBar`)

Figure C.2.2. Elements of master template

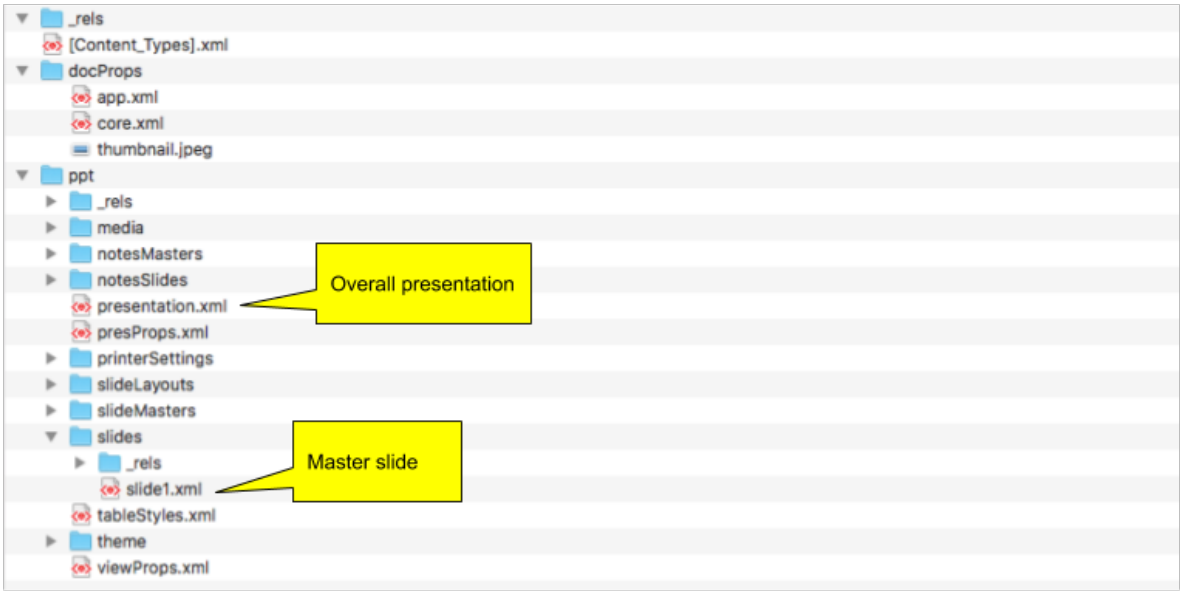


The use of this template allows an analyst's parent organisation to design a template that includes corporate branding, in order that the animated engagement tidily integrates into an outer presentation.

5.2. Folder structure

A `pptx` file is actually a renamed `zip` archive. The archive contains a series of folders/files, of which we are interested in two:

Figure C.2.3. PowerPoint folder structure



Overall presentation

This document contains metadata relating to the whole presentation, including the slide dimensions. The slide dimensions are signficant since some element positioning is performed relative to the size of the slide

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<p:presentation xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main" showSpecialPlsOnTitleSld="0"
saveSubsetFonts="1" autoCompressPictures="0">
    <p:sldMasterIdLst>
        <p:sldMasterId id="2147483648" r:id="rId1"/>
    </p:sldMasterIdLst>
    <p:notesMasterIdLst>
        <p:notesMasterId r:id="rId3"/>
    </p:notesMasterIdLst>
    <p:sldIdLst>
        <p:sldId id="257" r:id="rId2"/>
    </p:sldIdLst>
    <p:sldSz cx="12192000" cy="6858000"/>
    <p:notesSz cx="6858000" cy="9144000"/>
```

Master slide

This is the document that represents the specific slide that we're writing into, and it contains the named elements detailed above in Designated elements in template file

5.3. Master slide

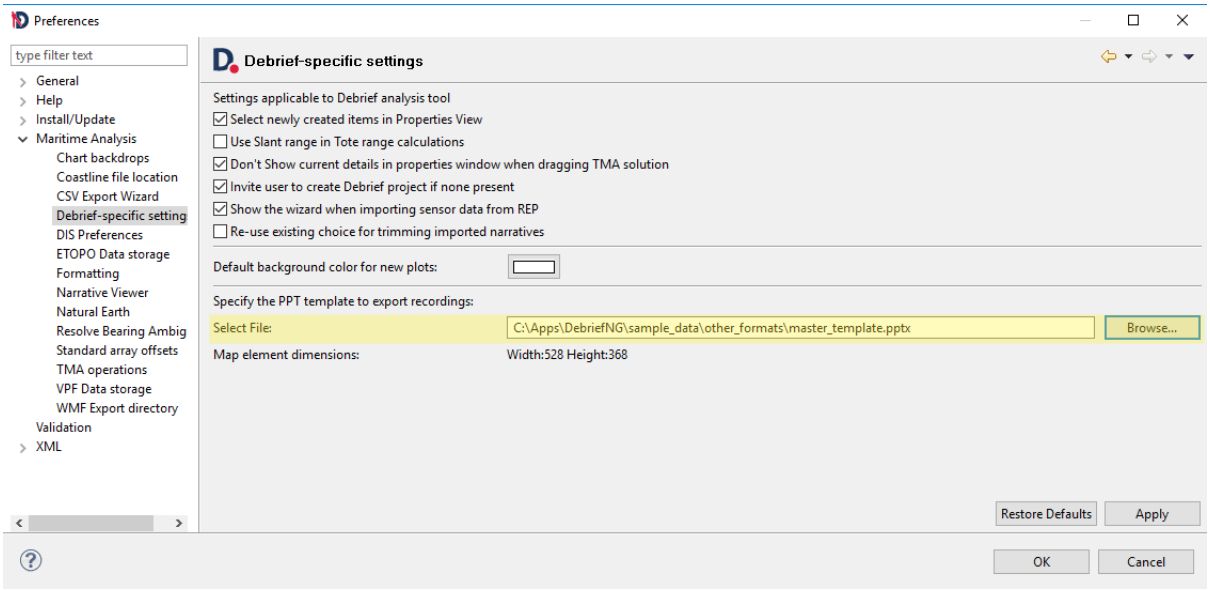
Significant elements in the master slide are denoted using their `name` attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<p:sld xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main" xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
    <p:cSld>
        <p:bg>
            <p:bgPr>
                <a:solidFill>
                    <a:schemeClr val="tx2">
                        <a:lumMod val="60000"/>
                        <a:lumOff val="40000"/>
                    </a:schemeClr>
                </a:solidFill>
                <a:effectLst/>
            </p:bgPr>
        </p:bg>
        <p:spTree>
            <p:nvGrpSpPr>
                <p:cNvPr id="1" name=""/>
                <p:cNvGrpSpPr/>
                <p:nvPr/>
            </p:nvGrpSpPr>
            <p:grpSpPr>
                <a:xfrm>
                    <a:off x="0" y="0"/>
                    <a:ext cx="0" cy="0"/>
                    <a:chOff x="0" y="0"/>
                    <a:chExt cx="0" cy="0"/>
                </a:xfrm>
            </p:grpSpPr>
            <p:sp>
                <p:nvSpPr>
                    <p:cNvPr id="2" name="map"/>
                    <p:cNvSpPr/>
                    <p:nvPr/>
                </p:nvSpPr>
```

The donor file is built using the following process:

1. Create a one-slide presentation that contains the elements described in the earlier table
2. Rename the `.pptx` file to `.zip`
3. Expand the zip-file into a new folder (probably via right-click)
4. Navigat to the slide1.xml file, located in the folder as detailed above in Section 5.2, “Folder structure”
5. Work through each required element, naming the relevant xml element using the ids listed earlier. Finding the relevant elements in the XML file can be made easier by inserting recognisable blocks (*Trumpton*) of text into the element using PowerPoint, then looking for that text in the XML editor.
6. Re-package the folder into a zip-file
7. Rename the `.zip` file back to `.pptx`

One the master slide has been produced, use Debrief preferences to indicate its location:



Note the *Map element dimensions* attribute in the above screenshot. Once a master template has been specified, the map element dimensions are retrieved. This is to allow the analyst to size the Debrief plot to the same dimensions, which will ensure the highest quality animated export - since the Debrief Plot and the target rectangular frame will be of identical proportions.

Part D. Reference Guide

If the tutorial hasn't provided you with all of the answers you need, just have a look at the pages here in the Reference Guide.

Chapter D.1. Debrief file formats

1. Replay file format

1.1. Track data

Debrief accepts files in the Replay datafile format. This format uses an character string containing data/time, vessel name, position, heading, depth and speed. More than one vessel track can be stored in each file. The gaps between the data fields can be any whitespace characters, that is any number of spaces or tabs, but there must be a value for each data field.

Note

Whilst file formats recorded here are shown against an 80 character index marker - the fields can be of any width - the data is broken down using delimiters, not fixed field widths.

The file-naming convention for files in the *Replay* format is to have a `.REP` suffix.

Through an extension to the format, annotation data can also be accepted. The annotation data format is described below. Whilst annotation and track data may be stored in single file, it is recommended that they are stored in separate files, to ease reading and plotting track-data only.

The standard data format is as follows:

```
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
YYMMDD HHMMSS.SSS XXXXXX SY DD MM SS.SS H DDD MM SS.SS H CCC.C SS.S DDD
 xx...xx
```

or since Autumn 2004 multi-word track names can be entered by enclosing them in double-quotation marks (") and 4-figure year values are supported:

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
YYYYMMDD HHMMSS.SSS "XYX XYX XYX" SY DD MM SS.SS H DDD MM SS.SS H CCC.C
 SS.S DDD xx...xxx
```

The field descriptions are:

Table D.1.1. Fields in Debrief position entry

Field number	Description
1	Date, either 2 of 4 figure date, followed by month then date
2	Time
3	Vessel Name - either as single, unquoted word, or as a multi-word phrase enclosed in quotation marks.
4	Symbology (2 or 5 chars, see below)
5	Latitude Degrees (Debrief is able to handle decimal degrees - provide zeros for mins and secs)
6	Latitude Minutes (Debrief is able to handle decimal mins - provide zeros for secs)

Field number	Description
7	Latitude Seconds
8	Latitude Hemisphere
9	Longitude Degrees (Debrief is able to handle decimal degrees - provide zeros for mins and secs)
10	Longitude Minutes (Debrief is able to handle decimal mins - provide zeros for secs)
11	Longitude Seconds
12	Longitude Hemisphere
13	Heading (0..359.9 degrees)
14	Speed (knots)
15	Depth (metres) [a]
16	Text label (optional). Any text appearing after the depth value is used as a label for that position. If no label is supplied a time-related label is generated. *Note: since Jan 2018 it has been possible to also provide a comment. See Section 1.6, "Entry comments"*

[a]Where depth data is not available, the IEE symbol NaN (Not a Number) may be used. Debrief reflects the absence of this data where applicable.

1.2. Symbology attributes

The symbology data field describes the representation required for this vessel track, specifying the colour of track to use and symbol-type to represent that vehicle. The Symbol style read in is used when tracks are to be shown by using symbol (in 2D or 3D). Control of when symbols are shown, and examples of the symbols are shown in Chapter B.5, *Symbol sets*.

Table D.1.2. Debrief symbology color codes

Colour			
A	Blue	J	Light Green
B	Green	K	Pink
C	Red	L	Gold
D	Yellow	M	Light Grey
E	Magenta	N	Grey
F	Orange	O	Dark Grey
G	Purple	P	White
H	Cyan	Q	Black
I	Brown		

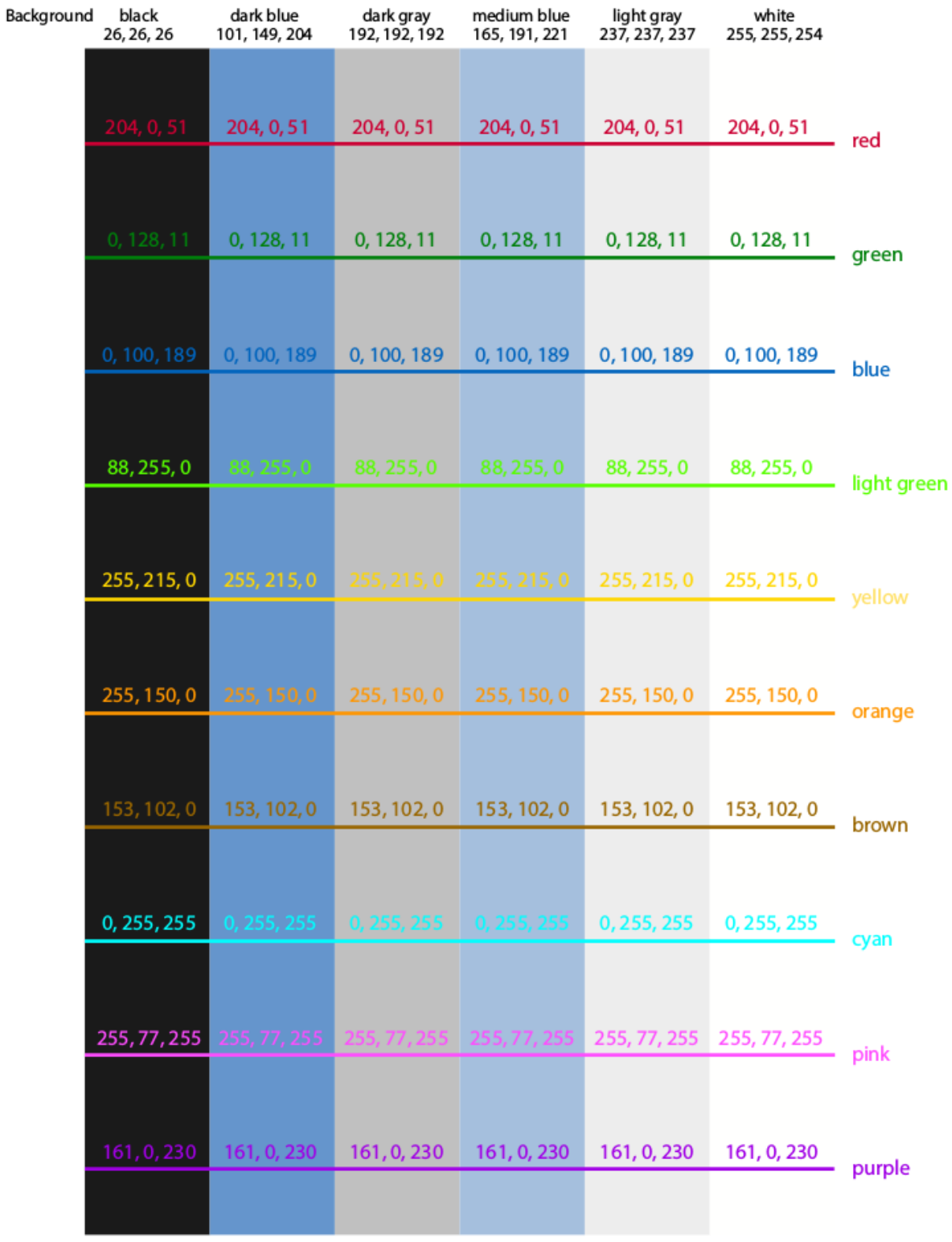
Table D.1.3. Debrief symbology symbol codes

Internal Symbols		SVG Symbols	
A	Aircraft	a	Merchant
C	Carrier	b	Fishing
D	Destroyer	c	Pleasure_craft
F	Frigate	d	Coastguard law enforcement
H	Helicopter	e	Friend Surface

Internal Symbols		**SVG Symbols**	
M	Merchant	f	Friend Subsurface
N	Minesweeper	g	Neutral Air
P	Torpedo	h	Neutral Surface
R	Troop Carrier	i	Neutral Subsurface
S	Submarine	j	Enemy Air
T	TA Frigate	k	Enemy Surface
U	Cruiser	l	Enemy Subsurface
V	Fishing Vessel	m	Unknown Air
@	Unknown	n	Unknown Surface
I	Cross	o	Unknown Subsurface
J	Square	p	Buoy_1
K	Datum	q	Buoy_2
L	Reference Position	r	Missile
Q	Circle	s	Torpedo
W	Wreck	t	Generic arrow
E	Filled Square	u	Drop point
G	Filled Circle	v	Splash point
0	Active Buoy	w	Vector_1
1	DIFAR Buoy	x	Vector_2
2	LOFAR Buoy		
3	BARRA Buoy		
4	Hidar Buoy		
5	Kingpin Marker		

Here's an illustration of how the new colors appear against the range of Debrief background colors:

Figure D.1.1. 2015 Debrief shades



Note

In 2015 the Debrief symbology attributes were extended to allow richer formatting to be applied to data on import. The new attributes mean it is possible to use a 2 or 5 character symbology element for each REP line. This was of particular value in supporting quick turnaround, in-stride, fast-look analysis. It is also of value when high

volumes of annotations are produced using a simulation environment such as Odin [https://www.atlas-elektronik.com/what-we-do/submarine-systems/odin/].

Table D.1.4. Debrief extended symbology data fields

Line Type		Line Thickness		Fill Style	
@	SOLID	0	Hairwidth	0	No-fill
A	DOTTED	1	1 pixel	1	Solid fill
B	DOT_DASH	2	2 pixels	2	Semi-transparent fill
C	SHORT_DASHES	3	3 pixels		
D	LONG_DASHES	4	4 pixels		
E	UNCONNECTED	5	5 pixels		

Note

In 2017 the Debrief symbology attributes were extended to allow the provision of some optional named parameters. The first of these is the specification of the layer into which the annotation should be inserted. Note: in the initial implementation spaces are not allowed in the entries. The second of these is the specification of the SVG Symbol to be used in the track.

Table D.1.5. Debrief extended symbology data fields

Example	Usage
[LAYER=Other_Target_Layer]	Put this annotation into the layer named "Other_Target_Layer". Note: spaces not allowed. Use this mechanism to load a track as a high performing *Lightweight Track*.
[SYMBOL=SVG_ICON_NAME]	This annotation places the specified icon as a symbol. Note: spaces not allowed. Icon name must be the exact name that appears in the SVG Section.

Please, take a look at the following example of the extended symbology at use

```
951212 050700.000 NELSON @C[SYMBOL=missile] 22 11 10.08 N 21 43 20.34 W
 270.5   2.0      0
951212 050701.000 COLLINGWOOD @C[SYMBOL=torpedo,LAYER=Support] 22 11 10.08
 N 21 43 20.34 W 270.5   2.0      0
```

1.3. Annotation Data Intro

In addition to the positional vessel track data, annotations can also be added. Each annotation is placed on a single line in a replay file, each line beginning with the comment marker; a semi-colon ';'.To put a comment on a line, begin with two semi-colons.

Note

Note, when null positions are entered for *Sensor* data, the position is taken from the track named in "ownship name". Debrief finds the nearest track location equal to or greater than the indicated time, and uses this as the sensor lat and long.

Note

As with normal positional data, annotation entries which use a track/ownship name may contain multi-word phrases if they are enclosed in quotation marks ("), and the year can be expressed as 2 or 4 figures.

Note

Debrief can handle decimal values for degrees or minutes. If you original data is in decimal degrees or degrees plus decimal minutes feel free to use these values directly, but remember to provide a zero for the unused column(s). Examples of this are shown below.

```
;; example using D, M, S
951212 050100.000 NELSON @C 22 11 10.58 N 21 42 2.98 W 269.7 2.0 0
;; example using D
951212 050100.000 NELSON @C 22.455 0 0 N 21.432 0 0 W 269.7 2.0 0
;; example using D, M
951212 050100.000 NELSON @C 22 11.6678 0 N 21 42.543 0 W 269.7 2.0
 0
```

Note

Where labels are provided for shapes, Debrief allows multi-line text labels - just insert a \n character to indicate the line-break

Annotation positions are specified in degrees as in the standard replay file format, and symbology representations are as in the above tables.

1.4. Core Elements

The format for the different types of annotations is:

```
123456789012345678901234567890123456789012345678901234567890123456789
012345678901234567890123456789

;LINE: @@ DD MM SS H DDD MM SS.S H DD MM SS.S H DDD MM SS H XX.XXX
;; symb, start lat & long, end lat & long, text label (optional)

;VECTOR: @@ DD MM SS H DDD MM SS.S H RRR BBB XX.XXX
;; symb, start lat & long, range (yds), brg (degs) , text label
 (optional)

;TEXT: @@ DD MM SS H DDD MM SS H XX..XX
;; symb, lat & long, text (note: trailing text is optional, so just
 the symbol can be specified)

;NARRATIVE:  YYMMDD HHMMSS  TTT.TTT  XX..XX
;; dtg, track name, narrative entry

;TIMETEXT: @@ YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H XX..XX
;; symb, date, lat & long, text
```

```
;PERIODTEXT: @@ YYMMDD HHMMSS YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S
 H
DDD XX..XX
;; symb, start date, end date, lat & long, Depth(optional) text

;GRID: @@ DD MM SS.S H DDD MM SS.S H MM.MM MM.MM XX..XX
;; symb, centre lat & long (N13) lat increment, long increment, text

;RECT: @@ DD MM SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H XX..XX
;; symb, tl corner lat & long, br corner lat & long, label

;POLY: @@ YYMMDD HHMMSS YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H DD MM
 SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H  XX...XX
;POLY: @@ DD MM SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H DD MM
 SS.S H DDD MM SS.S H  XX...XX
;; symb, start date (optional), end date (optional), lat & long 1, lat
 & long 2, lat & long xxx (until end of line, or non-numeric label
 encountered)

;POLYLINE: @@ YYMMDD HHMMSS YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H
 DD MM SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H  XX...XX
;POLYLINE: @@ DD MM SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H DD
 MM SS.S H DDD MM SS.S H  XX...XX
;; (this is an open polygon) symb, start date (optional), end date
 (optional), lat & long 1, lat & long 2, lat & long xxx (until end of
 line, or non-numeric label encountered)

;SGSAGEOG: @@ YYMMDD HHMMSS YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H
DD MM SS.S H DDD MM SS.S H XX..XX
;; symb, start date, end date, tl lat & long, br lat & long,
text label (1 word)

;SGSAGRID: @@ YYMMDD HHMMSS YYMMDD HHMMSS AA NN AA NN XX..XX
;; symb, start date, end date, tl corner, br corner, text label (1
 word)

;WHEEL: @@ YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H YYY YYY XX.XX
;; symb, centre date, centre lat, centre long, inner radius (yards),
outer radius (yards), label

;CIRCLE: @@ DD MM SS.S H DDD MM SS.S H YYY XX.XX
;; symb, centre lat, centre long, radius (yards), label

;ELLIPSE: BD YYMMDD HHMMSS DD MM SS.SS H DD MM SS.SS H CCC XXXX YYYY
 xx.xx
;; symb, date, time, lat, long, orientation, maxima (yards),
minima (yards), label

;ELLIPSE2: BD YYMMDD HHMMSS YYMMDD HHMMSS DD MM SS.SS H DD MM SS.SS H
 CCC XXXX YYYY xx.xx
;; symb, start-date, start-time, end-date, end-time, lat, long,
 orientation, maxima (yards),
minima (yards), label

;BRG: BD YYMMDD HHMMSS DD MM SS.SS H DD MM SS.SS H CCC XXXX xx.xx
;; symb, date, time, lat, long, orientation, length (yards), label
 (one word)
```

```
;SENSOR: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
BBB.B RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
bearing (degs), range(yds) [or the single world NULL], sensor name,
 label (to end of line),
, optional comment - see Section 1.6, “Entry comments”

;SENSOR2: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
 BBB.B CCC.C
FFF.F RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
bearing (degs) [or the single word NULL], ambigous bearing (degs) [or
 the
single word NULL], frequency(Hz) [or the single word NULL],
 range(yds)
[or the single word NULL], sensor name, label (to end of line),
 optional
comment - see Section 1.6, “Entry comments”

;SENSOR3: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
 BBB.B  CCC.C
FFF.F GGG.G RRRR yy..yy xx..xx
;; date, ownship name, symbology, sensor lat/long (or the single word
 NULL),
bearing (degs) [or the single word NULL], bearing accuracy (degs)
[or the single word NULL], frequency(Hz) [or the single word NULL],
frequency accuracy (Hz) [or the single word NULL], range(yds)
[or the single word NULL], sensor name, label (to end of line),
 optional
comment - see Section 1.6, “Entry comments”

;TMA_POS: YYMMDD HHMMSS.SSS AAAAAA @@ DD MM SS.SS H DDD MM SS.SS H
TT...TT OOO.O  XXXX YYYY  CCC SSS DDD xx.xx
;; date, time, ownship name, symbology, tma lat, tma long, track
 name,
ellipse orientation (deg from north),  maxima (yds), minima (yds),
 course,
speed, depth (m), label string

;TMA_RB: YYMMDD HHMMSS.SSS AAAAAA @@ BBB.B RRR.R TT...TT OOO.O
XXXX YYYY  CCC SSS DDD xx.xx
;; date, time, ownship name, symbology, bearing (deg), range (yds),
track name, elipse orientation (deg from north),  maxima (yds), minima
 (yds),
course, speed, depth (m), label string

;NARRATIVE:  YYMMDD HHMMSS  TTT.TTT  XX..XX
 ;; dtg, track name, narrative entry. Note the track name can be
 multi-word if
surrounded by quotes (").

;NARRATIVE2:  YYMMDD HHMMSS  TTT.TTT AA..AA XX..XX
 ;; dtg, track name, narrative entry-type, narrative entry.  Note the
 track name
and entry type can be multi-word if surrounded by quotes (").
```

1.5. Planning Legs

Note that the final leg in a set of planning legs can take an optional *CLOSING* field. This will make it join back to the start point.

```
;; PLANNING TRACKS

;PLANNING_ORIGIN: YYMMDD HHMMSS AAAAAA @@ 22 12 10.28 N 21 32 40.33 W
 0
;; origin point for a set of planning legs
;; date-time, platform, symbology, origin, depth (metres)

;PLANNING_RANGE_SPEED: AAAAAA BBBBBB RRRR SSS CCC
;; leg based upon range/speed
;; platform, leg-name, range (yds), speed (kts), course (degs)

;PLANNING_RANGE_TIME: AAAAAA BBBBBB RRRR TTT CCC
;; leg based upon range/duration
;; platform, leg-name, range (yds), duration (secs), course (degs)

;PLANNING_SPEED_TIME:  AAAAAA BBBBBB SSS TTT CCC
;; leg based upon speed/time
;; platform, leg-name,  speed (kts), duration (secs), course (degs)

;PLANNING_RANGE_SPEED:  AAAAAA BBBBBB RRRR SSS CCC CLOSING
;; If supplied as the last in a series of legs, this becomes a closing
 leg.
```

1.6. Entry comments

Note: some entry types allow an additional comment to be appended to the end of an entry. The comment follows a // separator.

```
951212 050100.000 NONSUCH @A@00 22 12 10.51 N 21 32 14.81 W 269.9 2.0
 0
;; label is normal DTG value, comment is empty (legacy behaviour)

951212 050200.000 NONSUCH @A@00 22 12 10.51 N 21 32 14.81 W 269.9 2.0
 0 Standard label
;; label is "Standard label", comment is empty (legacy behaviour)

951212 050300.000 NONSUCH @A@00 22 12 10.51 N 21 32 27.27 W 268.7 2.0
 0 Standard label // Custom comment
;; label is "Standard label", comment is "Custom comment"

951212 050400.000 NONSUCH @BA10 22 12 10.28 N 21 32 40.33 W 270.6 2.0
 0 // Custom comment
;; label is normal DTG value, comment is "Custom comment"

;SENSOR2: 951212 051600.000 NEL_STYLE2 @B NULL 59.3 300.8 49.96 NULL
 SENSOR LABEL // COMMENT
;; label is "SENSOR LABEL", comment is "COMMENT"
```

For track positions, it's possible to indicate whether the comment should be shown, using the Comment Showing attribute of the position. The comment is always placed opposite the label, even if the label isn't shown

1.7. Towed Array Extensions

In Spring 2017 additional annotations were introduced to support more complex Towed Array datasets

```
;TA_FORE_AFT: 090722 041522 NONSUCH SENSOR_A_1 44.2 260.2 45.4 262.3
;; data from module sensors, in depth/heading pairs
;; date-time, platform, sensor, two pairs of depth (m) & heading
 (degs) data.  Or replace the pairs with four NULL values.

;TA_MODULES: YYMMDD HHMMSS AAAAAA BBBBBB DD.D HH.H ... DD.D HH.H
;; 12 sets of depth/heading data from module sensors. Either first 4,
 or last 8 will be populated
;; the other pairs should have zero values
;; date-time, platform, sensor, pairs of depth & heading data, to the
 end of line.
;; But, if you're not interested in the spec at all, you could just
 include four NULL values.

;TA_COG_ABS: YYMMDD HHMMSS AAAAAA BBBBBB LAT.Y LONG.X DD.D
;; absolute measurement of towed array aperture Centre of Gravity
;; date-time, platform, sensor, lat/long (Degrees) depth (m). Lat/Lon/
D may be null.

;TA_COG_REL: YYMMDD HHMMSS AAAAAA BBBBBB XX.X YY.Y DD.D
;; relative measurement of towed array aperture Centre of Gravity
;; date-time, platform, sensor, x/y/depth(m). X/Y/D may be null
```

1.8. Dynamic Elements

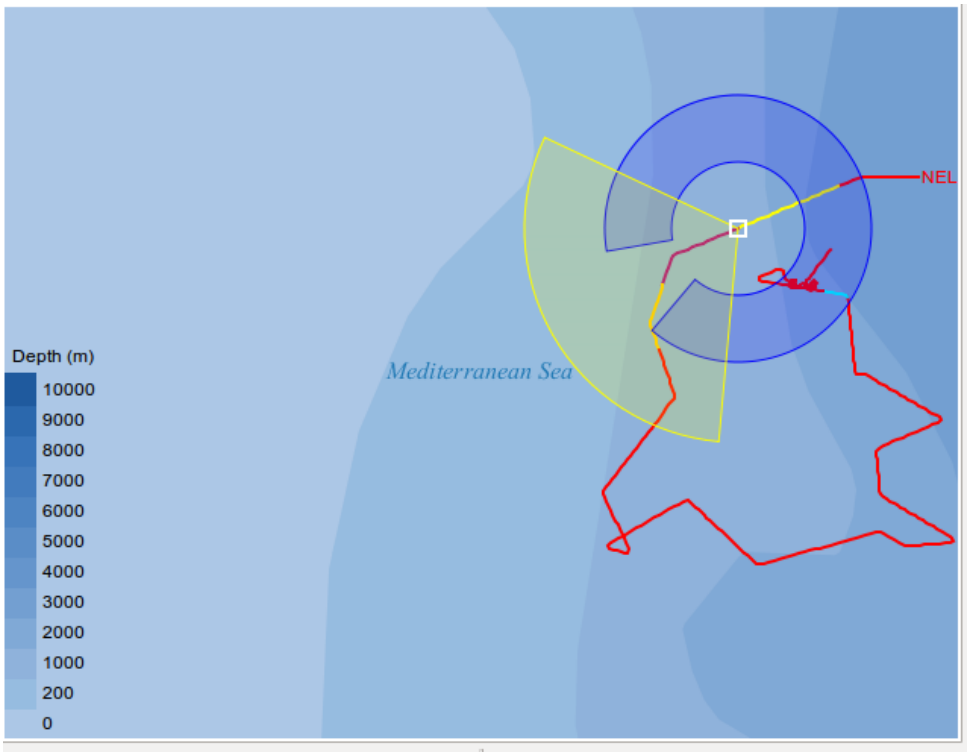
In addition to individual annotation shapes it is possible to define a dynamic annotation. Do this by using a series of dynamic annotations, all with the same annotation name. Once loaded, Debrief will only show the shape nearest to the current scenario time (according to the Time Controller). The dynamic annotations largely take the same format as their non-dynamic equivalents as described above, save for the introduction of a Shape Name

The format for the different types of dynamic annotations is:

```
;;DYNAMIC_CIRCLE: @@@@@ "NAME" YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S
 H YYY XX.XX
;;DYNAMIC_POLY: @@@@@ "NAME" YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H
 DD MM SS.S H DDD MM SS.S H DD MM SS.S H DDD MM SS.S H  XX...XX
;;DYNAMIC_RECT: @@@@@ "NAME" YYMMDD HHMMSS DD MM SS.S H DDD MM SS.S H
 DD MM SS.S H DDD MM SS.S H XX..XX
;; symb, shape name, date-time [other fields as in the standard
 annotation type]
;; example:
;DYNAMIC_CIRCLE: @@011 "Circle Track A" 151212 143000 54 05 12.1 N 4 0
 12.4 W 1000 At time 1430
```

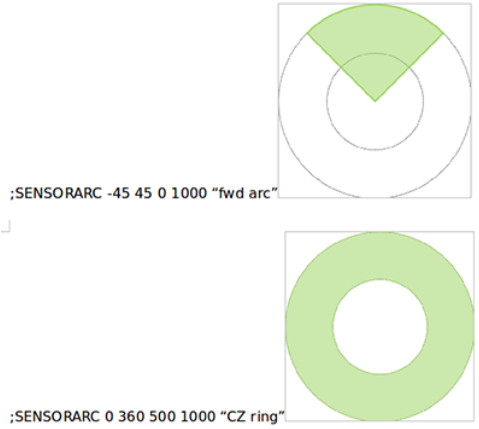
```
;DYNAMIC_CIRCLE: @@011 "Circle Track B" 151212 143100 54 05 14.1 N 4 0
 12.1 W 1100 At time 1431
;DYNAMIC_POLY: @@011 "Circle Track C" 151212 143200 54 05 15.1 N 4 0
 12.1 W  54 06 15.1 N 4 0 11.1 W  54 05 13.1 N 4 1 14.1 W At time 1432

;DYNAMIC_RECT: @@011 "Circle Track D" 151212 143300 54 05 17.1 N 4 0
 12.1 W  54 05 10.1 N 4 0 12.6 W At time 1433
```

1.9. Dynamic Track Annotations

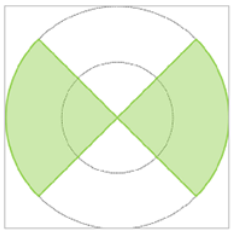


It is also possible to attach annotations to tracks - so that the annotation moves (and rotates) with the track. These annotations can be grouped. This allows the dynamic shape itself to change dimensions during its life, but with the whole collection of shapes switched on and off as one.

Here are a series of examples (but note that the REP lines have been abbreviated):



;SENSORARC -135 -45 0 1000 45 135 0 1000 “beams”



The format for the different types of dynamic track annotations is shown below. Note that each line may contain a series of definitions, and that the distance units are in yards.

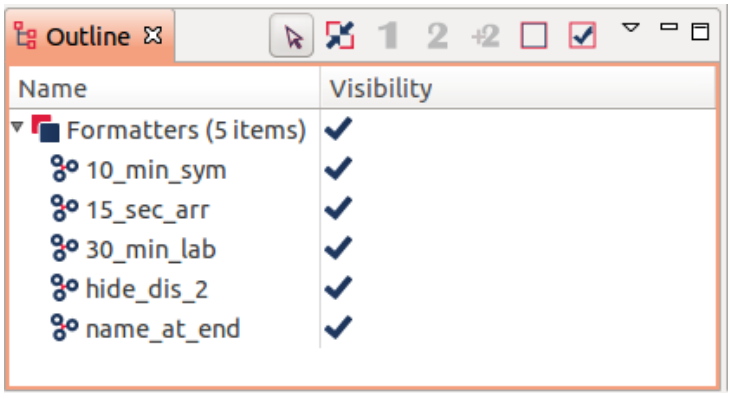
```
;SENSORARC YYMMDD HHMMSS.SSS YYMMDD HHMMSS.SSS TRACKNAME SYMBOLOGY
 LEFT RIGHT INNER OUTER [LEFT RIGHT INNER OUTER] LABEL
;; (start time, end time, parent track, styling, min-max angles of the
 arc, inner-outer radiuses, collective label.
;; simple example

;SENSORARC 951212 062800 951212 063000 NELSON @@ -75 35 0 1000 "fwd
 dynamic"
;; dynamic example - so donut spreads outwards in time using 3 steps

;SENSORARC: 951212 053000 951212 054000 NELSON @A 0 360 500 1000
 CZ_Ring
;SENSORARC: 951212 054000 951212 055000 NELSON @B 0 360 600 1100
 CZ_Ring
;SENSORARC: 951212 055000 951212 060000 NELSON @C 0 360 700 1200
 CZ_Ring
```

Note

A NULL can be provided instead of one or more of the times. This will result in the dynamic shape being used for part or all of the track lifetime.

1.10. Formatting helpers

Entries can be included in REP files to help format data as it is loaded. These commands can specify the frequency at which labels/symbols are shown, or perform other presentation configuration, either on specific tracks/layers or for all data.



Tip

You can use these helpers to create a set of default look & feels for your plots. Create a set of label, symbol, arrow markers to suit your data, then store them in a REP file, called something like `LONG_RANGE_FORMAT.REP`. Then, when conducting a new piece of analysis of long range tracks, start by opening this REP file. Now, when you drag in your tracks, they will all get formatted according to your preferred styling.

The format for these helpers is shown below.

```
;FORMAT_FIX: 30_min_lab LABEL NULL NULL TRUE 1800000
;; 30_min_lab - name for this formatter (shown in outline view)
;; LABEL - what we're applying to (SYMBOL, LABEL, ARROW)
;; NULL - track name we apply to (or null for all tracks)
;; NULL - symbology we apply to (e.g. @A) (or null for all tracks)
;; TRUE - whether interval should fall on regular interval or start
;;        from first data point
;; 1800000 - interval (in millis) to apply formatting

;FORMAT_FIX: 10_min_sym SYMBOL NELSON NULL TRUE 600000
;; show symbol on every 10 minute marker (00:00,00:10, 00:20, etc)
;;    for track Nelson

;FORMAT_LAYER_HIDE: format_name layer_name
;; name for this formatter, name of the layer to hide

;FORMAT_LAYER_HIDE: format_name layer_1_name "layer 2 name"
;; name for this formatter, name of the layers to hide
;;     quotes can be used if name contains spaces

;FORMAT_TRACK_NAME_AT_END: format_name track_names
;; name for this formatter, list of track names for which the name
;;     should be shown at the end, or empty to apply to all tracks
```

Note

A NULL can be provided instead of one or more of the times. This will result in the dynamic shape being used for part or all of the track lifetime.

2. Debrief file format

2.1. Introduction

DPF is the Extensible Markup Language. Like its predecessor SGML, *XML* is a meta-language used to define other languages. The article in the Reference Guide taken from MSDN gives some background to *XML*

2.2. Adoption of XML

An acknowledged shortcoming of the former file format (`dpl`) used in Debrief was the frequency with which Debrief modifications rendered previous `DPL` files unreadable. This use of Java serialization offered low cost in development terms and was seen as essential to store the more complex data used in Debrief 2000 (such as projections, and formatting data).

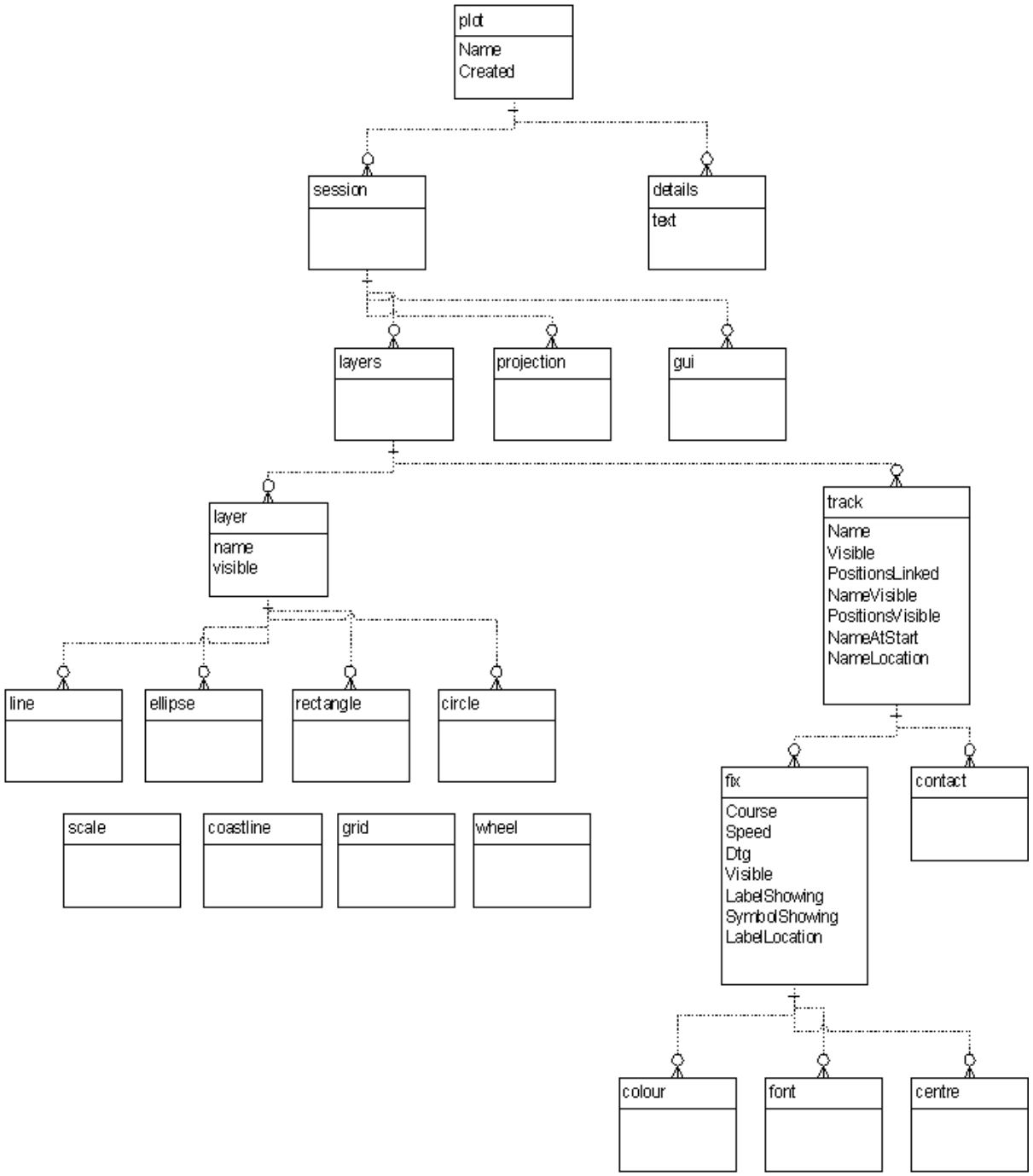
The only way to divorce the Debrief version from the file format has been to adopt an independent file format. This independent format must be structured in order to store Debrief data tidily, and the development to support it must not be too "expensive". The adoption of XML meets both of these criteria = it is quite capable of storing Debrief data, and there are a number of libraries available which support it.

XML provides additional benefits beyond those described above:

- It is human-readable, offering the following benefits:
  - Apparently corrupt data files can be examined in a text editor and repaired
  - Data files can be edited outside Debrief
- It is a world-wide standard independent of Debrief and Java
  - Applications are freely available which support it: Internet Explorer will open XML data files for inspection
  - When used in conjunction with the Debrief DTD, the IBM Xeena editor provides easy editing/ creation of Debrief plots
  - As more organisations/industries adopt XML, there is increased likelihood that import/export filters will not be required, since XML documents are easily translated between each other.

The XML structure adopted within Debrief is as follows:

Figure D.1.2. XML file structure in Debrief



Note

In Autumn 2014 the `dpf` suffix was adopted. The data is still stored in XML format, but the adoption of a unique suffix makes it easier to find Debrief files from a File Explorer.

3. KML file format

3.1. Introduction to KML file format

Keyhole Markup Language (KML) is an XML-based language schema for expressing geographic annotation and visualization on existing or future Internet-based, two-dimensional maps and three-dimensional Earth browsers. KML was developed for

> use with Google Earth, which was originally named Keyhole Earth Viewer. It was created by Keyhole, Inc, which was acquired by Google in 2004. The name "Keyhole" is an homage to the KH reconnaissance satellites, the original eye-in-the-sky military reconnaissance system first launched in 1976. KML is an international standard of the Open Geospatial Consortium. Google Earth was the first program able to view and graphically edit KML files, and other projects such as Marble have also started to develop KML support
>
> —WikiPedia [http://en.wikipedia.org/wiki/Kml], 2010

Whilst it has military roots, the KML dialect of XML has received wide adoption for the recording and distribution of positional information. KML extends the GML standard. KML and GML provide a number of ways of recording positional data. This section will record the KML dialects that Debrief understands.

3.2. LineString KML files

Handheld GPS systems (including mobile phones) tend to store positional data in a **Placemark** that contains a single, lengthy **LineString** element containing a series of long, lat, altitude elements - devoid of time stamps. Debrief makes the following assumptions regarding this format:

1. If altitude data is missing, its comma separator will still be present
2. The filename will be of the form w20090802164801.kml, where the digits represent year, month, day, hour, minute, second
3. The data items will have been recorded at one second intervals
4. The filename (without suffix) is used as the track identifier

3.3. MultiGeometry KML files

More capable recording systems tend to store positional data in a more refined, time-stamped format - using series of **Placemark** entries - each containing a **MultiGeometry** record. Additionally, these more complex files are able to represent data for a number of different vehicles. Debrief makes the following assumptions regarding this file format:

1. Only **Placemark** elements that contain a **MultiGeometry** element are treated as compliant and processed
2. The **Placemark** name element is used to determine the track id: any digits/letters appearing before a "-" hyphen are treated as a track id, this multiple positions are attributed to their relevant track
3. Tracks imported into Debrief from this file format are named as the first component of the filename (before the "."), followed by the unique id declared in the Placemark entries.
4. The contents of the **when** element are treated as the time-stamp of the position record (time-zone ignored, always treated as GMT)
5. The long/lat/altitude coordinates are taken from the first **coordinates** element in the **Placemark**
6. Course and speed data is taken from the **description** element, based on a presumption that the element contains a string formatted like: `<br>Course: 345.2<br>Speed: 9.6 knots<br>Date:`

4. Flat file format

4.1. Introduction to Flat file (SAM) format

One Debrief user organisation frequently has to use exported Debrief tracks in a subsequent analysis application (SAM). This application requires tab-separated columns of data using a flat-earth

measurements in imperial units. The data is formatted as specified below. Note that records are only included for times for which there is sensor datums - which may be a shorter time period than that which the user specified.

4.2. Schema for Flat file format

```
STRAND Scenario Report 1.00
<Operation_Name>
<Platform_Name>
<Ownship_Track_Name>
<Sensor_Track_Name>
<Measured_Tgt_Name>
<Predicted_Tgt_Name>
<Start_Time>
<End_Time>
<Bearing_SIF>
<Range_SIF>
<Combined_SIF>
<Num_Records>
<Geo_Origin_X> <Geo_Origin_Y>
Time OS_Status OS_X OS_Y OS_Speed OS_Heading Sensor_Status Sensor_X
 Sensor_Y
Sensor_Brg Sensor_Bacc Sensor_Freq Sensor_Facc Sensor_Speed Sensor_Heading
Sensor_Type Msd_Status Msd_X Msd_Y Msd_Speed Msd_Heading Prd_Status Prd_X
 Prd_Y
Prd_Brg Prd_Brg_Acc Prd_Range Prd_Range_Acc Prd_Course Prd_Cacc Prd_Speed
 Prd_Sacc
Prd_Freq Prd_Freq_Acc
<scenario_sample_time> <OS_Data_Status> <OS_X> <OS_Y> <OS_Speed>
 <OS_Heading>
  <Sensor_Data_Status> <Sensor_X> <Sensor_Y> <Sensor_Brg> <Sensor_Bacc>
 <Sensor_Freq>
  <Sensor_Facc> <Sensor_Speed> <Sensor_Heading> <Sensor_Type> <Msd_status>
 <Msd_X>
  <Msd_Y> <Msd_Speed> <Msd_Heading> <Prd_Status> <Prd_X> <Prd_Y> <Prd_Brg>
  <Prd_Brg_Acc> <Prd_Range> <Prd_Range_Acc> <Prd_Course> <Prd_Cacc>
 <Prd_Speed>
  <Prd_Sacc> <Prd_Freq> <Prd_Freq_Acc>
<scenario_sample_time> <OS_Data_Status> <OS_X> <OS_Y> <OS_Speed>
 <OS_Heading>
  <Sensor_Data_Status> <Sensor_X> <Sensor_Y> <Sensor_Brg> <Sensor_Bacc>
 <Sensor_Freq>
  <Sensor_Facc> <Sensor_Speed> <Sensor_Heading> <Sensor_Type> <Msd_status>
 <Msd_X>
  <Msd_Y> <Msd_Speed> <Msd_Heading> <Prd_Status> <Prd_X> <Prd_Y> <Prd_Brg>
  <Prd_Brg_Acc> <Prd_Range> <Prd_Range_Acc> <Prd_Course> <Prd_Cacc>
 <Prd_Speed>
  <Prd_Sacc> <Prd_Freq> <Prd_Freq_Acc>
<scenario_sample_time> <OS_Data_Status> <OS_X> <OS_Y> <OS_Speed>
 <OS_Heading>
  <Sensor_Data_Status> <Sensor_X> <Sensor_Y> <Sensor_Brg> <Sensor_Bacc>
 <Sensor_Freq>
  <Sensor_Facc> <Sensor_Speed> <Sensor_Heading> <Sensor_Type> <Msd_status>
 <Msd_X>
  <Msd_Y> <Msd_Speed> <Msd_Heading> <Prd_Status> <Prd_X> <Prd_Y> <Prd_Brg>
  <Prd_Brg_Acc> <Prd_Range> <Prd_Range_Acc> <Prd_Course> <Prd_Cacc>
 <Prd_Speed>
  <Prd_Sacc> <Prd_Freq> <Prd_Freq_Acc>
```

```
<scenario_sample_time> <OS_Data_Status> <OS_X> <OS_Y> <OS_Speed>
 <OS_Heading>
  <Sensor_Data_Status> <Sensor_X> <Sensor_Y> <Sensor_Brg> <Sensor_Bacc>
 <Sensor_Freq>
  <Sensor_Facc> <Sensor_Speed> <Sensor_Heading> <Sensor_Type> <Msd_status>
 <Msd_X>
  <Msd_Y> <Msd_Speed> <Msd_Heading> <Prd_Status> <Prd_X> <Prd_Y> <Prd_Brg>
  <Prd_Brg_Acc> <Prd_Range> <Prd_Range_Acc> <Prd_Course> <Prd_Cacc>
 <Prd_Speed>
  <Prd_Sacc> <Prd_Freq> <Prd_Freq_Acc>
....
  Own Ship Data Fields    Sensor Data Fields    Measured Target Data Fields

  Predicted Target Data Fields

Data Dictionary
All fields are tab delimited.  All text shown in bold is to be included in
 the output file exactly as shown.

Header Information
<Operation_Name>     The name of the operation taken from the header
 Information of the own ship file
<Platform_Name>      The name of the own ship platform taken from the
 headerinformation of the own ship track.
<Ownship_Track_Name> The name of the own ship track taken from the header
 information of the own ship track
<Sensor_Track_Name>  The name of the sensor track taken from the header
 informationof the sensor track
<Measured_Tgt_Name>  The name of the measured target track taken form the
 target trackheader information
<Predicted_Tgt_Name> The name of the predicted target track taken from the
 targettrack header information
<Start_Time>         The time of the first data record in the form:
                      hh:mm:ss <TAB> dd/mm/yyyy
<End_Time>           The time of the last data record in the form:
                      hh:mm:ss <TAB> dd/mm/yyyy
<Bearing_SIF>        The system integrity factor based on bearing
 information
<Range_SIF>          The system integrity factor based on range information
<Combined_SIF>       The system integrity factor for the whole system
<Num_Records>        The number of data records in the file's body
<Geo_Origin_X>       The X component of the geographical origin in
 cartesian coordinates
<Geo_Origin_Y>       The Y component of the geographical origin in
 cartesiancoordinates

Body Information
All records consist of a sequence of TAB delimited fields generating a
 tabular output.  Each column will be identified by a title as shown in
 bold type.

<scenario_sample_time> The time of the scenario information contained in
 the record.
                        Measured in seconds from the start time given in
 the header.
<OS_Data_Status>      Flag defining the availability of own ship data built
 from the sum of the following codes.
```

```
                        0 = No own ship data, 1 = OS Position, 2 = Os
 Speed,
                        4 = OS Heading.  For example if all own ship data
 is available the status = 1+2+4 = 7
<OS_X>                The X component of the own ship position in
 cartesiancoordinates at scenario sample time.
                        Use STRANDPos type - integers measuring scenario
 location in yards
<OS_Y>                The Y component of the own ship position in
 cartesiancoordinates at scenario sample time.
                        Use STRANDPos type - integers measuring scenario
 location in yards
<OS_Speed>            The speed of the own ship at scenario_sample_time in
 knots.
                        Positive float
<OS_Heading>          The heading of the own ship at scenario_sample_time
 in degrees.
                        Float in range 0 to 359.99
<Sensor_Data_Status>  Flag defining the availability of sensor data built
 from the sum of the following codes.
                        0 = No sensor data, 1 = Sensor position, 2 = sensor
 bearing,
                        4 = sensor frequency, 8 = Sensor Speed, 16 = Sensor
 Heading,
                        32 = Sensor Type.  For example if all sensor data is
 available
                        the status = 63
<Sensor_X>            The X component of the sensor position in cartesian
 coordinates at scenario sample time.
                        Use STRANDPos type - integers measured in yards
<Sensor_Y>            The Y component of the sensor position in cartesian
 coordinates at scenario sample time.
                        Use STRANDPos type - integers measured in yards
<Sensor_Brg>          The true bearing measured by the sensor in degrees.
                        Float in range 0 to 359.99
<Sensor_Bacc>         The accuracy of the measured bearing in degrees.
 Float in
                        range 0 to 180.0
<Sensor_Freq>         The frequency measured by the sensor in hertz.
 Positive Float
<Sensor_Facc>         The accuracy of the measured frequency in hertz.
 Positive float
<Sensor_Speed>        The speed of the sensor in knots. Positive float
<Sensor_Heading>      The heading of the sensor in degrees. Float in range
 0 to 359.99
<Sensor_Type>         The type of sensor the data refers to.
                        1 character H = hull or T = towed array
<Msd_status>          Flag defining the availability of measured target
 data built from the sum of the following codes.
                        0 = no measured target data, 1 = measured target
 position,
                        2 = measured target speed, 4 = measured target
 heading.
<Msd_X>               The X component of the measured target position in
 cartesian coordinates at scenario sample time.
                        Use STRANDPos type - integers measured in yards
<Msd_Y>               The Y component of the measured target position in
 cartesian coordinates at scenario sample time.
```

```
                        Use STRANDPos type - integers measured in yards
<Msd_Speed>            The measured target speed in knots. Positive float
<Msd_Heading>          The measured target heading in degrees. Float in
 range 0 to
                        359.99
<Prd_status>           Flag defining the availability of predicted target
 data built from the sum of the following codes.
                        0 = no predicted target data, 1 = predicted
 position,
                        2 = predicted  bearing, 4 = predicted bearing
 accuracy,
                        8 = predicted range, 16 = predicted range accuracy,
                        32 = predicted course, 64 = predicted course
 accuracy,
                        128 = predicted speed, 256 = predicted speed
 accuracy,
                        512 = predicted frequency, 1024 = predicted
 frequency accuracy
<Prd_X>                The X component of the predicted target position in
 cartesian coordinates at scenario sample time.
                        Use STRANDPos type - integers measured in yards
<Prd_Y>                The Y component of the predicted target position in
 cartesian coordinates at scenario sample time.
                        Use STRANDPos type - integers measured in yards
<Prd_Brg>              The predicted target bearing in degrees.  Float in
 range 0 to
                        359.99
<Prd_Brg_Acc>          The predicted target bearing accuracy in degrees.
 Float in
                        range 0 to 180.0
<Prd_Range>            The predicted target range in yards.  Positive float
<Prd_Range_Acc>        The predicted target range accuracy in yards.
 Positive float
<Prd_Course>           The predicted target course in degrees.  Float in
 range 0 to
                        359.99
<Prd_Cacc>             The accuracy of the predicted course in degrees.
 Positive
                        float in range 0 to 359.99
<Prd_Speed>            The predicted target speed in knots.  Positive float
<Prd_Sacc>             The accuracy of the measured speed in knots.
 Positive float
<Prd_Freq>             The predicted target range in hertz  Positive float
<Prd_Freq_Acc>         The predicted target frequency accuracy in hertz
 Positive float
```

5. Multipath analysis datafiles

5.1. Introduction

The Multi Path Analysis view (see Section 2.7, “Multipath analysis”) allows a target depth to be estimated by visually aligning a calculated set of time delays with a measured set. This view requires two additional data files to be loaded. Both of these files are in textual `csv` format.

5.2. SVP file

The Sound Velocity Profile file is a two-column csv text file:

depth floating point depth (m)

sound speed floating point sound speed (m/s)

5.3. Time delays file

The time-delays data file is a multi-column csv text file:

```
YYYY,MM,DD,HH,MM,SS,mmm,TIME_DELAY(msec),POWER(dB)
2009,04,22,18,37,00,254,6.797688,22
```

date Series of fields recording date and time (including milliseconds) of observation

time-delay floating point time delay (mSecs)

power ignored in current implementation

Note, you can determine the timestamp using the unix timestamp [http://www.unixstamp.com/] web site.

6. S2087 Track files

6.1. Introduction

The S2087 system is able to provide ownship track. On occasions when the command system isn't able to provide ownship track then this data can be used as a fallback.

The following assumptions have been made regarding the data:

- The file ends in .csv
- Course is in degrees, and speed is in knots
- Measured Time is recorded in GMT
- Measurement time is stored in the SYS_ORDINAL_TIME column

Note

Local IT settings may hide the file suffix for MS Windows devices. So, a file titled `context.csv` may just appear as `context`, with MS Windows indicating that the file is a *Microsoft Office Excel Comma Separated Value file*.

6.2. File format

Here is a sample of the expected format (line-breaks inserted to aid legibility).

```
SYS_INTERNAL_TIME,SYS_ORDINAL_TIME,SHIP_ORDINAL_TIME,LATITUDE,LONGITUDE,SPEED_MADE_GOOD
SHIP_COURSE,
 LOG_SPEED,SHIP_HEADING,TA_SCOPE,TA_DEPTH,TA_X,TA_Y,TA_HEADING,TA
 Deployed,TB_SCOPE,
TB_DEPTH,TB_X, TB_Y, TB_HEADING, TB Deployed,SPEED_OF_SOUND, TEMPERATURE,
 SHIP_ROLL, SHIP_PITCH,
WIND_DIRECTION, WIND_SPEED, DRIFT_NORTH, DRIFT_EAST
23-08-2014 02 36 53,01-09-2014 00 00 12,31-08-2014 23 00
 08,48.55566666666667,-9.359333333333334,
8.922246514887131,318.8,9.349891895292325,319.2,303,48,-570.0,511.3,317.0,TA_DEPLOYED,7
66.3,319.0,TB_DEPLOYED,1513.51,16.0,0.0,0.0,0.0,0.0,0.0,0.0
23-08-2014 02 36 55,01-09-2014 00 00 13,31-08-2014 23 00
 08,48.57566666666667,-9.339333333333334,
```

```
8.722246514887131,319.8,9.449891895292325,320.2,303,48,-570.0,511.3,317.0,TA_DEPLOYED,70
66.3,319.0,TB_DEPLOYED,1513.51,16.0,0.0,0.0,0.0,0.0,0.0,0.0
23-08-2014 02 36 57,01-09-2014 00 00 15,31-08-2014 23 00
 08,48.59566666666667,-9.319333333333334,
8.522246514887131,321.8,9.549891895292325,321.2,303,48,-570.0,511.3,317.0,TA_DEPLOYED,70
66.3,319.0,TB_DEPLOYED,1513.51,16.0,0.0,0.0,0.0,0.0,0.0,0.0
23-08-2014 02 36 59,01-09-2014 00 00 19,31-08-2014 23 00
 08,48.61566666666667,-9.269333333333334,
8.322246514887131,323.8,9.649891895292325,322.2,303,48,-570.0,511.3,317.0,TA_DEPLOYED,70
66.3,319.0,TB_DEPLOYED,1513.51,16.0,0.0,0.0,0.0,0.0,0.0,0.0
```

7. SVG Symbol Format

7.1. Introduction

Traditionally, symbols were hard-coded in Java code, within Debrief. When a new symbol was required, the graphically talented analyst would develop a set of pairs of coordinates for the new shape in Excel, then pass these to the maintainer, for their translation into source code. Adopting SVG as a standard is seen as a way to get artist-drawn icons and symbols into Debrief. Learn more about the symbols provided in Chapter B.5, *Symbol sets*.

7.2. Implementation

SVG files are XML-based language schema showing how to render the images. Debrief has implemented a subset of the SVG format. It supports the following shapes: Circle, Ellipse, Line, Poligon, Polyline and Rectangle. Any other shape would be ignored in the rendering process. There are also some extra rules to guarantee that all the icons in Debrief have a consistant appearance.

- They must have a size of 100x100 pixels
- There must be an circle shape having the id `origin`. It indicates where is the center of the image, which is important to place it in the correct position on tracks, labels, etc. This origin is also the centre of rotation for the shape, for shapes that are able to rotate.
- If the user needs the shape to be able to rotate, you need to set the Circle color to Green. If you need to remain aligned as the original, just assign any other color
- Image filling is very similar to the regular SVG shape. If the style is set to `` `fill: none` `` or `` `fill-opacity: 0` `` it will be empty. It could also be assigned using the traditional XML attribute `fill-opacity` instead of style.
- In case you need to specify a color, you could do it using the attribute `` `fill` ``, assigning the color in hexagonal format.

7.3. Here are some examples:

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="100px" height="100px" viewBox="0 0 100 100" version="1.1"
 xmlns="http://www.w3.org/2000/svg">
  <polyline fill="#000000" fill-opacity="0" fill-rule="nonzero"
 stroke="#000000" stroke-width="5" points="5 53 20 24 35 53 50 24 65 53 80
 24 95 53"/>
  <polyline fill="#000000" fill-opacity="0" fill-rule="nonzero"
 stroke="#000000" stroke-width="5" points="5 76 20 47 35 76 50 47 65 76 80
 47 95 76"/>
  <circle id="origin" cx="50" cy="50" r="2.5" style="fill: rgb(255, 0,
 0);"/>
</svg>
```

In this case, we have an image with two polyline shape. Also pay attention to the circle shape which _must_ have the `id` attribute value as `origin`.

```
<?xml version="1.0" encoding="utf-8"?>
<svg width="100px" height="100px" viewBox="0 0 100 100" version="1.1"
 xmlns="http://www.w3.org/2000/svg">
  <polygon id="Triangle-2" points="49.5 0 67.5 100 32.5 100"/>
  <circle id="origin" cx="50" cy="69" r="2.5" style="fill: rgb(0, 255,
 0);"/>
</svg>
```

We have only one polygon in this case. Please notice that this shape is able to rotate according to the subject vehicle.

8. Third Party BRT Format

8.1. Introduction

Some analysts use a third party software package to produce their bearing data. Unfortunately, for towed array data the package calculates the absolute bearing using the current platform heading, not the heading for the platform was following when it was at the current array centre.

But, since the packagea already outputs a data-file containing the relative bearing we're able to do our own calculation of absolute bearing, using a better value for ownship heading.

8.2. Format

The data is in two columns, elapsed seconds since the epoch, and the relative bearing in Degreees, like this:

```
1263297600.000000, 69.00
1263297840.000000, 67.85
1263298080.000000, 65.63
1263298320.000000, 63.33
```

Chapter D.2. Scripting Cookbook

1. Introduction

A cookbook helps users learn a programming api/language by building up a repository of snippets that implement a specific task. The **Debrief Scripting Cookbook** contains code snippets for some of the sample tasks that can be performed using the Scripting Perspective feature of Debrief. You can use these code snippets as building blocks to solve more complex issues.

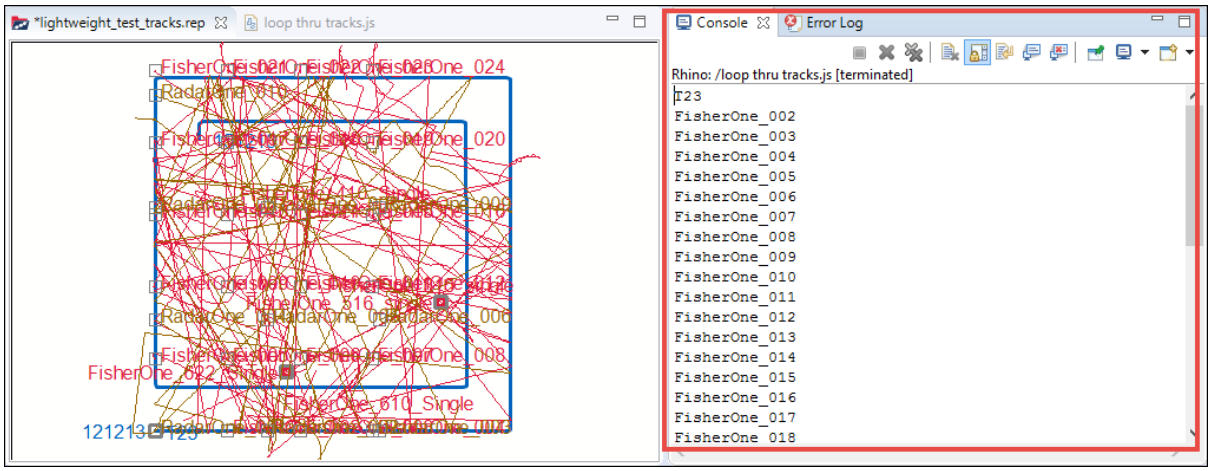
Refer to the Debrief Scripting Tutorial to gain a higher level understanding of developing scripts in Debrief.

1.1. Looping through Tracks

A lot of **Debrief** operations will need to loop through all loaded tracks, in a particular plot. The script below will facilitate that:

```
loadModule('/Debrief/Core');
loadModule('/Debrief/Annotations');
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot();
// @type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers();
let tracks = layers.getTracks();
for (let count =0 ; count < tracks.length; count++)
{
  //@type Debrief.Wrappers.TrackWrapper
  let track = tracks[count];
  print(track.getName())
}
```

The result of the above code snippet on the file **lightweight_test_tracks.rep** is shown in the below screenshot. The result is the list of names of the tracks in the plot **lightweight_test_tracks.rep** will display in the **Console**.



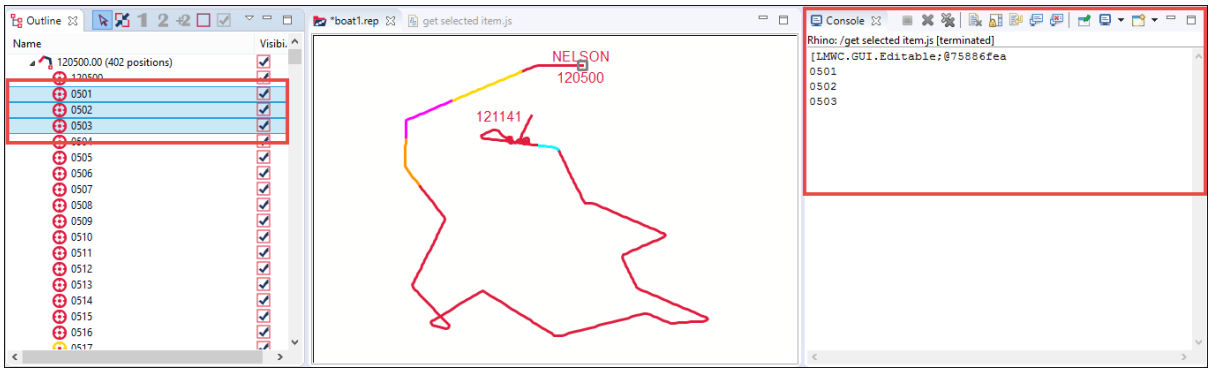
1.2. Getting Selected Item

Sometimes you may want to select the subject of a script by selecting an item in the Outline View, rather than hard-coding it. This script does that.

```
loadModule('/Debrief/Core');
```

```
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot();
//@type org.mwc.debrief.scripting.wrappers.Outline
let outline = plot.getOutline();
let selectedObjects = outline.getSelection();
print(selectedObjects);
let count;
for (count =0 ; count < selectedObjects.length; count++ )
{
  print(selectedObjects[count].getName());
}
if(selectedObjects.length == 0)
{
 print("Please select one or more items in the Outline View");
 exit(0);
}
```

The result of this script when it is run on the file **boat1.rep**is shown in the below screenshot. The selected items, in the outline view, in this example are positions **0501**, **0502**, and **0503**.

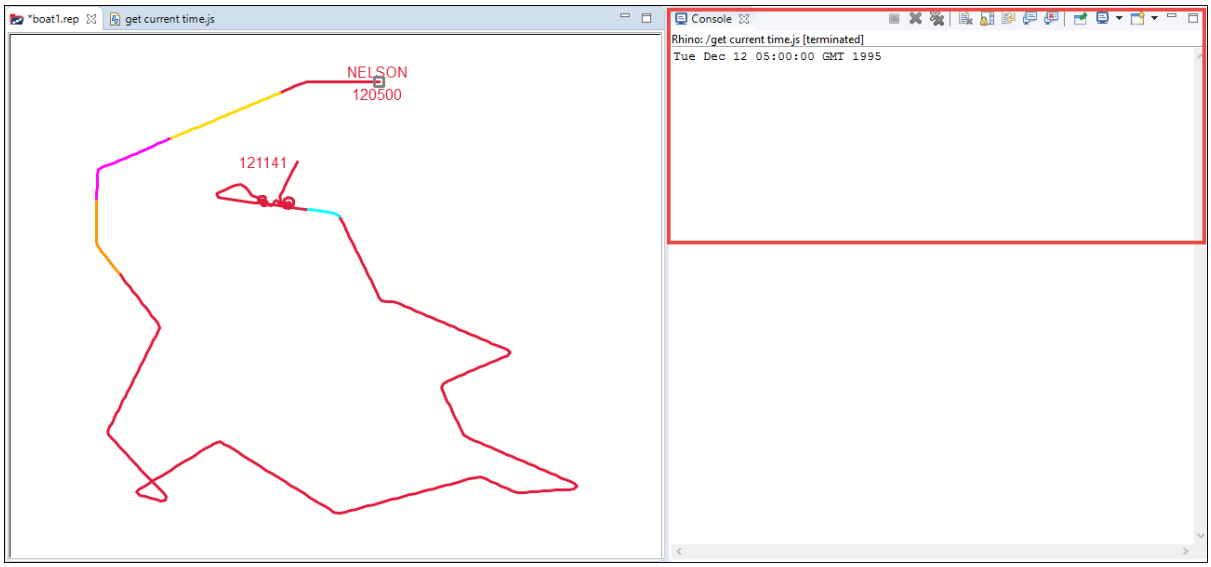


1.3. Getting Current Time of a Scenario

When you are analysing a track and moving forward through the scenario, you may want to retrieve the current time of the scenario to perform some calculation. The below script addresses this need:

```
loadModule('/Debrief/Core');
loadModule('/Debrief/Annotations');
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot();
// @type MWC.GenericData.HiResDate
let currentTime = plot.getTime();
print(currentTime.getDate());
```

The result, the current time of the scenario, of the above code snippet on the file **boat1.rep** is shown in the below screenshot. The current time of the scenario will display in the **Console**:

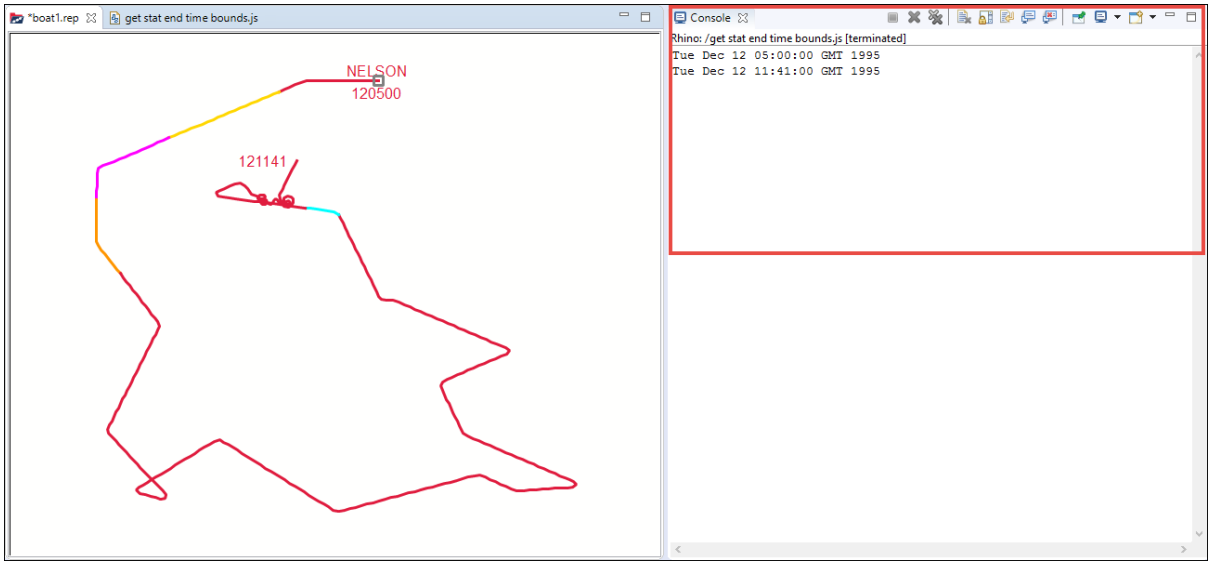


1.4. Getting Start and End Time Bounds of a Scenario

You may, at times, want to retrieve the start and end time bounds of a plot, possibly to set the axes on a graph. The below script gets the start and end time bounds of a scenario:

```
loadModule('/Debrief/Core');
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot();
//@type MWC.GenericData.TimePeriod
let period = plot.getTimePeriod();
print(period.getStartDTG().getDate());
print(period.getEndDTG().getDate());
```

The result of this script, i.e, the start and end time bounds of the scenario, on the file **boat1.rep** is shown in the below screenshot. The result will display in the**Console**:



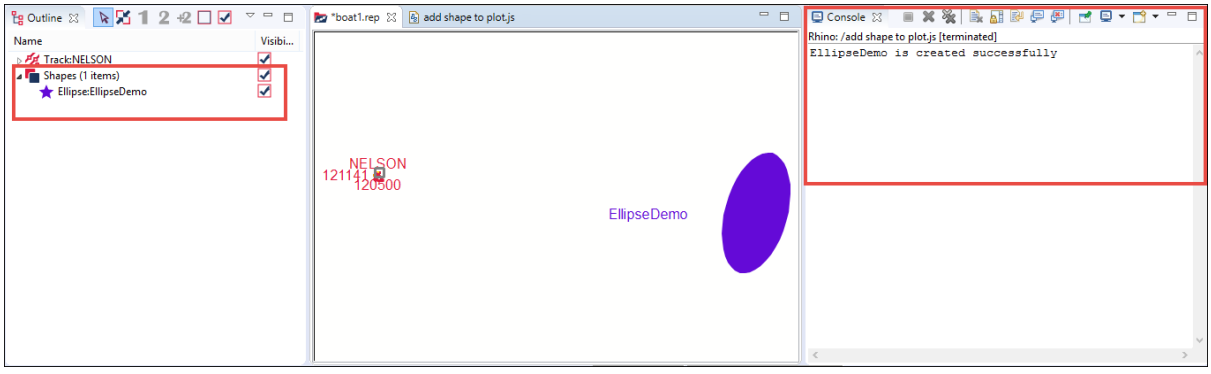
1.5. Adding Shape to a Plot

You can add value to a plot by adding shapes to it. You can use shapes, in a plot, to represent a wide range of real world entities. Also shapes can be organised in layers. The below script demonstrates creating a layer named **Shapes**, creating the shape ellipse named **EllipseDemo**, defining the position of the shape EllipseDemo, adding the newly created shape (**EllipseDemo**) to the newly created

layer (**Shapes**), setting the color of the shape, and displaying the output message "**EllipseDemo is created successfully**" in the **Console**.

```
loadModule('/Debrief/Core');
loadModule('/Debrief/Shapes');
loadModule('/Debrief/Spatial');
// get active plot
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot();
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers();
//create color
let purple =  createColorRGB(100, 10, 215) ;
// @type MWC.GUI.Layer
let shapeLayer = layers.createLayer("Shapes");
layers.add(shapeLayer );
// prepare data necessary to create ellipse
// @type MWC.GenericData.WorldLocation
let ellipseLoc = createLocation(21,-10, 0);
// @type MWC.GenericData.WorldDistance
let wdDistance1 = createDistance(200.5, KM) ;
// @type MWC.GenericData.WorldDistance
let wdDistance2 = createDistance(100.5, KM) ;
// @type Debrief.Wrappers.ShapeWrapper
let myecllipse = createEllipse(ellipseLoc, 15.3, wdDistance1,
    wdDistance2, "EllipseDemo", purple);
myecllipse.getShape().setFilled(true);
shapeLayer.add(myecllipse)
print(myecllipse.getName() + " is created successfully")
```

The result of the script on the file **boat1.rep** is shown in the screenshot below. The **Outline** view will display the newly added layer **Shapes** and the shape **EllipseDemo**.



1.6. Loading a Track from an Unusual Track Format

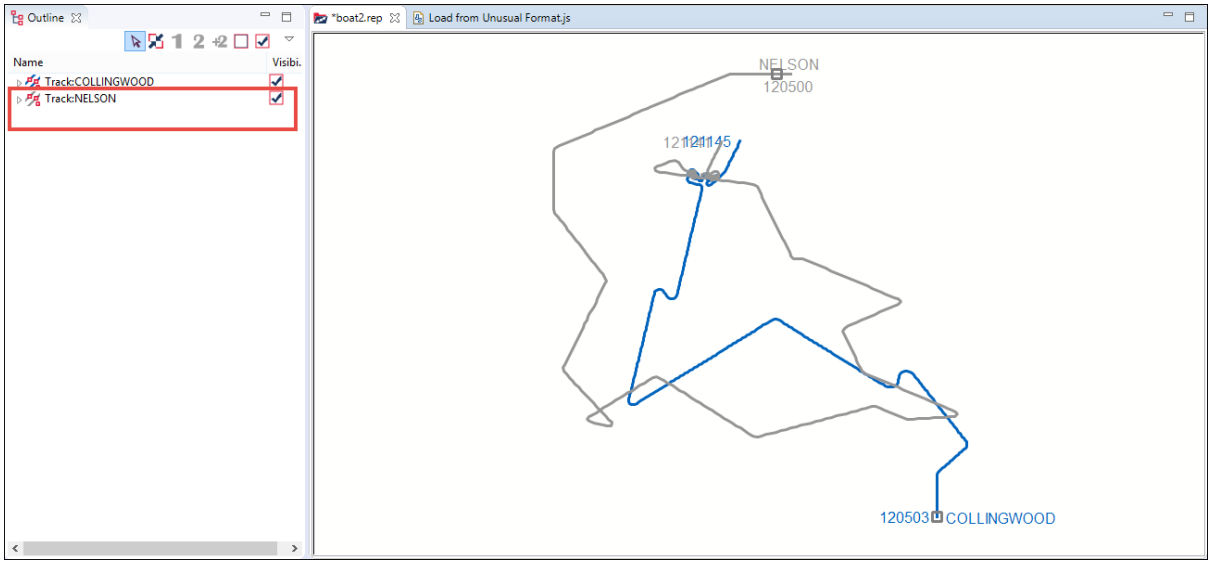
Debrief supports a variety of file types. With the added advantage of scripting in **Debrief**, you will be able to handle new or rare file types also.

```
loadModule('/System/Resources', false);
loadModule('/Debrief/Core'); // date
loadModule('/Debrief/Spatial'); // location
loadModule('/Debrief/Tactical'); // for plot, duration, date
// get project instance, in order to refer to a file
// @type org.eclipse.core.resources.IProject
var project = getProject("sample_data");
if (!project.exists())
{
```

```
  print("Can't find project");
  exit(0);
}
// to get this script work , please choose CSV_EXCHANGE_SAMPLE.csv file
// from other_formats folder
// @type org.eclipse.ease.modules.platform.IFileHandle
let pathToSelectedFile = showFileSelectionDialog("workspace://",1,
    "Select File", "Select your File that has track information") ;
// create a file within the project
var file = openFile(pathToSelectedFile);
if (!file.exists())
{
  print("Can't find file");
  exit(0);
}
var track = null;
// ok, now loop through the file contents
var nextLine;
var ctr = 0;
while ((nextLine = readLine(file)) != null)
{
  ctr++;
  if (ctr > 2)
  {
    /**
     * Note: format looks like this: Two header lines,
     * then rows of CSV entries.
     * # UK TRACK EXCHANGE FORMAT, V1.0 #
     * Lat,Long,DTG,UnitName,CaseNumber,Type,Flag,Sensor,
       MajorAxis,SemiMajorAxis,SemiMinorAxis,Course,Speed,Depth,
       Likelihood,Confidence,SuppliedBy,Provenance,InfoCutoffDate,
       Purpose,Classification,DistributionStatement
     * 22.1862861,-21.6978806,19951212T050000Z,NELSON,D-112/12,
       OILER,UK,S2002,1.0,0.5,0.5,269.7000,2.0000,0.0,Remote,Low,
       UNIT_ALPHA,NELSON,19951212,For
     * planning,PUBLIC,"Quite a content."
    */
   var partsOfStr = nextLine.split(',');
     if (track == null)
     {
        // track not created yet. Go for it.
        track = createTrack(partsOfStr[3])
     }
     // location
     let dLat = parseFloat(partsOfStr[0]);
     let dLong = parseFloat(partsOfStr[1]);
     let location = createLocation(dLat, dLong, 0);
     // dtg components
     let dtgStr = partsOfStr[2];
     let yrs = dtgStr.substring(0, 4);
     let mons = dtgStr.substring(4, 6) - 1;
     let days = dtgStr.substring(6, 8);
     let hrs = dtgStr.substring(9, 11);
     let mins = dtgStr.substring(11, 13);
     let secs = dtgStr.substring(13, 15);
     // date object
     let dtg = createDateCalendarFormat(yrs, mons, days, hrs, mins, secs);
     // course and speed
```

```
      let course = partsOfStr[11];
      let speed = partsOfStr[12];
      // create the fix
      // @type Debrief.Wrappers.FixWrapper
      let fix = createFix(dtg, location, course, speed);
      // store the fix
      track.addFix(fix);
  }
}
if (track == null)
{
  exit(0)
}
// ok, now add the new track to the plot
let plot = getActivePlot();
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers();
layers.add(track);
layers.fireModified();
```

We will run this script on the file **boat2.rep** which has one track **COLLINGWOOD**. When prompted select the file **CSV_EXCHANGE_SAMPLE.csv** file, from other_formats folder, which has one track **NELSON**. After running the script, the file **boat2.rep** will have two tracks **NELSON** and **COLLINGWOOD**, which will update in the **Outline** view and the **Plot Editor**. The result is shown in the below screenshot:



1.7. Writing Calculated Values to Text File

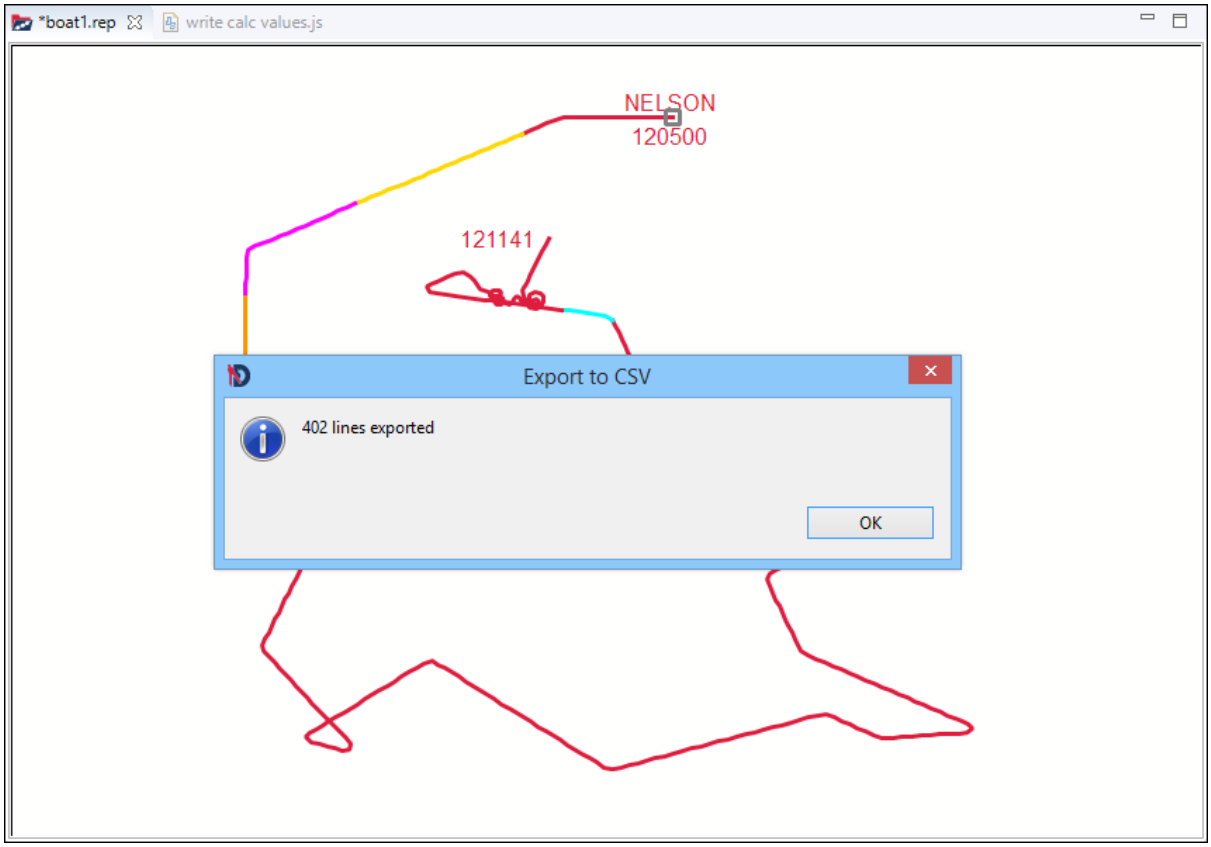
Scripting, in **Debrief,** is not only useful for reading in data, but it is also useful in producing outputs of new derived datasets. This script will help you do that:

```
loadModule('/System/Resources', false);
loadModule('/Debrief/Core'); // date
loadModule("/System/UI", false); // for the input dialogs
// get project instance
var project = getProject("sample_data");
if (!project.exists())
{
 print("Can't find project");
 exit(0);
}
```

```
// @type org.mwc.debrief.scripting.wrappers.Plot
let plot = getActivePlot(); // get the current plot
if (plot == null) // did we find a Debrief plot?
{
 print("There is no active plot");
 exit(0);
}
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers(); // get the layers for the plot
let lNames = layers.getTrackNames(); // get a list of layer names
let choice = showSelectionDialog(lNames,
  "Which track?","Choose track to export"); // let the user select a name
if (choice == null) // did user select something?
{
 print("You must choose an option");
 exit(0);
}
// create a file within the project
var file = openFile("workspace://sample_data/other_formats/tutorial_"
+ choice + "_export.csv", WRITE, true);
if (!file.exists())
{
 print("Can't create file");
 exit(0);
}
writeLine(file, ";; elapsed (secs), course rate (degs/sec)");
let track = layers.findTrack(choice); // try to get the selected layer
if (track == null) // did we find it the annotations layer?
{
 print("Layer was not found");
 exit(0);
}
var ctr = 0;
let fixes = track.getPositionIterator(); // to loop through positions
var lastCourse = null;
var lastTime = null;
var firstTime = null;
while (fixes.hasMoreElements()) // while there are more elements
{
 // @type Debrief.Wrappers.FixWrapper
 let fix = fixes.nextElement(); // get the next element
 let thisCourse = fix.getCourseDegs();
 let thisTime = fix.getDTG().getDate().getTime();
 if (firstTime == null)
 {
 firstTime = thisTime; // remember the start time
 }
 if (lastCourse != null)
 {
 let courseDelta = thisCourse - lastCourse;
 let timeDelta = (thisTime - lastTime) / 1000.0;
 let courseRate = courseDelta / timeDelta;
 let elapsed = (thisTime - firstTime) / 1000.0;
 let str = elapsed + ", " + courseRate;
 writeLine(file, str);
 }
 lastCourse = thisCourse;
 lastTime = thisTime;
```

```
 ctr++;
}
showInfoDialog(ctr + " lines exported", "Export to CSV");
closeFile(file);
```

When you run the script, in this case on **boat1.rep**, the dialog showing the number of lines exported will display.



The output file, in CSV format, will be saved in the sample_data folder of your **Debrief** installation.

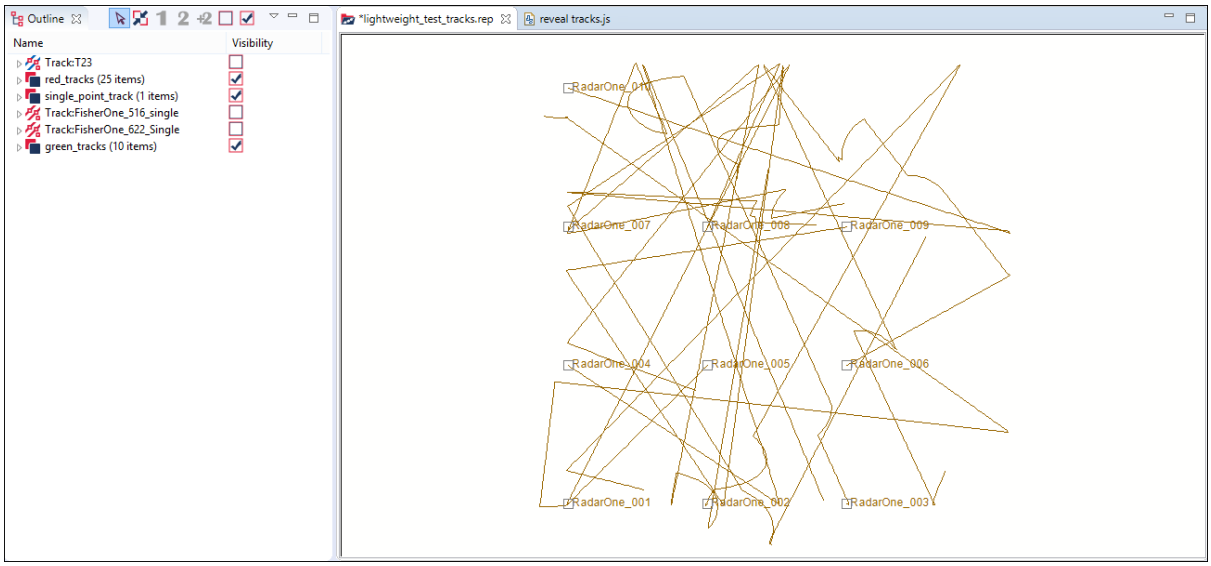
1.8. Revealing All Tracks in a Particular Color

At times you would want to modify all the elements in the plot that match certain criteria; instead of individually selecting elements to modify. The script below reveals all the tracks that are of a selected color, from the plot.

```
/**
* name: filter to only show tracks with matching color
**/
loadModule("/System/UI", false); // for the input dialogs
loadModule('/Debrief/Core'); // for plot, duration, date
/**
* function to return the unique entries in the provided list
from here
* https://coderwall.com/p/nilaba/
        simple-pure-javascript-array-unique-method-with-5-lines-of-code
**/
Array.prototype.unique = function()
{
  return this.filter(function(value, index, self)
  {
```

```
    return self.indexOf(value) === index;
  });
}
let plot = getActivePlot(); // get the current plot
if (plot == null) // did we find a Debrief plot?
{
  print("There were no active plot");
  exit(0);
}
// get the layers for the plot
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers();
let tracks = layers.getTracks();
// collate a list of symbols
var colors = [];
let len = tracks.length; // find number of tracks
if(len== 0)
{
   showInfoDialog("No tracks found", "Filter colors");
   exit(0);
}
for (var i = 0; i < len; i++) // loop through tracks
{
    let track = tracks[i]; // get this track
    let color = track.getColor(); // get the track color
    colors[i] = color; // add this color to the list
}
// find the unique list of colors
colors = colors.unique();
// ask the user which one to filter
let chosenColor = showSelectionDialog(colors,
  "Choose symbol to match","Filter to matching color");
if (chosenColor == null) // did one get chosen?
{
  exit(0); // ok, drop out.
}
// now loop through, and hide any that don't match
for (var i = 0; i < len; i++)
{
  //@type Debrief.Wrappers.TrackWrapper
  let track = tracks[i]; // get the next track
  // @type java.awt.Color
  let color = track.getColor(); // get its color
  let isVis = color == chosenColor; // does this match the chosen one?
  track.setVisible(isVis); // set visibility accordingly
}
layers.fireModified();
```

The result of the above script on the file **lightweight_test_tracks.rep**, showing the tracks of the color with **RGB** code **153,102,0,** is shown in the below screenshot:



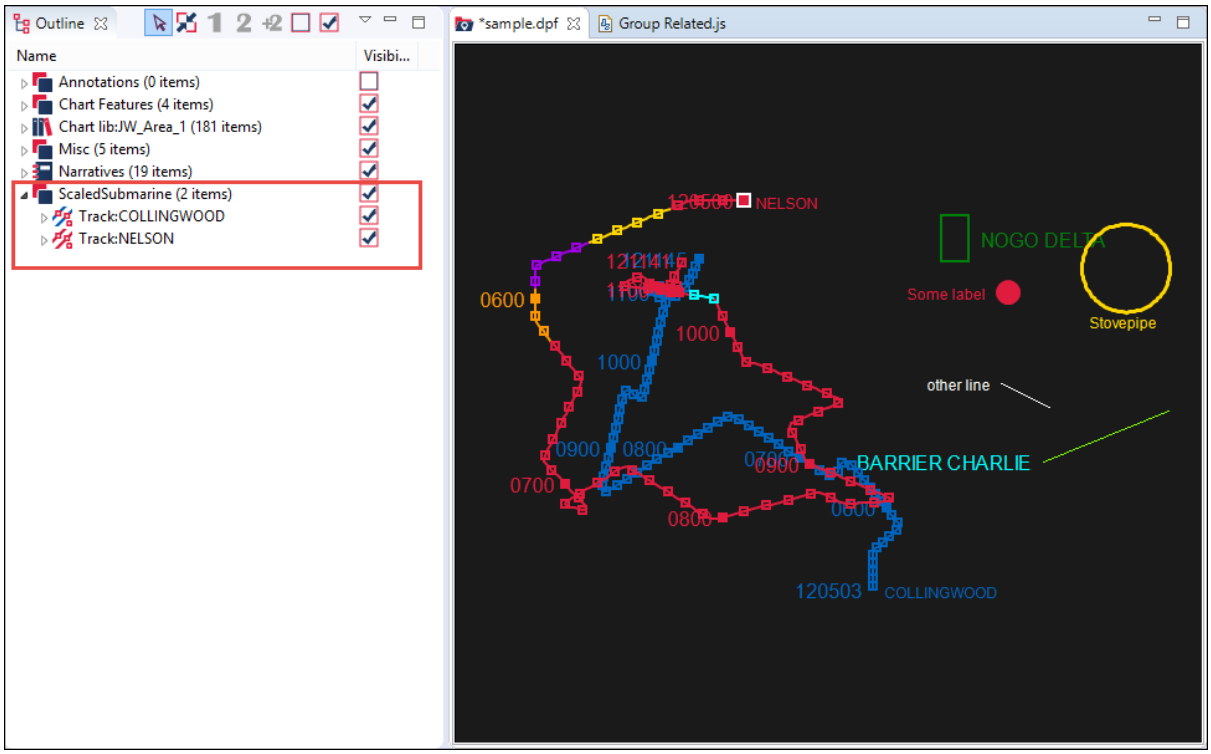
1.9. Grouping Related into a New Layer

A common operation which is applied to similar tracks is converting them into lightweight tracks. On the next level would be controlling the lightweight tracks together which can be achieved by organizing them into a layer. Sometimes the presence of a characteristic, such as the use of a platform-type symbol on a track, can be used to control how to operate on tracks for other platforms of that type. This script will group all the tracks, in a plot, with matching symbol into a new layer.

```
/**
* name: Group all the tracks with matching symbol
 *
*/
loadModule("/System/UI", false); // for the input dialogs
loadModule('/Debrief/Core'); // for plot, duration, date
/**
* function to return the unique entries in the provided list from here:
* https://coderwall.com/p/nilaba/
       simple-pure-javascript-array-unique-method-with-5-lines-of-code
*/
Array.prototype.unique = function() {
  return this.filter(function(value, index, self) {
    return self.indexOf(value) === index;
  });
}
let plot = getActivePlot(); //get active opened plot
if (plot == null) // did we find a Debrief plot?
{
  print("No track found");
  exit(0);
}
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = plot.getLayers(); // get the layers for the plot
let tracks = layers.getTracks();
// collate a list of symbols
var symbols = [];
let len = tracks.length; // find number of tracks
// do we have?
if(len== 0)
{
```

```
  showInfoDialog("No tracks found", "Filter symbols");
  exit(0);
}
for (var i = 0; i < len; i++) // loop through tracks
{
  let track = tracks[i]; // get this track
  let symbol = track.getSymbolType(); // get the symbol type
  symbols[i] = symbol; // add this symbol to the list
}
// find the unique list of symbols
symbols = symbols.unique();
// ask the user which one to filter
let chosenSym = showSelectionDialog(symbols,
    "Choose symbol to match","Filter to matching symbol");
if (chosenSym == null) // did one get chosen?
{
  exit(0); // ok, drop out.
}
// Then we create the new layer with the symbol name.
let newLayer = layers.createLayer(chosenSym);
/**
* now loop through, and move the track to the new layer if it matches
* with the symbol name
* */
for (var i = 0; i < len; i++)
{
  let track = tracks[i]; // get the next track
  let symbol = track.getSymbolType(); // get its symbol
  if ( symbol == chosenSym )
  {
   newLayer.add(track);
   layers.remove(track);
  }
}
layers.fireModified();
```

Run this script on the file **sample.dpf**, the tracks in the plot, **NELSON** and **COLLINGWOOD**, will be grouped under the selected symbol **ScaledSubmarine** and the result can be seen in the **Outline** View.



1.10. Making All Annotations Live for a Duration

Scripts can be used to perform bulk tasks which would otherwise take hours of effort. This script will make all the annotations in the selected layer, in the plot, live for a particular duration of time. And this is achieved by changing the finishing time of the scenario.

```
loadModule("/System/UI", false); // for the input dialogs
loadModule('/Debrief/Core'); // for plot, duration, date
let plot = getActivePlot(); // get the current plot
if (plot == null) // did we find a Debrief plot?
{
  print("There is no active plot");
  exit(0);
}
let layers = plot.getLayers(); // get the layers for the plot
let lNames = layers.getLayerNames(); // get a list of layer names
let choice = showSelectionDialog(lNames, "Which layer",
      "Choose ellipse layer"); // let the user select a name
if (choice == null) // did user select something?
{
  print("You must select an option");
  exit(0);
}
let anns = layers.findLayer(choice); // try to get the selected layer
if (anns == null) // did we find it the annotations layer?
{
  print("Layer selected was not found");
  exit(0);
}
// ok, sort out the duration
let mins = showInputDialog("How many minutes?", "10",
                     "Set ellipse duration");
let duration = createDuration(mins, DUR_MINUTES);
```

```
// now loop through the layer
let numer = anns.elements(); // retrieve the elements in this layer
while (numer.hasMoreElements()) // loop through the items in this layer
{
  let item = numer.nextElement(); // get the next element
  // @type java.lang.String
  let name = item.toString(); // get the string label for the element
  if (name.startsWith("Ellipse")) // see if it starts with "Ellipse"
  {
      let start = item.getStartDTG(); // retrieve the start date
      let newTime = (start.getDate().getTime() + duration
        .getValueIn(DUR_MILLISECONDS)); // calculate the new end time
      item.setEndDTG(createDate(newTime)); // set the new end time
  }
}
layers.fireModified();
```

Follow the below steps to execute the script:

1. Drag and drop the file **shapes.rep** into the **Plot Editor**.

2. In the **Outline** view, hide all the layers except the **TUAs (7 items)** layer and the 7 child items (Ellipses) under it.

3. In the **Properties** view, under the section **Time-Related**, note the **Time end** of each ellipse under the TUAs will display as **Unset**.

4. Now run the script on the file **shapes.rep**.

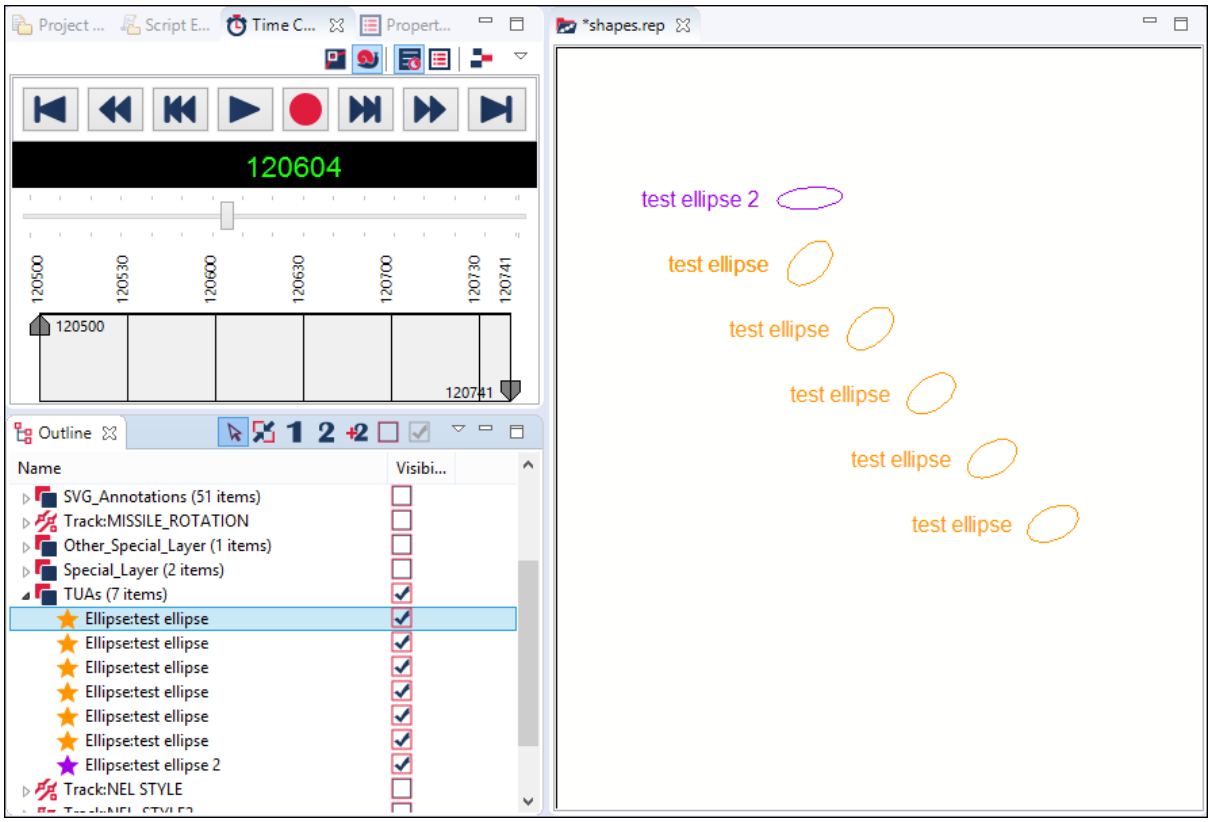
5. When prompted, select the ellipse layer **TUAs** and set the duration to **30** minutes.

6. In the **Properties** view, under the section **Time-Related**, note the change in the **Time end** of each ellipse. It will display **Time start** plus 30 minutes (the duration set in the previous step).

7. In the **Time Controller**, switch to **Snail** mode and drag the slider to **120552**. You will notice that first test ellipse will display in the **Plot Editor**.

8. Continue to drag the slider. For every two seconds you move the slider, the consecutive test ellipse will display in the **Plot Editor**.

The result of the script is shown in the below screenshot:



1.11. Reusing Code from Another File

As your scripting skills increase, you may want to group commonly used code snippets into utility files. This script will demonstrate how to refer to such commonly grouped script files.

Create a script file **hello.js** with the following code:

```
loadModule('/System/UI');
showInfoDialog("Hello World", "Title")
```

Create another script file **testCall.js** with the following code:

```
// Include must contain the project name, followed by the
// complete path. You could include the “workspace:/” tag:
include("workspace:/project/scripting/hello.js")
```

Note: You can copy the complete path of the workspace by navigating to **Window > Preferences > Scripting > Script Locations**.

As an extension, you can include functions/methods created in different files. Now create the script file **sum.js**.

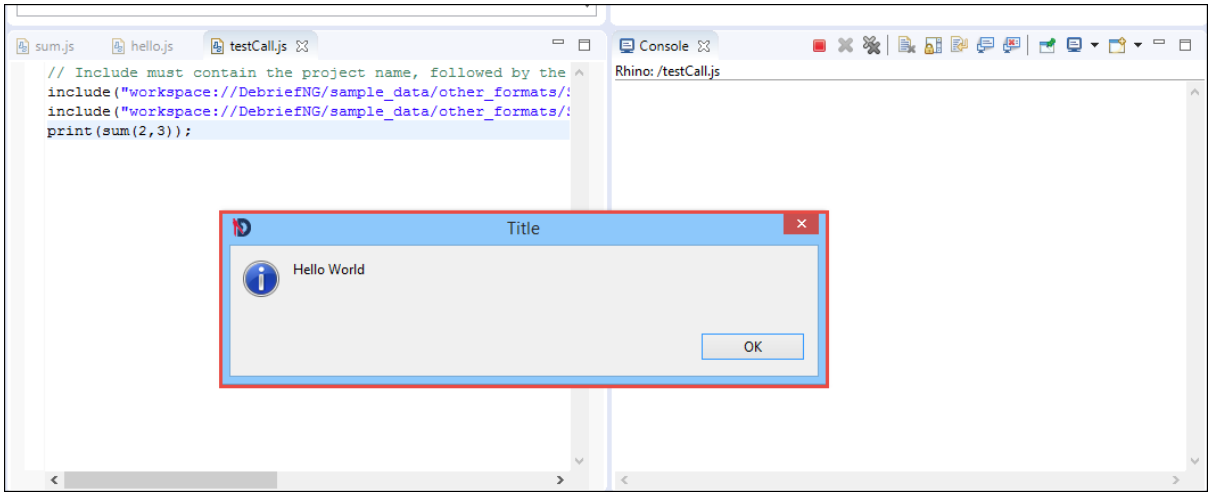
```
function sum(a, b)
{
  return a+b;
}
```

Now let us call the scripts **sum** and **hello** in the script **testCall.js**.

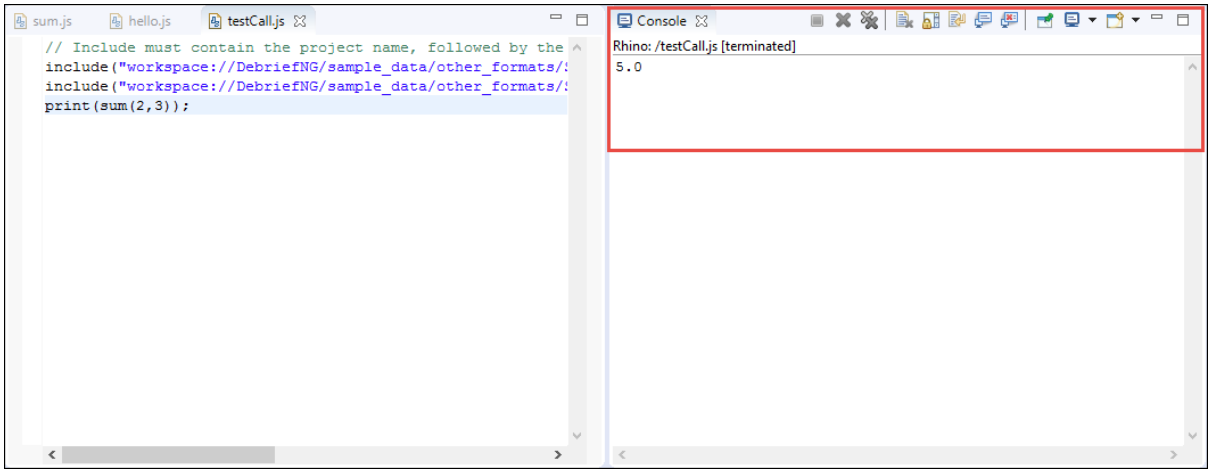
```
// Include must contain the project name, followed by the complete path.
// You could include the “workspace:/” tag
include("workspace://DebriefNG/sample_data/other_formats/Scripts/hello.js")
include("workspace:/project/scripting/sum.js")
```

```
print(sum(2,3));
```

The script **hello.js** will execute first and the dialog with the message **Hello World** will display.



Click **OK**. Then the next script **sum.js** will execute and the sum of the two integers will be displayed in the **Console**.



Chapter D.3. GeoTools in Debrief

1. Introduction

The display of chart data in Debrief is all down to the *GeoTools* [http://geotools.org/] library. Recently the SWT module was added to this library. The SWT module incorporates the SwtMapPane class upon which the Debrief plot now sits. With the incorporation of GeoTools brings a wide range of new file formats, both image and vector, plus more advanced topographic and earth model related capabilities. Rest assured, Debrief will exploit these library features as much as possible in coming months/years.

Chapter D.4. Debrief algorithms

1. Range/Bearing calculations within Debrief

Since Spring 2013, ranges and bearings within Debrief are calculated using the GeoTools GeodeticCalculator [http://svn.osgeo.org/geotools/trunk/modules/library/referencing/src/main/java/org/geotools/referencing/GeodeticCalculator.java] derivative of the T.Vincenty approximation:

```
        /*
         * Solution of the geodetic inverse problem after T.Vincenty.
         * Modified Rainsford's method with Helmert's elliptical terms.
         * Effective in any azimuth and at any distance short of antipodal.
         *
         * Latitudes and longitudes in radians positive North and East.
         * Forward azimuths at both points returned in radians from North.
         *
         * Programmed for CDC-6600 by LCDR L.Pfeifer NGS ROCKVILLE MD
 18FEB75
         * Modified for IBM SYSTEM 360 by John G.Gergen NGS ROCKVILLE MD
 7507
         * Ported from Fortran to Java by Daniele Franzoni.
         *
         * Source: ftp://ftp.ngs.noaa.gov/pub/pcsoft/for_inv.3d/source/
inverse.for
         *         subroutine GPNHRI
         *         version    200208.09
         *         written by robert (sid) safford
         */
```

LEGACY ALGORITHM In the past, Debrief used the Rhumb-line calculation for the display of measured range-bearing and for the projection of data onto the monitor. Briefly, to convert from real-world to screen coordinates the following procedure was followed:

1. Determine the area represented by the current viewed data in world coordinates
2. Determine the centre of this data area - this is used as the origin of the data
3. For each data point to be plotted, find its range and bearing from the origin
4. Convert this range and bearing into a delta-x and delta-y in screen coordinates
5. Produce a data point in screen coordinates by adding these deltas to the centre of the screen

Note

The result of this projection algorithm (as with most others) is that whilst the information plotted at the centre horizontal section (mid-latitude) of the screen is an accurate representation, travelling further north and south from it degrades the accuracy. This is negligible in data areas near the equator or where only a small area is covered (less than a couple of hundred nautical miles). With greater data areas, and nearer the poles, the effects are more noticeable, however.

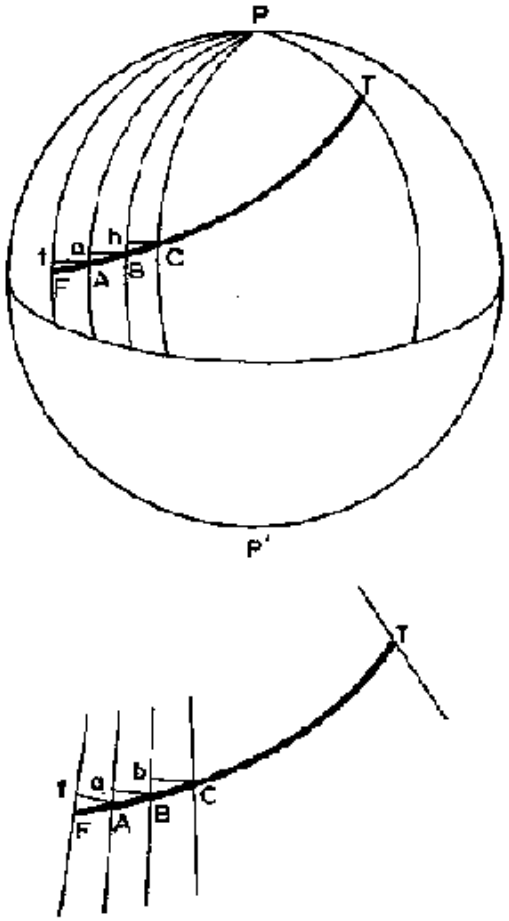
For those without access to the Admiralty Manual of Navigation, here is an abbreviated snippet from it:

1.1. LEGACY - The Rhumb-Line formulae

A line on the earth's surface which cuts all meridians at the same angle is called a rhumb line.

With his knowledge of the distance along a parallel of latitude and the departure between two places, the navigator can find the course he must steer in order to follow the rhumb line joining the two places, and also the distance he will travel along whilst doing so. In Figure D.4.1, “Rhumb line parts” FT is the rhumb-line divided into a large number of equal parts FA, AB, BC...

Figure D.4.1. Rhumb line parts



Af, Ba, Cb... are the arcs of parallels drawn through A, B, C ... and the angles at f, a, b ..., are therefore at right-angles. If the divisions of FT are made sufficiently small, the triangles FaF, ABa, BbB ... are themselves small enough to be treated as plane triangles. Also, since the course angle at F, A, B, C, ... remains constant by the definition of a rhumb-line, these small triangles are equal.

1.2. LEGACY - Short-distance sailing

By the term *short-distance sailing* is meant the following of a rhumb-line track for a distance not greater than 600'. Within this limiting distance, the navigator can obtain all he wants to know about the track from the three formulae:

```
departure = d.long cos (mean latitude)   (1)
departure = distance sin (course)     (2)
d.lat = distance cos (course)         (3)

The course is given by (2) divided by (3). Thus:
departure / d.lat = tan (course)
```

2. Worm in the hole algorithm

2.1. Overview

Frequently vehicles in Debrief use sensors with a different datum to the host platform - either because they're several metres from the position recorded datum on board the platform, or because they're an offboard sensor dragged behind the sensor. For an onboard sensor it is acceptable to locate the sensor by reference to its sensor length offset in conjunction with the current heading. However for an offboard sensor this coarse calculation may not be representative of the actual sensor location. When a platform is on a straight-line course, it is acceptable to plot the sensor datum back along the reverse heading. But, when a platform has travelled through a turn, plotting the sensor datum at the reciprocal of the platform heading will place the sensor in an unrealistic location. To counter this, Debrief offers a 'Worm in the hole' sensor locating model.

The 'worm in the hole' algorithm takes the sensor length offset value, and determines the point this specified distance back along the platform track. The metaphor is that a worm has opened a tunnel through a body of material. The sensor is being dragged through this tunnel, and we determine the sensor location accordingly.

Note

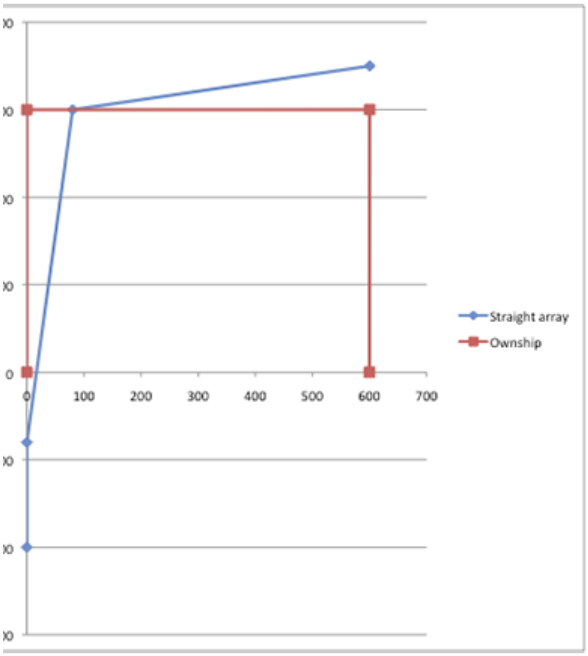
This is not a perfect algorithm. In truth, during a turn the forces on a towed array will cause it to slip sideways, cutting off the corner to an extent. It is the responsibility of the analyst to appraise himself with array performance and determine the applicability of the worm in the hole algorithm - though note its results are much more accurate than the non-worm in the hole straight line model.

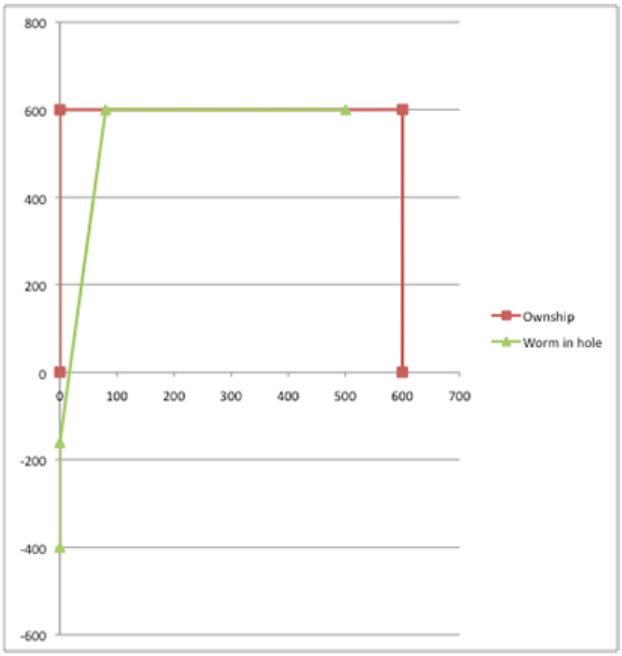
2.2. Example

The screenshot belows shows a comparison of the two models. The red ownship track has positions at times 100,200,300 and 400. The sensor track has positions at times 100,140,280,350. The array is 400m long.

The two array models show the same value for the first 3 points plotted: in the first two of these the array falls behind the start of the ownship track - thus the reciprocal of ownship heading is necessary to determine the position. In the third position, the length of the 400m array falls completely within the 600m leg.

The fourth position is, however, different. The straight-line model plots it back along the reciprocal of the platform heading at time 300. The worm in the hole model, however, turns back along the ownship track from position 200 to 300. At time 350, the ownship platform will be 1/2 way down between positions 300 and 400 - this will consume 1/2 of the 600m separation (300m), leaving another 100m of array to be consumed - this the algorithm moves 100m back down between the points at 200 & 300.





2.3. Algorithm

The worm in the hole algorithm is implemented as follows:

1. Determine platform track, the time at which sensor location is required, the specified length offset, and whether the sensor is 'ticked' for 'worm in the hole algorithm'.

2. Work through the platform track to find the position immediately after the specified dtg, together with a list of points preceding that. These represent the 'nextPoint' and the backTrack from that dtg.

3. Calculate the platform position at the specified dtg by interpolating between the last item of the backTrack and the nextPoint , at the specified dtg. Name this new position nextPoint - it represents the platform position at the specified dtg

4. Now repeatedly perform the following:

   a. Work out the distance from nextPoint to the last item in the backTrack

   b. If this distance is greater than the remaining array offset length then:

      i. Work out the proportion of the distance that the remaining sensor length represents, name it timeDelta

      ii. Interpolate back from nextPoint to the last item in the backTrack by the timeDelta parameter. This is the resultant sensor location

   c. If the calculated distance is less than the remaining sensor length then

      i. reduce the remaining sensor length by the calculated distance (to represent the array being 'consumed' by the backTrack)

      ii. if there are any more points in the backTrack then

         A. store the last point on the backTrack as nextPoint, and move back along the backTrack by one element

      else

         A. calculate the offset back from the start of the platform track, of the remaining sensor length, along the reciprocal of the vessel heading. This is the resultant sensor location.

3. Other Debrief algorithms

3.1. Frequency algorithms

In the Single Sided Reconstruction extensions to Debrief (2009) the use of frequency data in track reconstruction, specifically involving the application of Doppler Offsets. The algorithms used have been loosely taken from the Royal Naval Submarine School training guide titled *The Bearing Frequency Plot and Narrowband Analysis*. The core doppler calculation is as follows:

```
 private static double calcDopplerComponent(final double theBearingRads,
      final double myCourseRads,
      final double mySpeedKts, final double observedFreq)
 {
  final double speedOfSoundKts = 2951;
  double relBearingRads = theBearingRads - myCourseRads;

  final double ownSpeedAlongKts = Math.abs(Math.cos(relBearingRads)
    * mySpeedKts);

  // put rel brg into +/- 180 domain
  while (relBearingRads > Math.PI)
   relBearingRads -= (2 * Math.PI);
  while (relBearingRads < -(Math.PI))
   relBearingRads += (2 * Math.PI);

  double dopplerOffset = (ownSpeedAlongKts * observedFreq) /
 speedOfSoundKts;

  if (Math.abs(relBearingRads) < (Math.PI / 2))
   dopplerOffset = -dopplerOffset;
  return dopplerOffset;
 }
```

The doppler calculation is used in calculation of the corrected frequency - which involves the removal of the ownship doppler component:

```
 public double getCorrectedFrequency()
 {
  double correctedFreq = 0;

  final double theBearingDegs = getBearingToTarget();
  final double theBearingRads = MWC.Algorithms.Conversions
    .Degs2Rads(theBearingDegs);
  final double myCourseRads = _hostFix.getCourse();

  final double mySpeedKts = _hostFix.getSpeed();
  double observedFreq = _sensor.getFrequency();
  final double dopplerComponent = calcDopplerComponent(theBearingRads,
    myCourseRads, mySpeedKts, observedFreq);

  correctedFreq = observedFreq + dopplerComponent;

  return correctedFreq;
 }
```

The next frequency calculation involves the addition of ownship and target dopplers to the base frequency:

```
 public double getPredictedFrequency()
```

```
{
 double predictedFreq = 0;

 if (_targetTrack instanceof RelativeTMASegment)
 {
  RelativeTMASegment rt = (RelativeTMASegment) _targetTrack;
  final double theBearingDegs = getBearingToTarget();
  final double theBearingRads = MWC.Algorithms.Conversions
    .Degs2Rads(theBearingDegs);
  final double myCourseRads = _hostFix.getCourse();

  final double mySpeedKts = _hostFix.getSpeed();
  double baseFreq = rt.getBaseFrequency();
  final double myDopplerComponent = calcDopplerComponent(theBearingRads,
    myCourseRads, mySpeedKts, baseFreq);

  final double hisCourseRads = _targetFix.getCourse();
  final double hisSpeedKts = _targetFix.getSpeed();

  final double hisDopplerComponent = calcDopplerComponent(Math.PI
    + theBearingRads, hisCourseRads, hisSpeedKts, baseFreq);

  predictedFreq = baseFreq - (myDopplerComponent + hisDopplerComponent);
 }
 return predictedFreq;
}
```

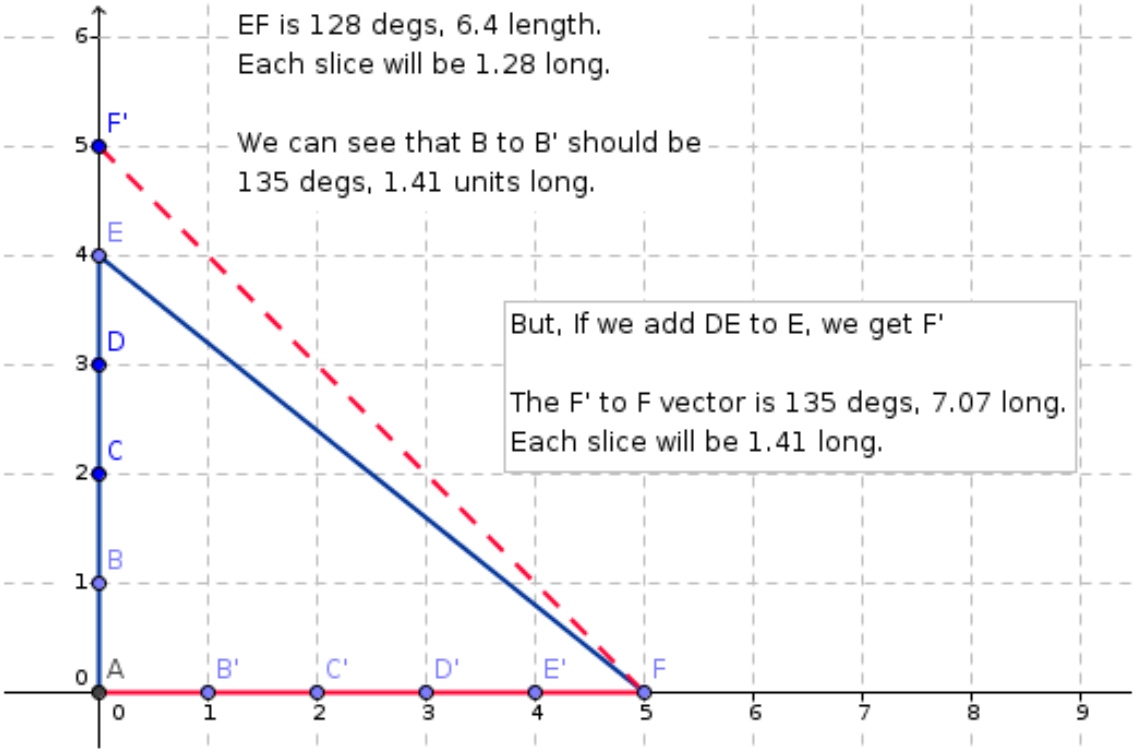
4. Remove Jumps

4.1. Background

Note

Read the user perspective on removing jumps in the user guide (Section 7.7, “Removing Jumps”)

The initial strategy to remove drift was to find the range/direction of the jump from the last dived point to the first GPS fix. But, during implementation it became clear that this wouldn't work. What was actually needed was to "plot-on" the last dived point to derive where the dived track would be at the time of the GPS fix. The following diagram shows a stylized representation of a period of dived drift.



In the diagram, the measured track is AEF, but the true track was AF. We can see that the correct location for B is B', C is C', and so on. So, the vector necessary to transform B to B' is 135 degs length 1.41 (sqrt(2)). This correct vector can be obtained by producing the virtual point F', which is defined as “where would F be if there hadn’t been a jump”. Using F'-F as the offset vector gives the correct result.

4.2. Algorithm

1. Analyst selects period of data.

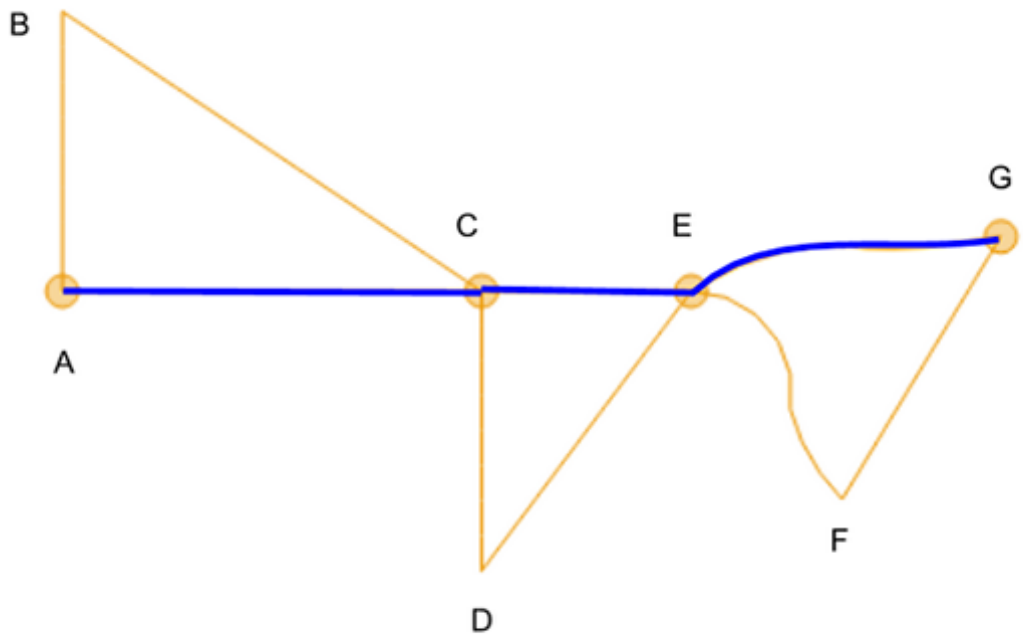
2. The first position is stored as a “lock point”.

3. Loop forwards through data, looking for large position deltas (where calculated speed equates to 3 or more time the measured speed travelled in the previous step).

4. Each time we find a jump, the position after the jump is a “lock point”, the position before jump is the “jump point”. The delta between the two is the “jump offset”.

5. From the first lock-point, move forwards through the data until you find a jump point. If a jump point is found, pass through those points, applying the relative sliced jump offset to each point.

The diagram belows shows the results of remove jumps. The yellow track ABCDEFG is the recorded track. The points A,C,E,G are the lock-points (GPS fixes). The blue track is the track after running remove jumps. Note that AC is longer than AB, so the algorithm has rotated and scaled AB. Similarly, CE is shorter than CD and gets scaled to produce CE. EG shows how a complex shape gets transformed to fit the lock points.



5. Contouring algorithm

The contouring algorithm employed within Debrief was originally produced by Paul Bourke back in 1987, for submission to the Byte magazine. Here's the article pretty much verbatim:

5.1. Introduction

This article introduces a straightforward method of contouring some surface represented as a regular triangular mesh.

Contouring aids in visualizing three dimensional surfaces on a two dimensional medium (on paper or in this case a computer graphics screen).

Two most common applications are displaying topological features of an area on a map or the air pressure on a weather map. In all cases some parameter is plotted as a function of two variables, the longitude and latitude or x and y axis. One problem with computer contouring is the process is usually CPU intensive and the algorithms often use advanced mathematical techniques making them susceptible to error.

5.2. CONREC

To do contouring in software you need to describe the data surface and the contour levels you want to have drawn. The software given this information must call the algorithm that calculates the line segments that make up a contour curve and then plot these line segments on whatever graphics device is available.

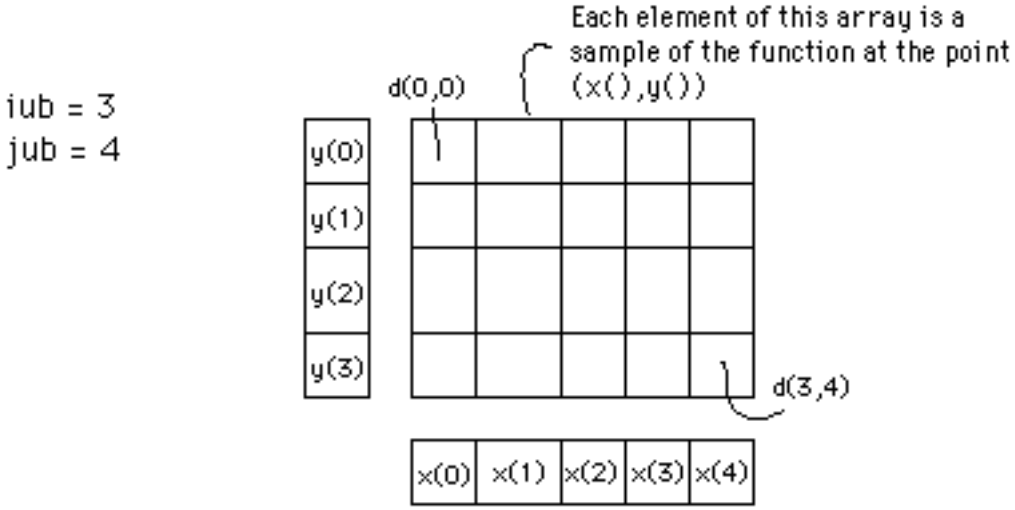
CONREC satisfies the above description, it is relatively simple to implement, very reliable, and does not require sophisticated programming techniques or a high level of mathematics to understand how it works.

The input parameters to the CONREC subroutine are as follows :

- The number of horizontal and vertical data points designated iub and jub.
- The number of contouring levels, nc.

- A one dimensional array z(0:nc-1) that saves as a list of the contour levels in increasing order. (The order of course can be relaxed if the program will sort the levels)
- A two dimensional array d(0:iub,0:jub) that contains the description of the data array to be contoured. Each element of the array is a sample of the surface being studied at a point (x,y)
- Two, one dimensional arrays x(0:iub) and y(0:jub) which contain the horizontal and vertical coordinates of each sample point. This allows for a rectangular grid of samples.

Figure D.4.2. Some of the CONREC input parameters.



The contouring subroutine CONREC does not assume anything about the device that will be used to plot the contours. It instead expects a user written subroutine called VECOUT. CONREC calls VECOUT with the horizontal and vertical coordinates of the start and end coordinates of a line segment along with the contour level for that line segment. In the simplest case this is very similar to the usual LINE (x1,y1)-(x2,y2) command in BASIC. See the source code listing below.

5.3. Algorithm

As already mentioned the samples of the three dimensional surface are stored in a two dimensional real array. This rectangular grid is considered four points at a time, namely the rectangle d(i,j), d(i +1,j), d(i,j+1), and d(i+1,j+1). The centre of each rectangle is assigned a value corresponding to the average values of each of the four vertices. Each rectangle is in turn divided into four triangular regions by cutting along the diagonals. Each of these triangular planes may be bisected by horizontal contour plane. The intersection of these two planes is a straight line segment, part of the contour curve at that contour height.

Depending on the value of a contour level with respect to the height at the vertices of a triangle, certain types of contour lines are drawn. The 10 possible cases which may occur are summarised below

a. All the vertices lie below the contour level.

b. Two vertices lie below and one on the contour level.

c. Two vertices lie below and one above the contour level.

d. One vertex lies below and two on the contour level.

e. One vertex lies below, one on and one above the contour level.

f. One vertex lies below and two above the contour level.

g. Three vertices lie on the contour level.

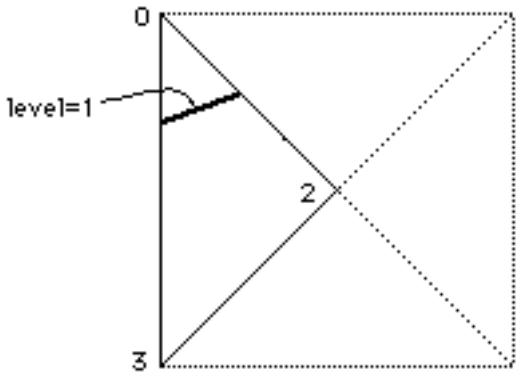
h. Two vertices lie on and one above the contour level.

i. One vertex lies on and two above the contour level.

j. All the vertices lie above the contour level.

In cases a, b, i and j the two planes do not intersect, i.e. no line need be drawn. For cases d and h the two planes intersect along an edge of the triangle and therefore line is drawn between the two vertices that lie on the contour level. Case e requires that a line be drawn from the vertex on the contour level to a point on the opposite edge. This point is determined by the intersection of the contour level with the straight line between the other two vertices. Cases c and f are the most common situations where the line is drawn from one edge to another edge of the triangle. The last possibility or case g above has no really satisfactory solution and fortunately will occur rarely with real arithmetic.

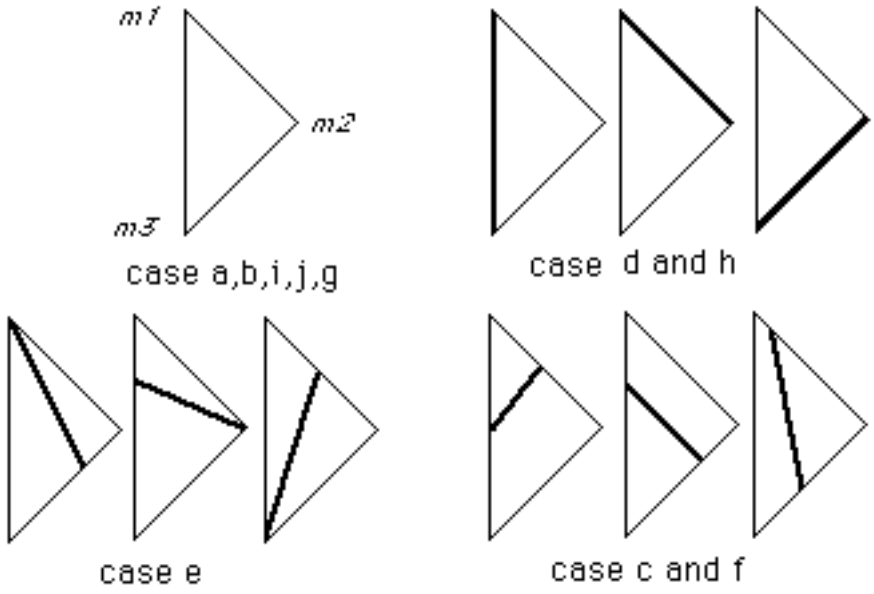
Figure D.4.3. Possible line orientations



5.4. Example

As a simple example consider one triangle with vertices labelled m1,m2 and m3 with heights 0, 2 and 3 respectively

Figure D.4.4. Line permutations within contouring algorithm.

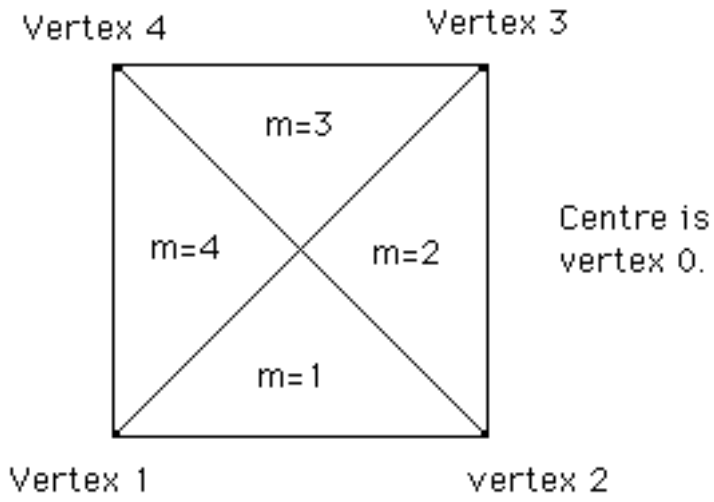


To calculate where a contour line at a height of 1 should be drawn, it can be seen that this is case f described earlier. Level 1 intersects line segment m1-m2 half the way along and it intersects line segment m1-m3 one third of the way along. A line segment is drawn between these two points. Each rectangular mesh cell is treated this way.

5.5. Subroutine

An attempt is made at optimization by checking first to see if there are any contour levels within the present rectangle and second that there are some contour levels within the present triangle. The indices i and j are used to step through each rectangle in turn, k refers to each contour level and m to the four triangles in each rectangle.

Figure D.4.5. Some of the notation used for identifying the rectangles and triangles in the subroutine



Note that for large arrays the whole data array need not be stored in memory . Since the algorithm is a local one only requiring 4 points at a time, the data for each rectangle could be read from disk as required.

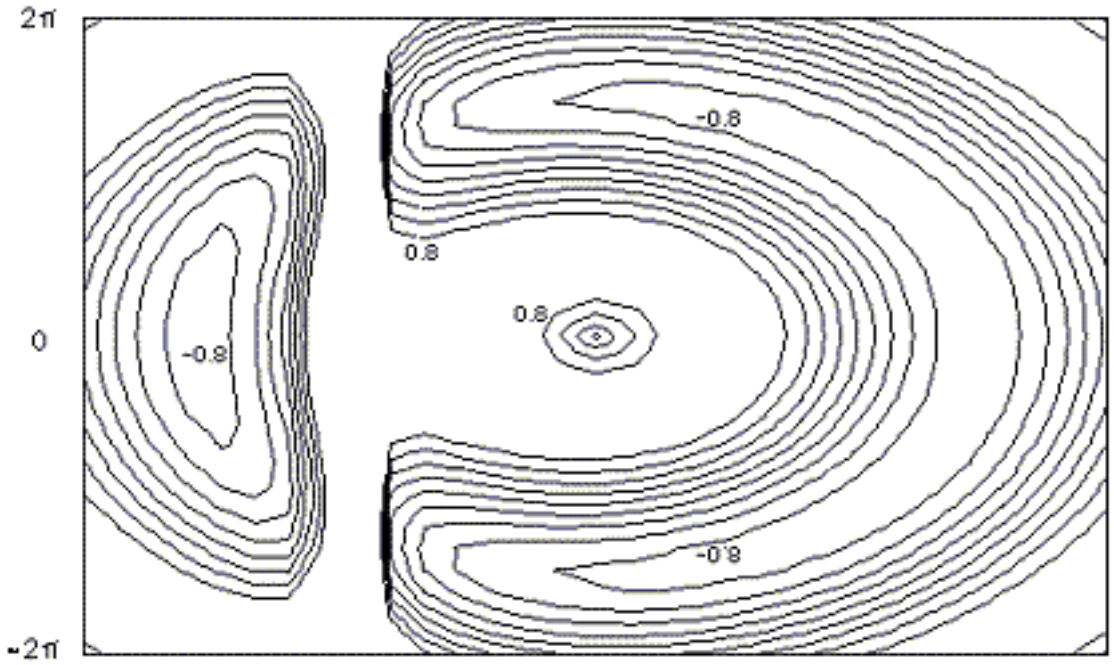
5.6. Example

Contour map and the following function

Figure D.4.6. Function to be contoured

$$f(x,y) = \sin\left((x^2+y^2)^{1/2}\right) + \frac{1}{((x-c)^2+y^2)^{1/2}}$$

Figure D.4.7. Sample contour plot using CONREC algorithm



Contouring algorithm

The sample contour plot itself

5.7. Note

On occasion users have reported gaps in their contour lines, this should of course never happen. There is however a pathological case that all local contouring algorithms suffer from (local meaning that they only use information in the immediate vicinity to determine the contour lines). The problem arises when all four vertices of a grid cell have the same value as the contour level under consideration. There are a number of strategies that can be employed to overcome this special event, the correct way is to consider a larger region in order to join up the contours on either side of the problem cell. CONREC doesn't do this and just leaves the cell without any contour lines thus resulting in a gap. This special case essentially never happens for real values data, it is most commonly associated with integer height datasets. The simplest solution is to offset the contour levels being drawn by a very small amount.

Chapter D.5. Semi Automated Track Construction (SATC)

1. High-Level SATC Concepts

This section records the high-level concepts related to the Semi-Automatic Track Construction (SATC) implementation in Debrief. The content presented here represents system documentation, not necessarily targeted at the routine SATC user, user guidance is contained in Section 3, “Semi-Automated TMA generation”.

1.1. Strategy

Some BOT strategies focus very strongly on the bearing data, with other aspects (priors, constraints, etc) used to refine the end product.

Since we're doing this process with hindsight, we have more data available. So we're taking a more free-form approach, where any type of information can contribute to the solution. This will make it easier to add other contributing information in the future.

Each time the analyst/user submits some information to be used in the algorithm he's contributing to the answer - so the piece of information is termed a contribution.

1.2. Process

In generating one or more vehicle solutions the system will first constrain the global problem space, then generate candidate solutions within that space, and lastly assess their performance. If none of the candidate solutions are acceptable the analyst may refine existing contributions, or supply more contributions to further inform the system. These phases are described in more detail later, once we've covered the necessary data concepts.

1.3. Data Concepts

1.3.1. Scenario

A single instance of performing SATC on a set of data, plus contributions is called a Scenario. This includes all contributions and their states. It also includes the measurements that inform a Measurement Contributions. A scenario may be saved to file, in order that its analysis may continue at another time.

1.3.2. Problem space

The problem space is a series of bounded vehicle state objects with a Vehicle Performance instance.

1.3.3. Vehicle Track

A track is comprised of a series of Vehicle State objects. It represents an estimate of the path followed by a vehicle for a period of time.

1.3.4. Leg

A leg represents a period of vehicle track. The leg may be a straight leg (representing the way vehicles normally move) or an alteration leg (representing how a vehicle makes the transition from one straight leg to another). The fact that the a set of Vehicle States (or Bounded Vehicle States) objects are in a straight or altering leg means we can infer further constraints/relaxations on them.

1.3.5. Vehicle State

A vehicle state is represented as a time, location, course and speed.

1.3.6. Bounded Vehicle State

A bounded vehicle state describes a range of possible vehicle states. It includes a course range (min-max, always clockwise), a speed range (min-max), and a polygon representing spatial constraints.

Note

We use a polygon for location instead of min/max values for lat/long values so that we gain access to more complex spatial operations.

Note

We may need to introduce non-continuous ranges for these states. We may need to constrain course, for example, to being in the range of either 040-060 or 130-150. But, as of Nov 2012 this requirement hasn't emerged yet.

1.3.7. Straight Leg

A straight leg is uniform, with constant course and speed. When constraining the problem space the leg will be formed from multiple bounded vehicle states, but when candidate solutions are being generated they will always have constant course and speed. During the Constrain the problem space phase we allow the straight leg to be comprised of varying bounded vehicle states to allow a manoeuvre detection algorithm to detect that there must actually be a course or speed change within the leg (since there isn't a single course or speed that is achievable along the whole leg), leading to the introduction of an alteration leg.

1.3.8. Alteration Leg

An alteration leg allows for change in course and/or speed. The alteration leg may represent a speed change, a course change, or a period for which insufficient data is available to suggest straight-line travel (in which case there may be an unpredictable number of course/speed changes). Consequently an Alteration Leg doesn't just have values for vehicle state at either end, it is expressed as a series of vehicle states (or bounded vehicle states in the constrain the problem space phase).

1.3.9. Contribution

A contribution is a piece of knowledge supplied by the analyst. Beyond the simple, abstract description provided below, specific contributions may contain great algorithmic complexity.

1.3.9.1. Contribution attributes

The contribution will always have at least one of

- *Hard constraints* - these are values that are used to constrain the problem space. Candidate solutions will not be considered that fall outside these constraints
- *Estimate* - these are values that are used to assess the performance of a candidate solution. The estimate may be a single value or a range. Candidate solutions will be assessed according to their proximity to the specified value/range.

It may also have:

- *Time bounds* - the period for which the contribution is valid. Note: a contribution does not need time bounds. For example, a hard constraint that a vehicle has a maximum speed of 20 knots is time independent.

- *Weighting* - used to affect the significance of this contribution when comparing candidate solutions (only used if estimate data is present)

1.3.9.2. Contribution abilities

A contribution is able to:

- *Constrain the problem space*: Take a set of bounded vehicle states, and return another set of bounded vehicle states. The returned set of bounded states will normally have been reduced in bounds by the contribution. But, the contribution may also have introduced new legs (with states). The introduction of new legs may be the product of a manoeuvre detection algorithm.
- *Assess candidate solutions*: Take a series of vessel states and measure the error from any estimate values/ranges.

1.3.10. Reference contribution

This type of contribution represents some known fact about the vehicle. It may include max/min attainable speed, max/min acceleration and deceleration rates, and max/min turning circles. These contributions are normally time independent, and typically only have hard constraints. Reference contributions are typically only effective in constraining the problem space, they do not assist in the assessment of candidate solutions. An example of a reference contribution is the Vehicle Performance Reference Contribution.

In implementation, it is expected that a ‘library' of reference data be made available to the analyst. The analyst may load a contribution from the library, but changes made to that contribution would only be made locally - the library would remain unchanged.

1.3.11. Forecast Contribution

This type of contribution represents an analyst's forecast of vehicle behaviour. The forecasts are typically time bounded. They normally have estimates, and may also have hard constraints: “the vehicle is probably travelling between 4 and 6 knots. It is certainly not travelling outside 2 and 10 knots”. These contributions are probably to be used in both constraining the problem space and assessing candidate solutions. Examples of a forecast contribution are the Speed Forecast Contribution and the Straight leg forecast contribution.

1.3.12. Measurement contribution

This type of contribution represents some measurement on the vehicle, such as that produced by a radar or sonar. The contribution is a time-stamped collection of measurements. Hard constraints may apply to error ranges on the sensor (bearing or range, for example). The measurement itself is interpreted as an estimate. An example of measurement contribution is the Bearing Measurement Contribution. The analyst should be provided with a means of suppressing some measurements. In the Debrief implementation it is proposed that this is done by setting them to not visible in the Outline View.

1.3.13. Analysis contribution

This type of contribution uses algorithms to either constrain the problem space, or assess the performance of a candidate solution - they contain no attributes. These contributions may not have to be explicitly specified by an analyst - they may be permanently present. An example of an analysis contribution is the Speed Analysis Contribution.

1.4. Algorithms

1.4.1. Constrain the global problem space

It is thought that the generation and assessment of candidate solutions is an expensive process, so the problem space will be constrained as far as possible prior to consideration of candidate solutions.

The analyst triggers the generation of a new solution, specifying a time period. Here's what happens next:

1. The algorithm initialises a problem space containing zero bounded state objects.

2. This unbounded set of states is then populated with any Reference Contributions - significantly using the vehicle performance characteristics (min/max speed).

3. Analyst-contributed measurement contributions are then applied to the bounded vehicle state. Measurement Contributions will typically generate a new bounded vehicle state for each measurement available.

4. The system then repeatedly applies the analysis contributions, reference contributions, forecast contributions until the bounded vehicle state becomes stable. We repeatedly run through the contributions since some of them are capable of generating new bounded vehicle constraints - to which hard constraints present in forecast contributions and reference contributions should be applied. Any changes in contributions then need to be processed by the analysis contributions.

Note

We may not achieve an unchanging bounded state due to perturbations in the algorithm, so we decide it's stable when there are negligible differences.

5. If the user has indicated Suppress Cuts then cuts are suppressed as follows:

   a. Decide how many cuts to be suppressed. Each precision level has a different %age, from 80% in Low Precision to 20% in High Precision

   b. Loop through the states, and identify each bounded state that doesn't represent the first or last state in a leg. For each of these "Available" states, determine the overlap (intersection) between its Location Polygon and the Polygon from the previous state.

   c. Store the "Available" states in ascending order of overlap since the previous polygon

   d. Delete the lowest 20% of these available states, or the required number to reach the target limit (whichever is the smaller value).

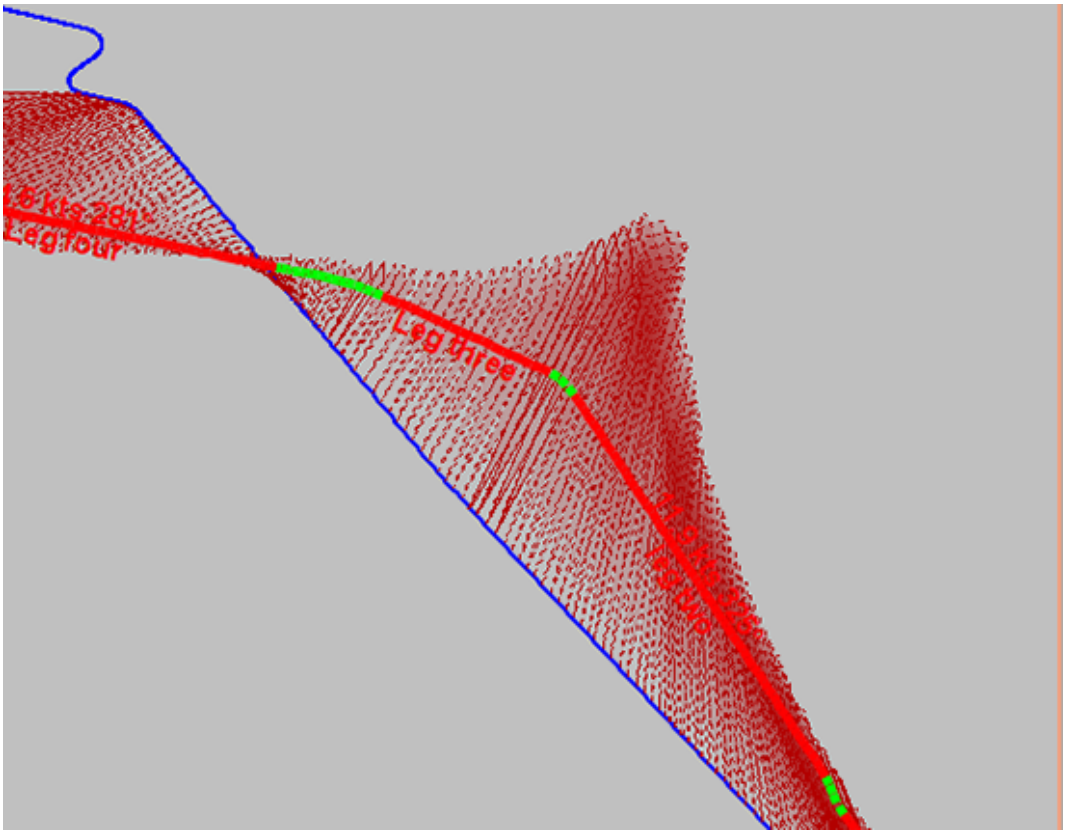
Note

We don't delete the required number of states in the first pass, we must remove a proportion at each step to ensure we retain the most significant location states.

Note

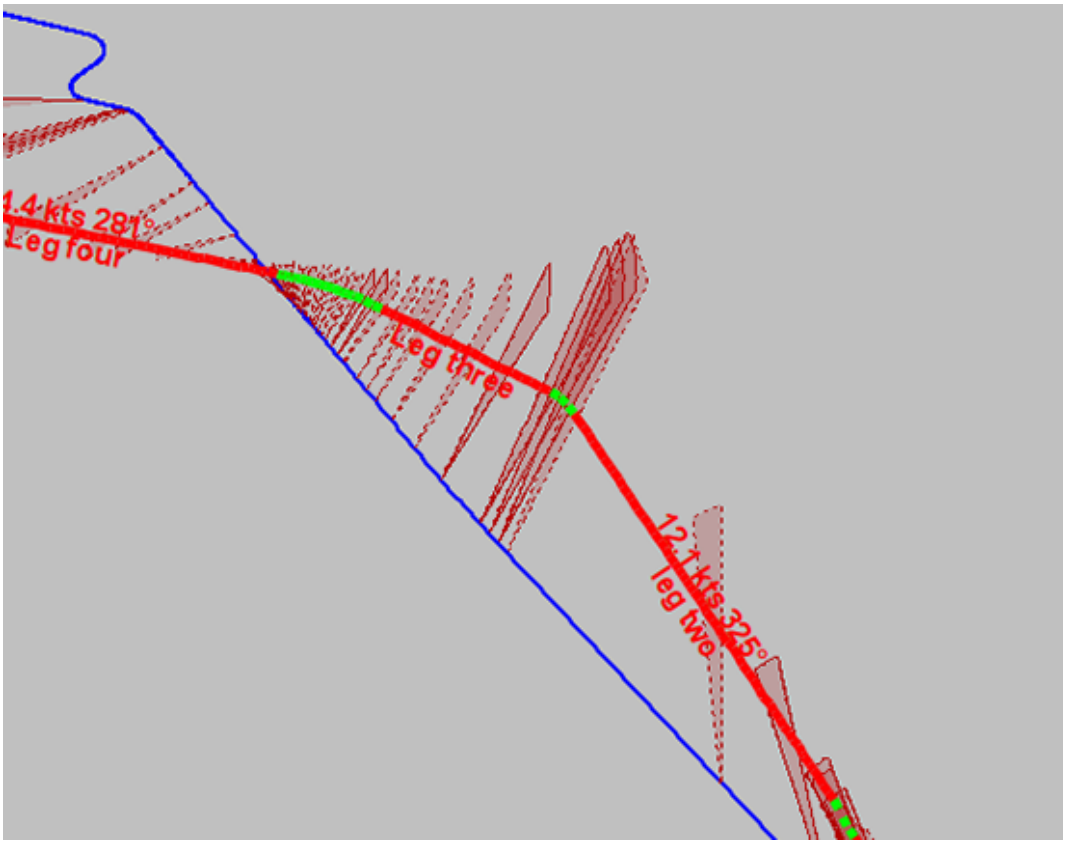
The following images show the effect of cut suppression. It is surprising how many cuts can be suppressed whislt still producing an effective solution.

Figure D.5.1. Solution showing all bounded states



A solution showing all bounded states

Figure D.5.2. Solution showing low precision cut suppression



A solution showing Low Precision cut suppression

On completion the algorithm moves on to generating and assessing candidate solutions.

1.4.2. Generate and assess candidate solutions

The system will investigate candidate solutions offering the single or multiple best candidates to the user.

1. A solution generator generates a candidate solution that falls within the problem space, defined as a series of vehicle states - one at each bounded vehicle state.

2. The candidate solution is then provided to the contributions, which return a performance score for that solution.

3. The score of a solution may then be passed to the solution generator to inform successive generations.

4. The system will then collate optimal solutions for presentation to the analyst.

5. The analyst may elect to convert one or more solutions into permanent vehicle tracks, tune existing contributions, or provide new ones.

The processing power available with modern desktop PCs is seen as an enabler that may allow a ‘brute-force' approach to solution assessment.

Beyond the brute-force approach, a feedback loop may inform the generation of successive solutions, using their assessed effectiveness. Such a loop would support the evolutionary production of candidate solutions, such as those in a genetic algorithm.

1.4.3. Solution generator

The solution generator is an algorithm that is able to take a problem space definition, and use the contributions to define possible course and speed permutations for a series of start points.

Solutions are generated leg-by-leg as follows:

- Apply a grid to the bounded locations for the start of the first leg.
- Apply a grid to the bounded locations for the end of the first leg.
- We can now create a matrix with start points down the left-hand side, and end points along the top. The matrix represents the relationship between each start point to each end point. We'll term it the “leg one attainable matrix”, since it will record which end points are attainable from each start point in leg one.
- Take the first point in the start grid.
- Pass this point to the set of contributions, and invite them to generate a set of bounded vehicle states specific to this start point. We can picture this set of bounded states as a polygon of possible end-points respective to this start points.
- Record in the matrix whether each end point is within this polygon.
- Now loop through the other start points.
- We now move along the solution, and consider the next leg. Do this by creating a “Leg two attainable matrix”. The set of end points for leg one now become the start points for leg two, and form the rows of the matrix. The end points for leg two form the columns.
- We repeat the process of working through the start points for this leg and determining the resultant polygon of achievable positions - marking which end points are achievable.
- This process continues until we have passed along the whole set of legs.
- Next we do some fancy maths to determine the attainable leg permutations: each leg matrix is multiplied against the matrix for the successive leg. The resulting matrix contains the possible leg permutations - from start points in the first leg, to end points in the last leg. So, it is only cells that have a non-zero value that need to be considered.
- At some point we need to consider introducing variance in the time periods for legs and contributions. I suspect we're ok doing this at the solution generation point - where we allow some subtle variance in the vessel state that is generated from the bounded vessel state.

1.4.4. Genetic algorithm

One methodology for producing candidate solutions is the use of a genetic algorithm. We already refer to the potential matching tracks as candidate solutions. We can then consider the error-from-estimate value as the fitness function for that solution. We can then consider the legs that comprise that solution as the discrete genes within the genotype.

Well, that makes reasonable enough sense, but it's 10 years since I've implemented a G.A. I'd rather work in the context that we have location, course, speed as the three genes that get compared - so that we're solving single leg solutions.

Maybe there's merit in a two-phase GA construct:

1. come up with a range of ‘fit' candidates for each leg (where the gene is loc, crse,spd).
2. then scale up so we have a much longer gene - a series of loc, crse, spd values representing all of the legs.

Note

Genetic Algortihms have proven to be the most versatile and effect optimisation strategy for SATC. The SATC implementation of GA is recorded in further detail at Section 3, “Solution Generator based on Genetic Algorithm”

1.4.5. Collate optimal solutions

It is expected that the system will generate and investigate many, many solutions. For some scenarios there may be one (or few) solutions with very high performance. These can be presented directly to the analyst for consideration. For other scenarios there may be many solutions that offer a similar performance result. It is proposed that the system has a means by which it offers distinct solutions to the analyst, in order to further refine the contributions. Such processes could include:

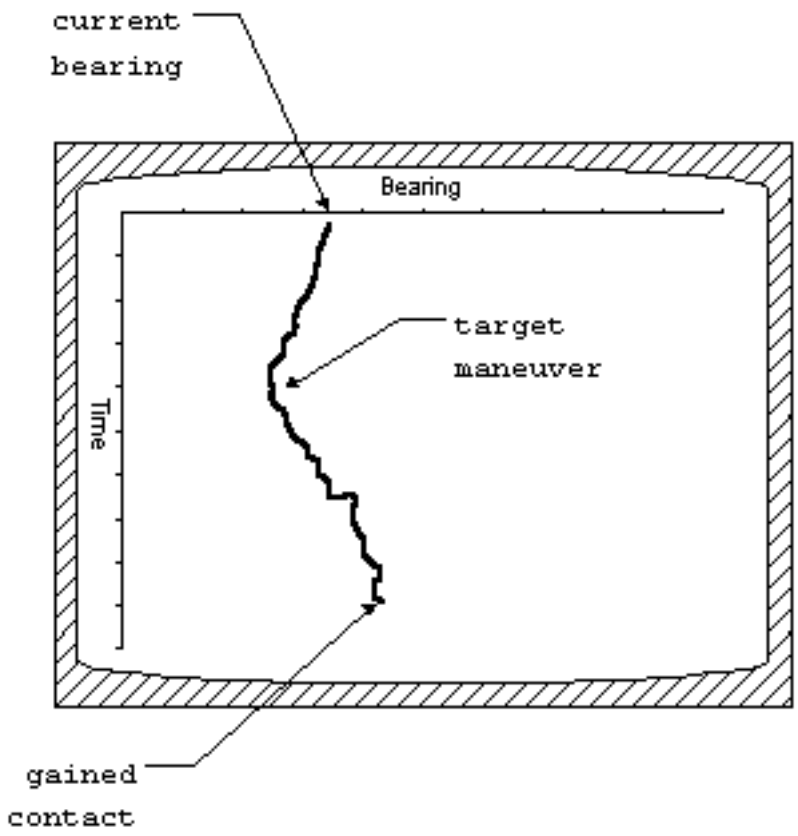
Extreme solutions	The system could examine the solutions, and highlight those that have have extremes of course, speed, location within the bounded vehicle states. For these extreme solutions the system could show how particular contributions ‘contributed' to their performance score - to allow the estimate or hard constraints on those contributions to be relaxed/tightened.
Bounded vehicle states	Alternatively, where too many solutions are provided, the system could (somehow) provide a visualisation of the set of bounded vehicle states. The analyst may determine that for a period of time there are insufficient or excessive constraints on the problem space and adjust the contributions accordingly to either tighten the space, or loosen the space to allow fresh solutions to be considered.

1.4.6. Manoeuvre Detection Algorithm

A measured contribution may be able to inspect its measurements in conjunction with the bounded vehicle states and determine that one of the legs is actually comprised of a combination of straight and alteration legs. For example, a measured frequency contribution may be able to recognise that a single leg is actually comprised of a straight leg, an alteration leg then another straight leg. Similarly, a contribution may be able to determine that an alteration leg actually contains a period of steady course and speed - and insert a straight leg covering that period. Note: if the leg has been created through an analyst's Straight leg forecast contribution specified as a hard-constraint then the leg will not be decomposed.

1.4.6.1. Bearing-based MDA

As two vessels move past each other, the bearings typically follow a fairly smooth/regular curve. Below is a very poor representation, but it is possible to produce a mathematical smoothing of the bearing curve representing “sensor and subject in steady state”. When the bearings start to diverge from this curve then something has changed - either the sensor or subject have changed their course/ speed vector. The only input necessary for this is the bearing data. But, it's of note that only the BearingMeasurementContribution knows the actual bearings, they're not visible outside. So, I believe the bearing-based MDA must sit “inside” the BearingMeasCont.



Note

While Bearing based MDA worked effectively with simulated data, it proved unable to recognise manoeuvres in real data. The simulated data typically had 3 decimal places of degrees bearing. But, our real world bearing typically only have one or zero decimal places - and the algortihms were unable to work with this low fidelity.

1.4.6.2. State-analysis MDA

Another way of detecting manoeuvres is by analysis of a series of bounded states. Where consecutive/adjacent states have incompatible bounds, we may be able to deduce that a manoeuvre must have happened. We may have a series of states with course bounds 040-080, then a series of states with a bounds 110-150. Clearly there must be a subject vehicle manoeuvre in between these states.

1.5. Examples of contributions

1.5.1. Vehicle performance reference contribution

1.5.1.1. Attributes

Hard constraints are available that specify that the vehicle has a minimum speed of 1 knot, and a maximum speed of 30 knots. The minimum turning circle is 1km radius. It is also known that the vehicle can decelerate at 1 knot per 5 seconds and accelerate at 1 knot per 10 seconds.

Potentially, such a constraint could also include an estimate on vehicle state, such as the vehicle normally travels between 8 and 12 knots.

1.5.1.2. Abilities

This contribution mostly affects the constrain the problem space phase.

- *Speed*. This contribution will examine all of the bounded vehicle states and ensure that all speeds are within those permissible for the platform. It will then pass through the bounded vehicle states and ensure that speeds in successive states are achievable given the known accel/decel rates.
- *Course*. The contribution will pass through the bounded vehicle states and ensure that courses in successive states are achievable given the known turn circles - constraining them as necessary.

Note

it is acknowledged that these two abilities will only really have an effect in alteration legs.

The contribution may affect the assess candidate solutions phase - the contribution will assess the candidate solution speeds and offer a performance measure based on the speed being within the normal speed range, or it's error outside that range.

1.5.2. Bearing measurement contribution

1.5.2.1. Attributes

This is a series of measured bearings. The bearings have a value (the estimate) plus an origin (sensor location). The dataset will also have error range (hard constraints). The error range may be relative bearing dependent, or the same for the whole dataset. The dataset may also have a maximum range value, beyond which no detection could be possible (another hard constraint).

1.5.2.2. Abilities

- *Constrain problem space.*

  For any leg received, the contribution will generate a bounded vehicle state for each time for which there is a measurement. The bounded vehicle state will contain a sector for the bounded location, but the other attributes will be unbounded. Note: we aren't using the course/speed bounds from the points before/after. These bounds may have been assigned by a time-limited Forecast Contribution, and we don't know the respective period. We leave them unbounded, and when the Forecast Contribution next runs it will fill in any bounded vehicle states in its time period.

  The combination of range, bearing and bearing errors can be used to generate a sector shape. The intersection between this shape and the existing bounded location will produce a reduced bounded location.

  The contribution will also examine the bearing rate through the leg, to perform both straight-leg and alteration-leg detection, inserting new legs as appropriate (as described in Manoeuvre Detection Algorithm).

- *Assess candidate solution.*

  Given a solution, the contribution will calculate the target location at the time of each measured bearing. The error between each measured bearing and the bearing to the calculated target location will be determined, the weighting applied to it, and the value returned.

1.5.3. Speed analysis contribution

1.5.3.1. Attributes

This contribution contains no data.

1.5.3.2. Abilities

Constrain the problem space.

This contribution will pass through the series of bounded vehicle states, examine the bounded locations and calculate the speeds necessary to travel between the closest points on the bounds (min

speed) and furthest points on the bounds (max speed). The speed bounds will be further constrained to these values.

Also, the speed analysis contribution will examine the speed bounds for successive bounded vehicle states. If the speed bounds do not overlap, then the contribution will determine that the leg must actually contain an alteration - and replace it with an alteration leg. I don't think it's able to determine how to segment a straight leg into a composite - it can only replace it whole-meal.

1.5.4. Speed forecast contribution

1.5.4.1. Attributes

The analyst indicates that for a set period the vehicle must be travelling between 1 and 8 knots, and is probably travelling between 4 and 6 knots. He also gives a weighting to this forecast.

1.5.4.2. Abilities

- *Constrain the problem space.*

  For any bounded vehicle states between the specified time period the speeds are constrained (or further constrained) to being between 1 and 8 knots. The contribution will also generate an additional bounded vehicle state (if necessary) at the start and stop times. These bounded vehicle states will only contain the hard speed constraints.

- *Assess candidate solution.*

  For the current candidate solution, each vessel state within the specified time period is examined. A performance figure is generated according to if the candidate speed for that state is between 4 and 6 knots, or how far outside that range the speed is.

1.5.5. Straight leg forecast contribution

1.5.5.1. Attributes

The analyst specifies a time period for which he believes the vehicle is travelling on a straight course at constant speed.

1.5.5.2. Abilities

Constrain the problem space.

The contribution will examine the current set of legs. If there isn't a straight leg defined for the specified time period, then the a leg is inserted.

2. Optimisation Strategies

Of the two phased approach to SATC, the second page - consider candidate solutions has a much larger processing requirement. It's in that stage that high volumes of potential solutions have to be considered, with each point of each solution being used in many error/performance calculations,

The following three strategies were considered for phase 2.

2.1. Brute Force

The first strategy considered involved Brute Force. The algortihm steadily progressed through all possible permutations, exhaustively considering performance of each one, and finally producing an optimal result. This straightforward method was easy to implement, and proved useful in the early implementation phases. But, as we strove for greater fidelity in solution produced the grids of

candidate start/end points got tighter and tighter, giving an exponential explosion in the volume of processing to be conducted.

2.2. Simulated Annealing

After we had reached the limits of how much the Brute Force Algortihm could be performance optimised, we needed to consider optimisation technique in order to get *good enough* solutions in a reasonable time frame.

The first technique considered was Simulated Annealing. In this technique a function is produced that is able to produce a candidate solution. This function contains a temperature variable. At the start of processing this variable is high, which allows for a great variety of solutions considered. As it cools, it reduces the allowable changes in the solution.

This algorithm gave a significant performance improvement in generating solutions, and carried a lot of hope. But, it fell short in not giving the necessary degree of control when it came to ensuring that consecutive legs were coherent with each other, as explained below in Figure D.5.13, “Inconsistent range”.

2.3. Genetic Algortihm

Genetic Algortihms were the third optimisation technique considered. They brought the performance improvements of Simulated Annealing with the tight control offered by Brute Force processing. It is described further in the next section.

3. Solution Generator based on Genetic Algorithm

3.1. Definitions

Gene	single permutation of a straight leg track segment
Chromosome, Individual	set of genes, set of straight routes and constructed alterations between them (a composite route)
Population	set of chromosomes which are used on current iteration (a collection of composite routes).
GA	the collection of algorithms/processing that lead to the delivery of an optimal composite track solution
Island	one instance of separate GA with its own characteristics.
Iteration	the production and assessment of a single solution
Epoch	number of iterations conducted before a migrations is performed. (20 iterations by default in Debrief-SATC).

Example D.5.1. Example

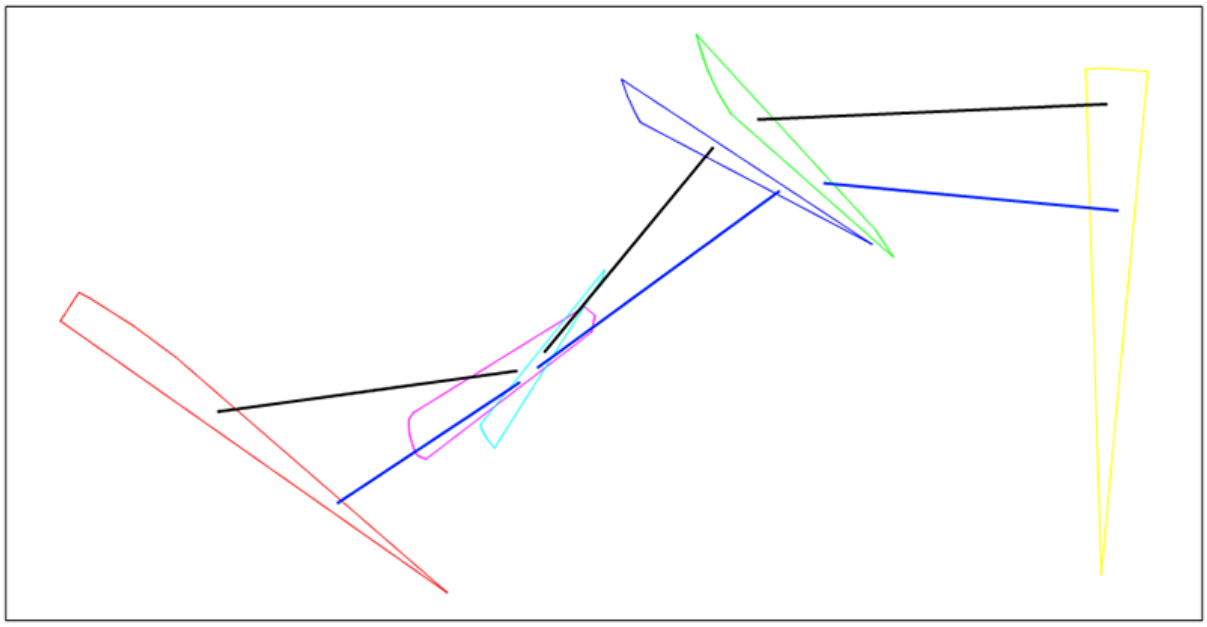
Gene of Chromosome 1 - any black straight line

Gene of Chromosome 2 - any blue straight line

Chromosome 1 - set of black straight lines

Chromosome 2 - set of blue straight lines

Figure D.5.3. Example



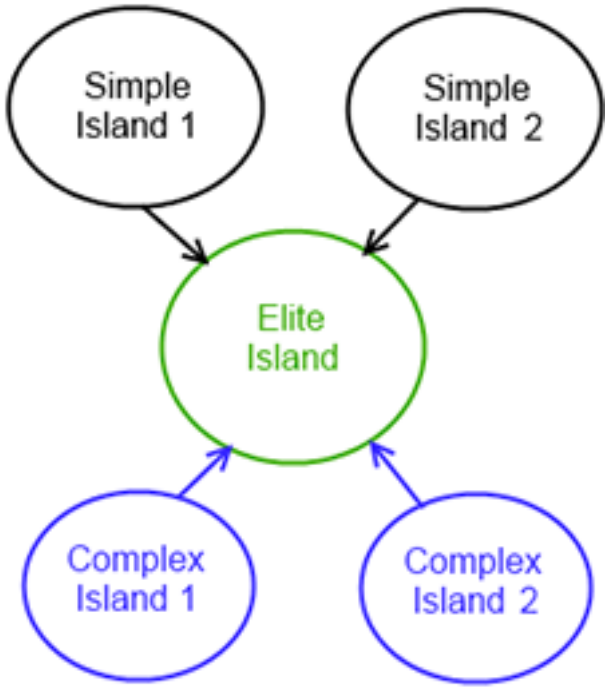
3.2. Structure

3.2.1. Islands

To allow multiple parallel threads of solution derivation, an Islands Model (Multiple-population GA) is used, with synchronized migrations (http://tracer.uc3m.es/tws/cEA/documents/cant98.pdf). This involves several separate populations which are optimized in parallel. After each epoch the islands exchange chromosomes between each other. Each island is separate GA with its own characteristics.

Our islands structure is shown on Figure D.5.4, “Islands structure”.

Figure D.5.4. Islands structure



The figure shows two simple and two complex islands. These are designed to produce good candidates from random search area. After each epoch every of simple and complex islands sends 5 elite individuals to elite island, which chooses the best candidate from these islands and optimizes it. The presence of different types of island allows the algorithm to produce an optimal solution across a range of problem types.

3.2.2. Genetic algorithm

Each island is one instance of GA with its own parameters. The different island types have different strategies for Selection, Crossover, and Mutation.

1. Candidate Generation - generates random chromosomes (see Section 3.3, “Candidate Factory”)

2. Selection Strategy - selects chromosomes from current population (see Section 3.4, “Chromosome selection strategy”)

3. Crossover - mates chromosomes selected by selection strategy and generates new chromosomes (see Section 3.5.1, “Crossover”)

4. Mutation - mutates some genes in chromosomes produced by crossover (see Section 3.5.2, “Mutation”)

5. Completion - the GA stops evolving after one of two criteria: time elapsed (typically set to 30 secs), or stagnation (where the optimal solution stops showing improvement in successive generations).

```
population = CandidateFactory.randomPopulation(populationSize)
while not finished(population)
   elite = take best from (population)
   newPopulation = SelectionStrategy.select(population)
   newPopulation = Crossover.mate(newPopulation)
   newPopulation = Mutation.mutate(newPopulation)
   add (elite) to (newPopulation)
   calculate fitness and sort (newPopulation)
   population = newPopulation
end
```

The Simple and Complex islands are configured as described in Section 3.7, “Island attributes”

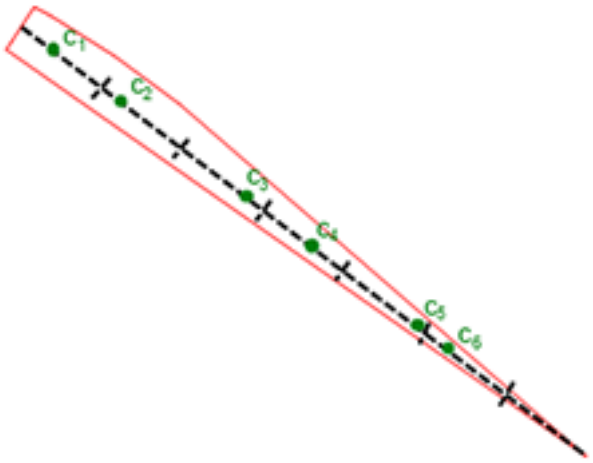
3.3. Candidate Factory

The Candidate Factory generates an initial set of candidate routes (individuals) within the problem space, together with further randomly generated routes.

No specific algorithm is used to generate initial population, it simply generates the required number of random solutions according to the following steps.

Usually start and end polygons contain part of corresponding bearing line. When present, the target bearing measurement is the most effective observation on the target track, so the best solutions start with initial points on this line. Where the polygon does contain part of bearing line this line splits on multiple segments and the candidate factory takes a random point from each segment consequentially, as shown on Figure D.5.5, “Bearing lines for new points”.

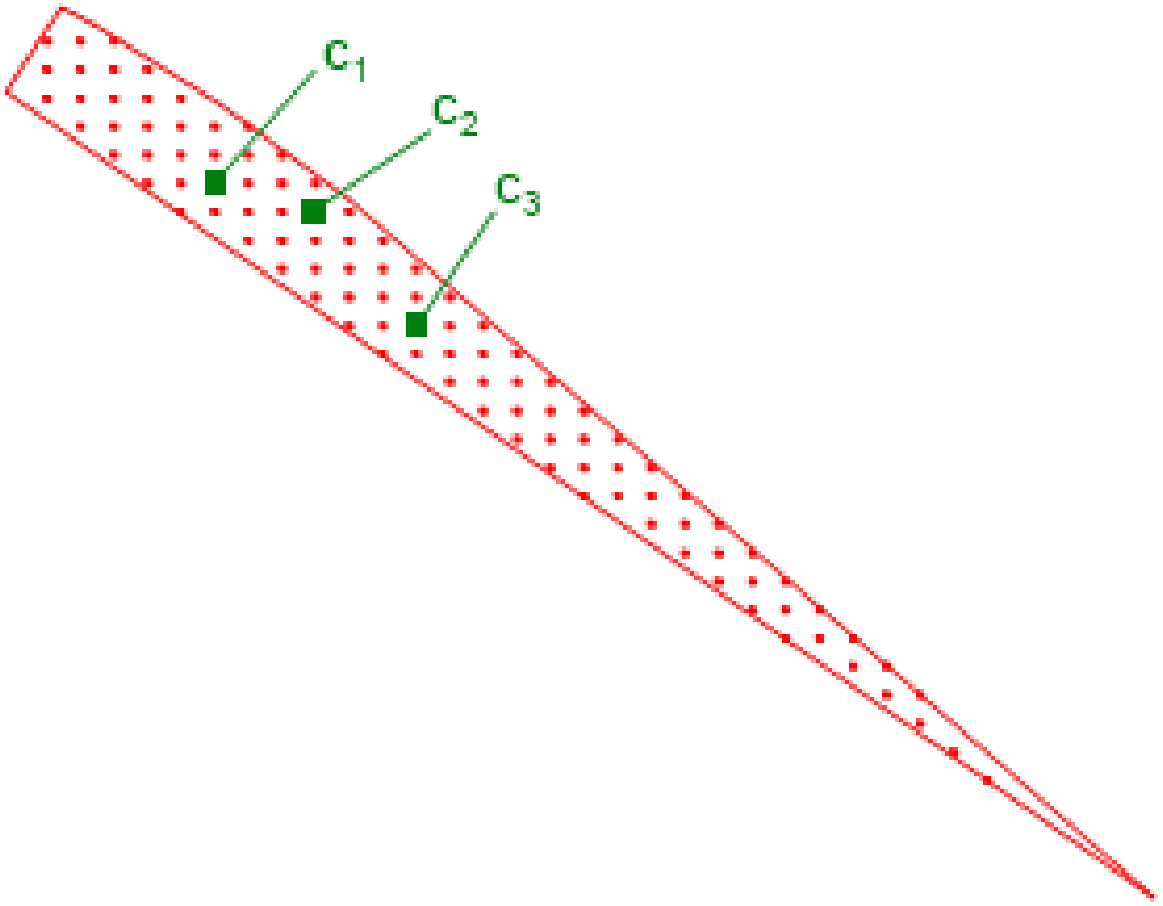
Figure D.5.5. Bearing lines for new points



When the candidate factory needs to generate a point for above polygon it takes first segment and generates c_1, when it needs a new point again it takes second segment and generates c_2 etc.

On occasions when the start or end polygon doesn't have part of corresponding bearing line (there is no sensor data for that time), the candidate factory generates a grid of points for this polygon, it applies a grid to the polygon then takes a random one each time, as in Figure D.5.6, “Polygon without bearing line”.

Figure D.5.6. Polygon without bearing line



To generate random chromosomes the candidate factory takes the start and end polygons for each straight leg and generates a point from each one using the above rules. The following pseudocode documents this:

```
I = empty chromosome
for (leg = every straight of Legs)
  S = start polygon of (leg)
  E = end polygon of (leg)
  s_a = take next point of (S)
  e_a = take next point of (E)
  a = route(s_a, e_a)
  add new gene (a) to chromosome (I)
end
```

3.4. Chromosome selection strategy

To select the chromosome to use in genetic operators we use tournament selection (http://en.wikipedia.org/wiki/Tournament_selection) with tournament size = 2 and a probability to allow the worse candidate, depending on the island type.

The tournament strategy is implemented according to this pseudocode:

```
population = current population
population = current population
p = probability to accept worse
selected = empty set of chromosomes
while (selected.size < populationSize)
  better = take random chromosome from (population)
  worse = take random chromosome from (population)
  if (better.score > worse.score)
    better <=> worse (swap them over)
  end
  if (use_worse_event(p))
    add new chromosome(worse) to (selected)
  else
    add new chromosome(better) to (selected)
  end
end
```

3.5. Genetic operators

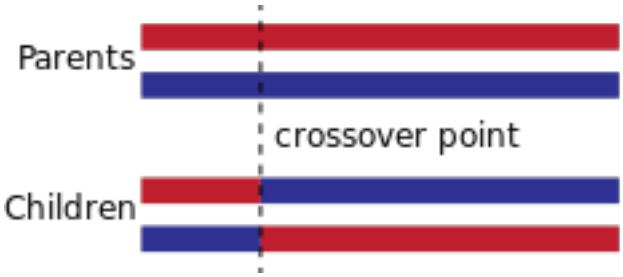
Two types of genetic operator are used: crossover and mutation.

3.5.1. Crossover

Crossover is genetic operator which is used to produce new chromosomes based on crossing two parent chromosomes. There are several techniques to implement crossover, we use two of them: one-point list crossover and arithmetic crossover.

3.5.1.1. One-point list crossover

One-point list crossover is well known and widely used crossover operator http://en.wikipedia.org/wiki/Crossover_(genetic_algorithm) [http://en.wikipedia.org/wiki/Crossover_%28genetic_algorithm%29]. It selects random split point in parent chromosomes and generates two child chromosome based on the following rule:



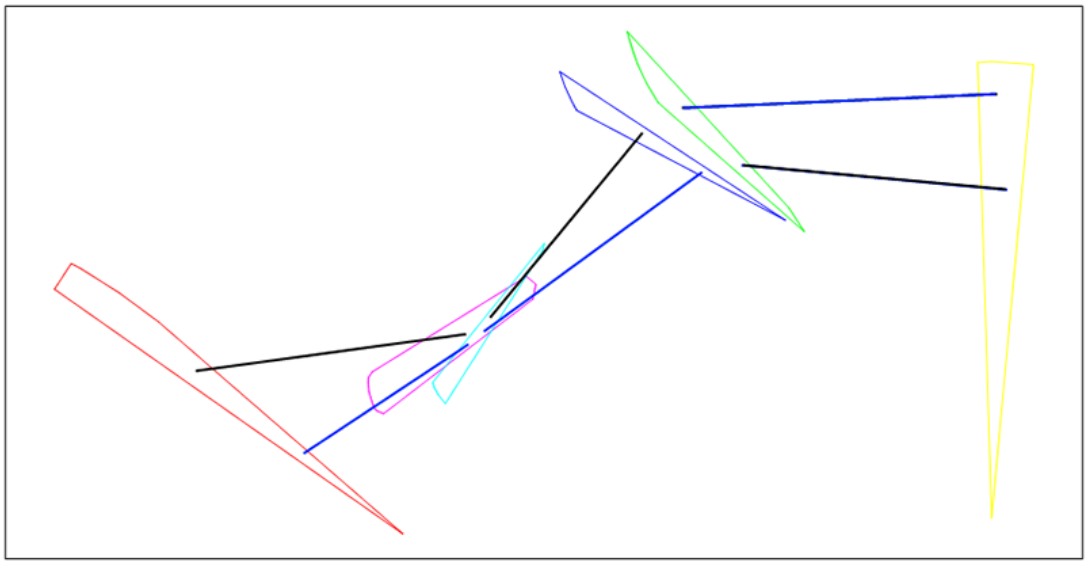
In our example (Figure D.5.3, “Example”): our two chromosomes

$X_1 = \{ a_1, b_1, c_1\}$ - black straight lines

$X_2 = \{ a_2, b_2, c_2\}$ - blue straight lines

let's take our split point = 1, in this case we will have two new chromosomes:

Figure D.5.7. one point crossover example



$I_1 = \{ a_2, b_1, c_1\}$ - black lines on the above figure

$I_2 = \{ a_1, b_2, c_2\}$ - blue lines on the above figure

3.5.1.2. Arithmetic crossover

Arithmetic crossover is GA crossover operator designed for continuous search spaces, it has several modification and described in many papers (for instance: http://www.researchgate.net/publication/228618503_A_new_genetic_algorithm_with_arithmetic_crossover_to_economic_and_environmental_econ file/9fcfd50a6971a00cc5.pdf)

The general idea of arithmetic crossover is to take two corresponding genes from parent individuals and create new gene as linear combination of parent genes:

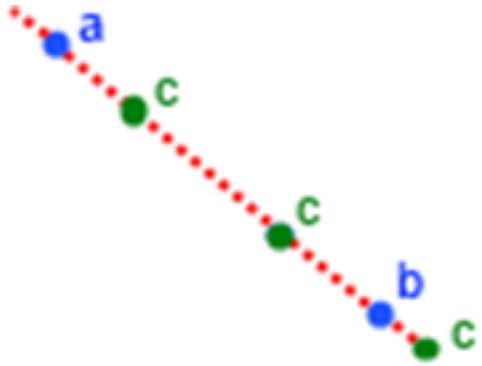
$$c = a \cdot r + b \cdot (1 - r)$$ where

- a - gene from first parent
- b - gene from second parent
- c - new gene
- r - random number [0; 1]

Because the above formula restricts a new gene to be between its parents, several authors propose extending the random number interval to [-d; 1 + d], where d is parameter which allows new gene to go outside. Usually d is less than 0.25.

For 2D points arithmetic crossover looks like: a, b - (blue) parent genes, c - (green) possible new genes:

Figure D.5.8. Illustration of arithmetic crossover for 2D points



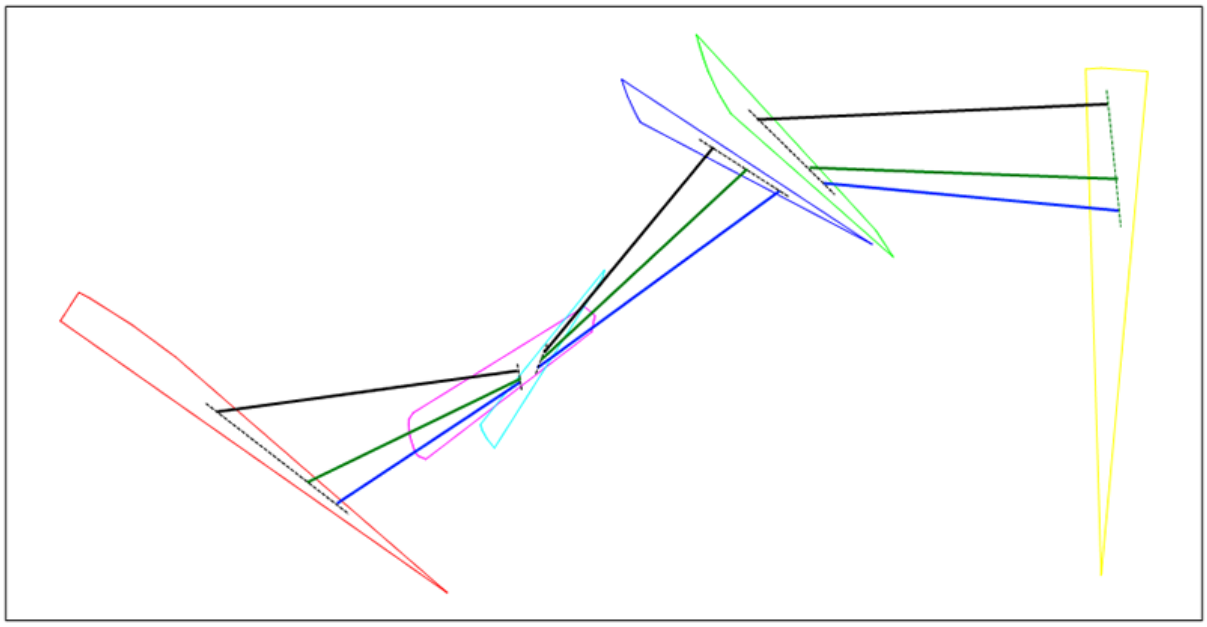
In our application we will use the following implementation of arithmetic crossover. Initial parameters: two parent chromosomes P_1, P_2.

Pseudocode:

```
I = empty chromosome

for (a = every gene of X1) (b = every gene of X2)
  sa = take start point of a
  sb = take start point of b
  ea = take end point of a
  eb = take end point of b

  // generate random from [-0.1; 1.1)
  // this allows to create new individuals
  // which aren't in parent's bounds sometimes
  r = 1.2 * random() - 0.1
  // new start point on road from sa to sb
  sc = r * sa + (1 - r) * sb
  // new end point on road from ea to eb
  ec = r * ea + (1 - r) * eb
  c = route(sc, ec)
  add new gene (c) to chromosome (I)
end
```

For our example from Figure D.5.3, “Example”, we generate new chromosome (green) with arithmetic crossover.

Figure D.5.9. Arithmetic crossover example



3.5.2. Mutation

The goal of mutation is to produce individuals which aren't present in current population - thus individuals that aren't achievable by crossover. Mutation is a genetic operator which produces new chromosomes by substituting genes in parent chromosome for newly generated ones with some specified probabilit. The specific mutation used in a GA instance depends on an understanding of the problem domain and data patterns.In our implementation we will use two techniques.

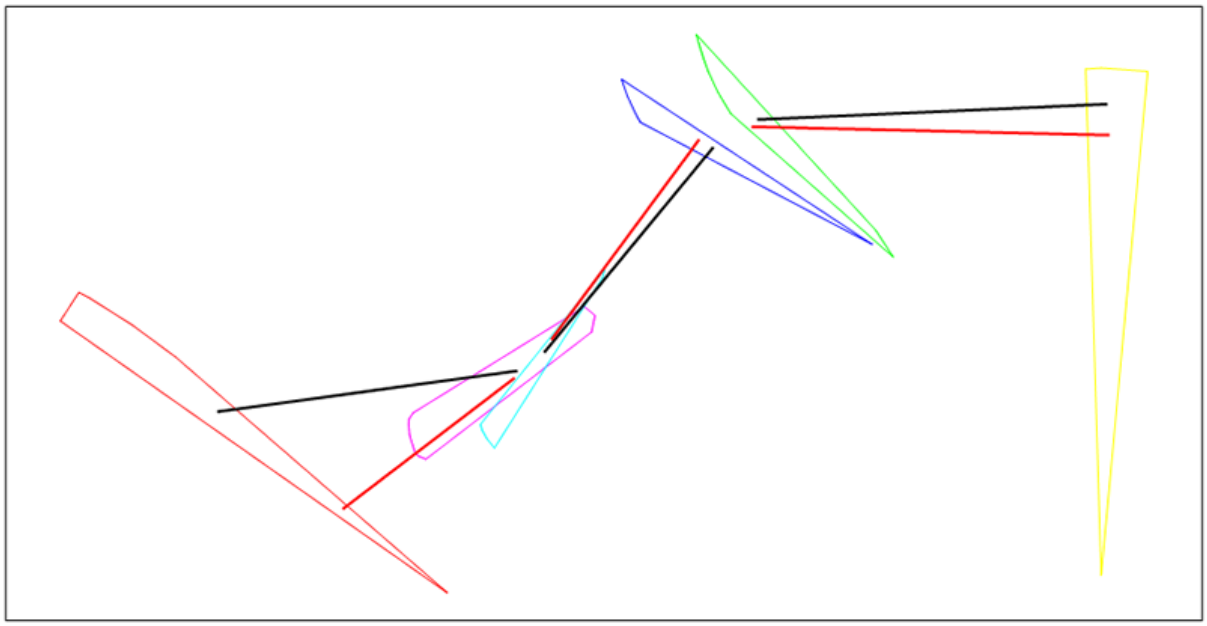
3.5.2.1. Random mutation

Random mutation substitutes some genes of a parent chromosome for corresponding genes randomly generated by candidates factory chromosome. Pseudocode of random mutation looks like:

```
X = parent chromosome
R = new random chromosome (from CandidatesFactory.generateRandom)
p = mutation probability
I = empty chromosome
for (a every gene of X) (b every gene of R)
  if (mutation_event(p))
    add new gene (b) to chromosome (I)
  else
    add new gene (a) to chromosome (I)
  end
end
```

Example: Let's take parent chromosome X (black) and random chromosome R (red). Mutation is made only on second gene this will produce new chromosome I (green)

Figure D.5.10. Random mutation



3.5.2.2. Mutation to vertex

Mutation to vertex comes to GA from Debrief's experimental Simulated Annealing (SA) optimized point generation implementation (see Section 3.5.4, “Optimised point generation”), this algorithm gives good results for SA optimization and there is value in also using it within GA with some specific modifications.

Because in GA on each iteration there are multiple individuals there is no reason to use two vertices and take a new point between them. It's simpler to take only one vertex and create a new point as linear combination of current point and chosen vertex:

$$c = v \cdot r + a \cdot (1 - r)$$ where

- a - current point
- b - chosen vertex
- c - new point
- r - random number between [0, 1]

To generate "r” we use Y distribution from SA implementation (see Section 3.5.3, “Standard VFA strategy”). This distribution depends on current iteration and produces values which are closest to 0 when number of iterations goes to infinity. To avoid very small values for "r” we take iteration parameter for this distribution as follows:

iteration = real iteration % 300

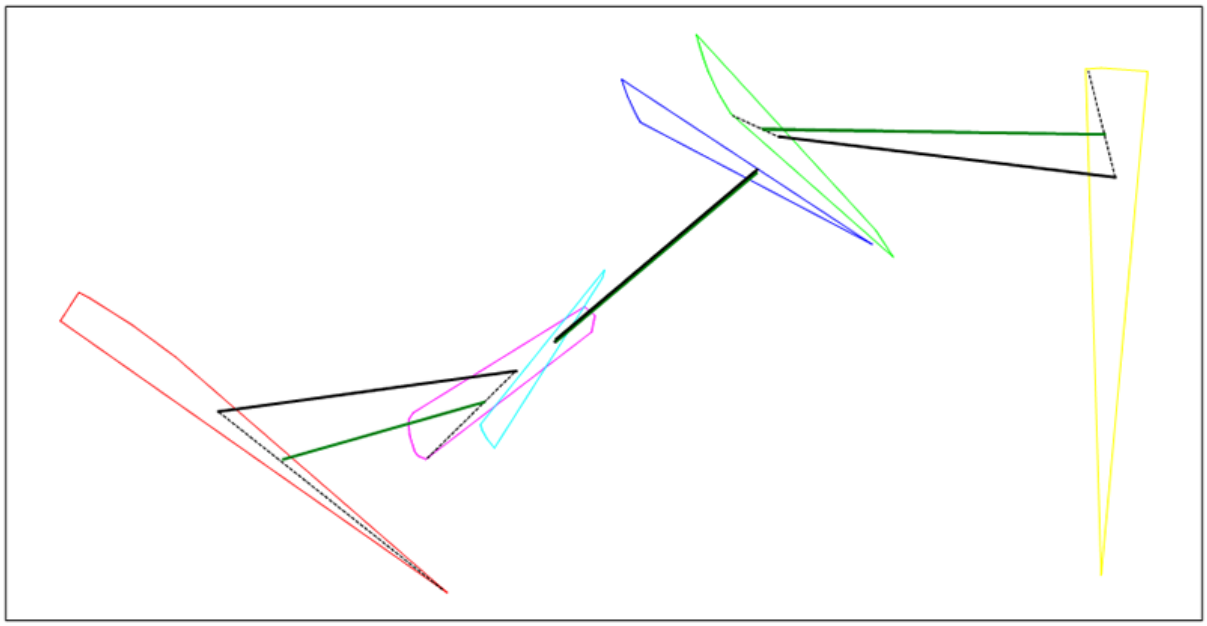
Pseudocode for mutation to vertex:

```
X - parent chromosome
p - mutation probability

I = empty chromosome
for (a every gene of X)
  if (mutation_event(p))
    s_a = start point of a
    e_a = end point of a
```

```
    S = find start polygon for (a)
    E = find end polygon for (a)
    r = y_random(iteration % 300)
    s_c = vertex(S) * r + (1-r) * s_a
    e_c = vertex(E) * r + (1-r) * e_a
    c = route(s_c, s_e)
    add new gene (c) to chromosome (I)
  else
    add new gene (a) to chromosome (I)
  end
end
```

Example: Parent chromosome X (black), new chromosome I (green). Mutation was made for first and third genes.

Figure D.5.11. Mutation to vertex example



3.5.3. Standard VFA strategy

The development of the semi-automated TMA in Debrief included some investigation into the value in Simulated Annealing (SA). In support of this, algorithms were developed that related to the Very Fast Algortihm (VFA) strategy for producing an improved SA temperature function.

The initial algorithm is an implementation on VFA selection algorithm with current point (P) and current temperature (T):

1. VFA defines random distribution (Y):

$$y = \mathrm{sgn}(u - \frac{1}{2})T\,[(1 + 1/T)^{|2u-1|} - 1]$$

   T - current temperature

   u - random value of uniform distribution

2. Calculate two random values y_1 and y_2 based on this Y distribution

3. Create new point: P_{new} = (P.x + y_1 * width, P.y + y_2 * height)

4. If current polygon doesn't contain P_{new} point go to step 2.

3.5.4. Optimised point generation

The VFA selection strategy described above has a high performance cost when run later in the process - a cost that isn't justified when the temperature is cooling, and only small steps are needed (when SA is in the "discrete improvements" phase as temperature approaches zero and only small jumps are allowed). So, a custom selection strategy is provided, which selects a random point in area with current point (P) and current temperature (T):

1. Select two random vertices of current polygon (V_1, V_2)

2. Create two segments: (P, V_1), (P, V_2)

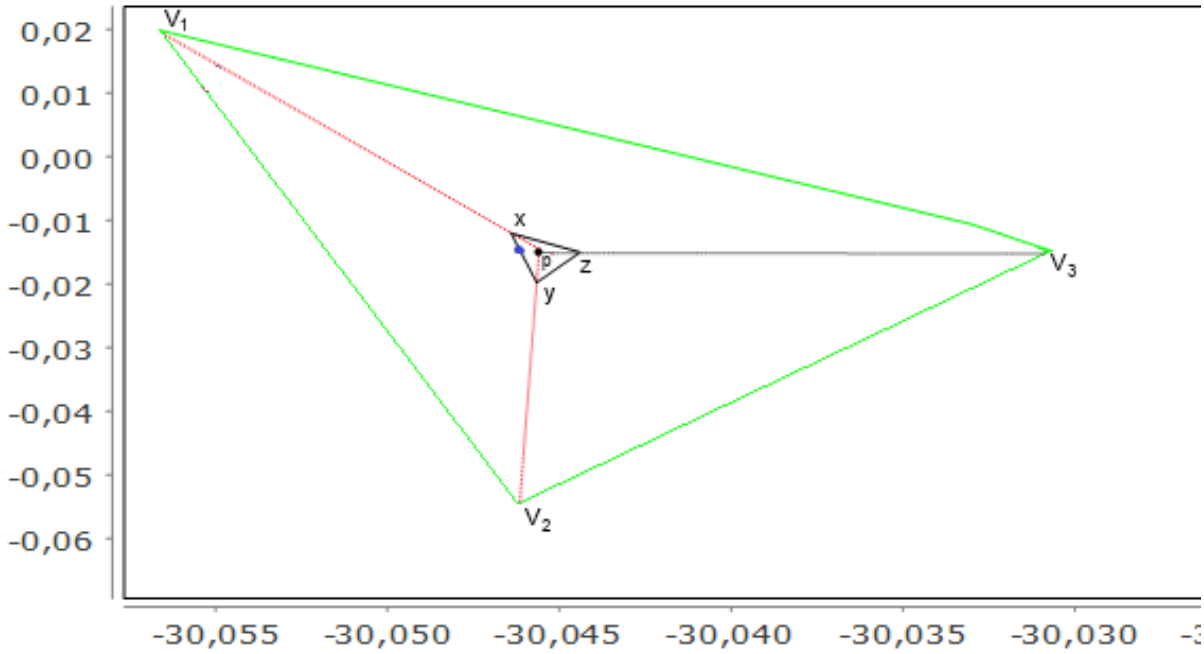
3. Calculate two random values y_1 and y_2 based on Y distribution from VFA (VFA - 1)

4. find X point on (P, V_1) segment on road from P to V_1, with distance d_1 = abs(y_1) * distance(P, V_1)

5. find Y point on (P, P_2) segment on road from P to V_2, with distance d_2 = abs(y_2) * distance(P, V_2)

6. find P_{new} on road from X to Y with distance = rand(0, 1) * distance(X, Y)

Figure D.5.12. Point generation



3.6. Fitness function

As previously explained, each chromosome in GA is a set of straight routes. The fitness function starts by verifying that each of these straight routes are achievable. If everything is ok it then constructs altering routes (cubic bezier curves) between them. The GA fitness score is the sum of two scores:

- a) contributions scores for straight and altering routes
- b) score how altering route is compatible with its previous and next straight routes.

The detail of the a) scores is described in the Debrief-SATC contribution documentation.

Score b) is very effective in producing a solution that is the sum of consistent straight line legs. It's quite easy for the GA to produce a solution that is the collection of the best performing individual legs, but where the legs aren't consistent with each other. In Figure D.5.13, “Inconsistent range” it is clear that whilst the green and purple legs are both valid, they aren't consistent with each other.

Figure D.5.13. Inconsistent range

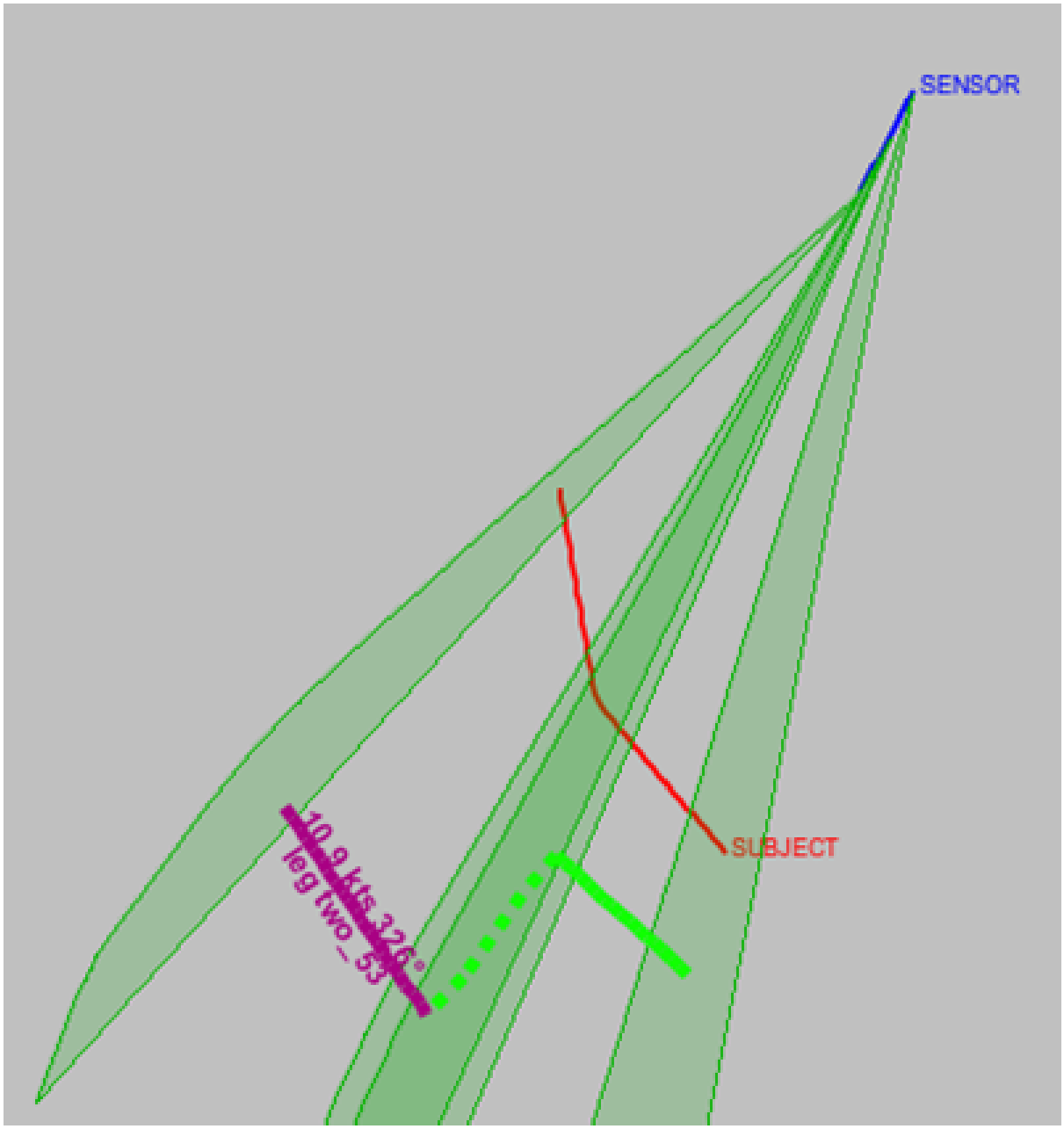
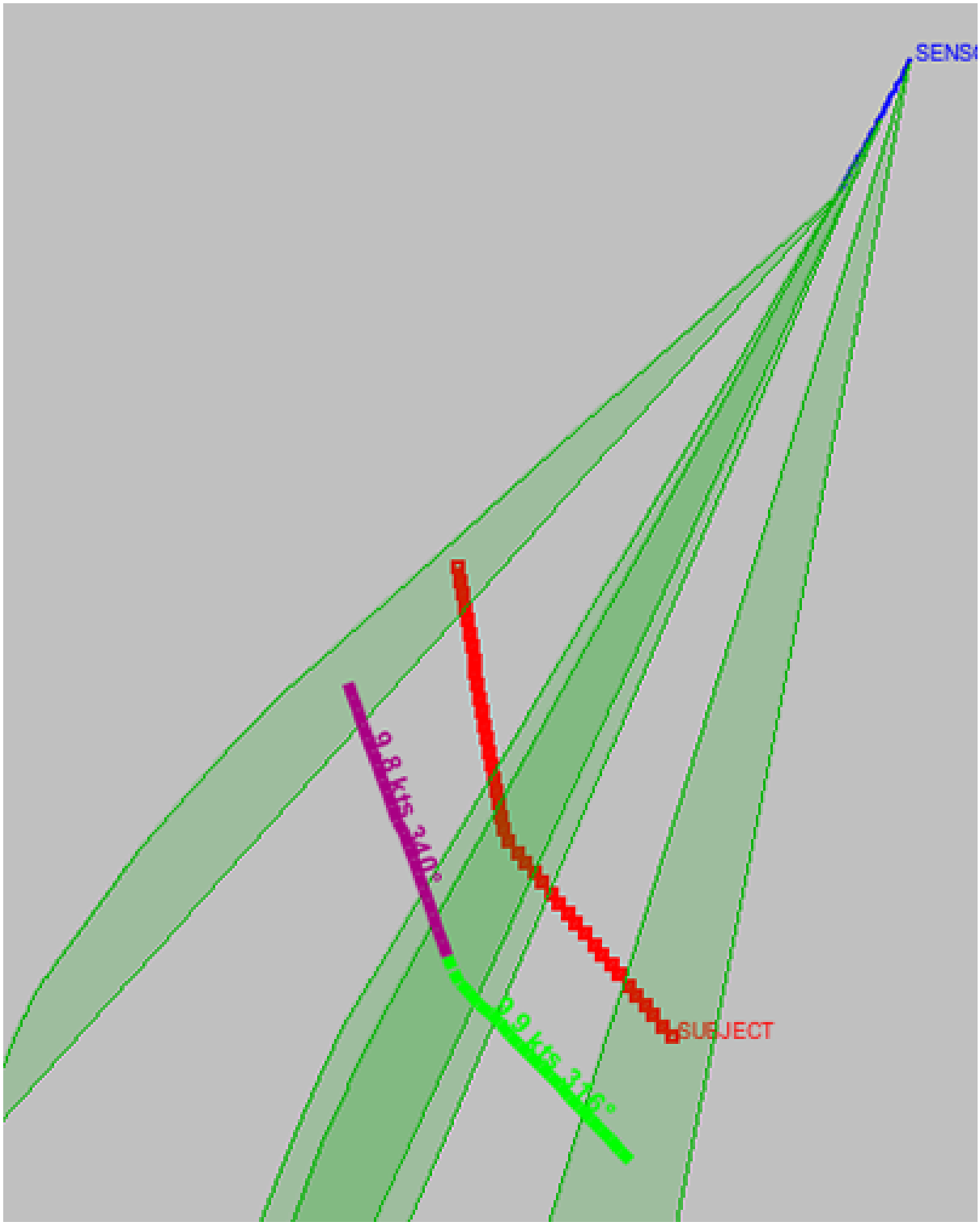


Figure D.5.14, “Consistent legs” demonstrates the inclusion of the consistent legs scoring - used to verify that it is possible for a vehicle to transition between the two legs in the available time.

Figure D.5.14. Consistent legs



Note that in Figure D.5.14, “Consistent legs”, the solutions still don't match the true ("SUBJECT") track - that requires further contribution from the analyst. But, the legs in the solution route are consistent with each other.

To score how well altering route is compatible with its previous and next straight routes the GA fitness function uses speed changes during alteration.

A smooth speed plot (without peaks) is the best situation (Figure D.5.15, “Speed plot with smooth speed change”), peaks which are near to previous and next speed are worse but valid (Figure D.5.16,

“Speed plot with peaks near to straight legs”), and huge peaks are the worst situation (Figure D.5.17, “Speed plot with huge peaks”)

Figure D.5.15. Speed plot with smooth speed change

Figure D.5.16. Speed plot with peaks near to straight legs

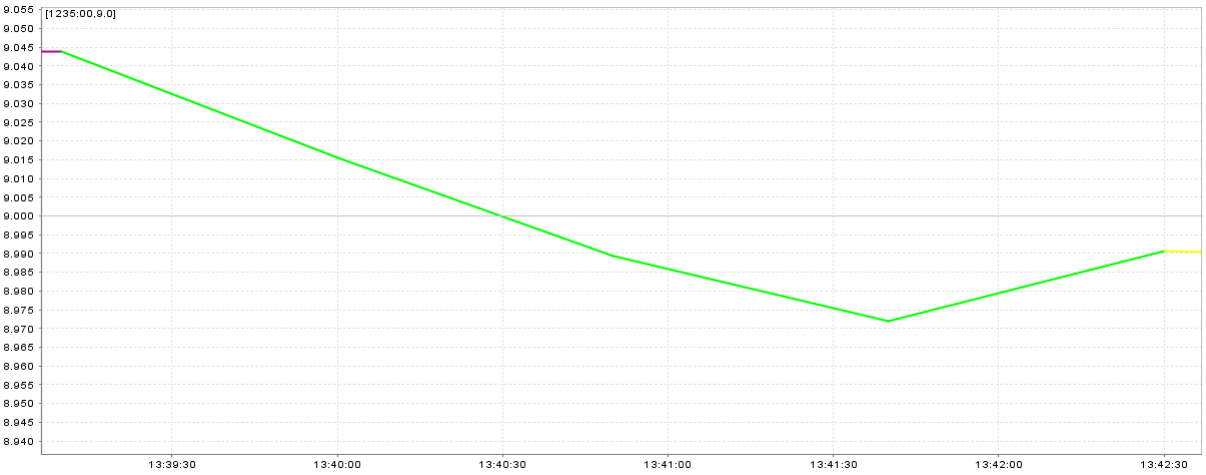


Figure D.5.17. Speed plot with huge peaks

Pseudocode for altering route score is:

```
a - altering route
previous - previous straight route
next - next straight route

peaks = a.speed_peaks_count
score = 0;
if (peaks == 0)
  score = 0;
else if (peaks == 1)
  score = (a.speed_peak - closest(previous.speed, next.speed))^2
else
  score = 1.5 * (a.max_speed_peak - a. min_speed_peak)^2
end
```

Pseudocode of fitness function:

```
X - chromosome to score
score = 0;
if (some route from (X) is impossible)
  score = MAX_SCORE;
  return;
end
alterings = generate alterings for (X)
for (every straight route (s) from X)
  score = score + calculate contributions score for (s)
end
for (every altering route (a) from (alterings))
  score = score + calculate contributions score for (a)
  score = score + compatible score (a, previous of(a), next of (a))
end
```

3.7. Island attributes

3.7.1. Elite island characteristics (beta)

- Fixed population size = 70 individuals by default
- Fixed chromosome size = count of straight legs
- Elitism = 10 individuals by default
- Tournament selection (probability to select worse = 0)
- Use altering legs
- Crossovers:
  - one-point crossover (20% of selected candidates)
  - adaptive arithmetic crossover (80% of selected candidates)
- Mutations:
  - mutation to vertex with 0.25 probability

3.7.2. Simple and complex island characteristics

The only difference between simple and complex island is simple islands use only straight legs and DON'T use altering ones, and complex islands use altering legs too.

- Fixed population size = 70 individuals by default

- Fixed chromosome size = count of straight legs
- Elitism = 10 individuals by default
- Tournament selection (probability to select worse = 0.3)
- Crossovers:
  - non-adaptive arithmetic crossover
- Mutations:
  - mutation to vertex with 0.4 probability (60% of selected candidates)
  - random mutation with 0.4 probability (40% of selected candidates)

Chapter D.6. System Documentation for DIS integration

1. Introduction

Debrief is able to integrate with network simulators that adhere to the DIS protocol. Debrief is able to receive a set of DIS messages (see Section 3, “Messages supported”). Debrief is also able to send StartResume and StopFreeze messages along the link. User support is provided both in the form of a Debrief tutorial, and here in the user guide (see Chapter B.12, *Support for DIS Protocol*).

1.1. DIS Standard

Debrief supports version 6 of the DIS standard (Version 6 - IEEE 1278.1a-1998 (amendment to IEEE 1278.1-1995)), as implemented by the Open Source OpenDis project: `https://github.com/open-dis/open-dis-java`.

In practice, the version of OpenDis used will be updated to the most recent release each time the code in Debrief' DIS plugin receives maintenance attention.

1.2. Adding support for further DIS message types

The following steps should be taken to support new DIS message types:

1. Introduce a new listener interface for the new message (similar to `IDISFireListener`)
2. Extend `DISModule` to allow listeners for the message to be managed
3. Extend `DISModule` to introduce a new handler method for this message (similar to `handleFire`)
4. Extend the `logPDU` method of `DISModule` listen for this message, and pass it to the handler
5. Introduce a new file listener for the event (like `FireFileListener`), and add it to `HeadlessDISLogger`
6. Introduce a new Debrief listener for the event (like `DebriefFireListener`), and add it to `DISListenerView`
7. Extend the list of messages supported (Section 3, “Messages supported”

2. Command line options

Some organisations may wish to specify the DIS IP Address & Port values in the command line when starting Debrief. This allows a parent process/simulator to start Debrief preconfigured with the correcft network parameters.

Set these values by using these settings on the command line: `-Dport=2345 -DipAddress=23.44.55.66`.

3. Messages supported

Table D.6.1. DIS Messages Supported

Message	Fields	Representation in Debrief
Entity State	• Entity Id • Location	Retrieve the vessel name from the `Launch` message lookup table, create a track for this fix (if necessary), and add this fix to that track

Message	Fields	Representation in Debrief
	• Orientation • Velocity	
Collision	• Issuing Entity • Colliding Entity	Create a narrative entry for this event, assigned against the Issuing Entity
Detonation	• Entity Id • Location	Create both a narrative entry for this event, assigned against the Issuing Entity. Also create a text label at the relevant location, placed in the `Detonations` layer
Fire	• Firing Entity Id • Target Entity Id	Create both a narrative entry for this event, assigned against the Issuing Entity. Also create a text label at the relevant location, placed in the `WPN Release` layer
Event Report	• Originator Id • Event Type • Variable Datums	The Event Report PDU is used as a handler for textual content across a range of simulation events. See the list of event report types below (Table D.6.2, “DIS Event Report messages”)
Start / Resume	• Time	Update the Debrief DIS Listener View to acknowledge that the simulator is running (e.g. enable the Pause and Stop buttons
Stop / Freeze	• Time • Reason	Update the Debrief DIS Listener View to acknowledge that the simulator has either stopped or paused (by inspecing the value of the `Reason` field)

Table D.6.2. DIS Event Report messages

Type	Event Type	Sample	Parsing	Representation in Debrief
Comms	10001	muteTime=0 txFreq=1804	Verbatim	Narrative entry
Launch	10002	`NAME:FF23`	Use colon as delimiter	Collate lookup table, to translate EntityId to entity name
New TMA Track	10003	DETECTION E3-3	E3-3	Collate lookup table, to translate EntityId to entity name
Tactics Change	10004	`XXXX from YYYY`	Verbatim	Narrative entry
New TMA Track (where subject is OSAT)	10005	DETECTION E3-2	E3-2	Collate lookup table, to translate EntityId to entity name

4. Other DIS specifics

Kind — A value of 1 denotes Platform, 2 denotes Munitions (Torpedo symbol), 8 for countermeasures, 101 the TMA track of interest (OSAT) and 102 other TMA tracks. Periods where a TMA track is denoted OSAT have fixes highlighted in a lighter shade (orange)

Domain — 0 for Other (sonar buoys), 3 for Surface (Frigate symbol), 4 for Sub-surface (submarine symbol), 6 for Torpedo

Force	0 for Other (Orange), 1 for Friendly (Blue), 2 for Opposing (Red), 3 for Neutral (Green)
Stop Reason	1 for Freeze (Pause), 2 for Stop (all complete), 7 for Iteration complete
Stop PDU Request ID	The total number of iterations performed
Start PDU Request ID	The id number of this iteration

Appendix I. Debrief Glossary

1. Introduction

This section contains explanations of terms used within Debrief

Glossary

AUTEC

Atlantic Underwater Test Facility. Data files from this facility which are to be imported into Debrief should be suffixed with "RAO". The origin of AUTEC is:

```
23º 26' 37.6280" N
77º 38' 6.8250" W
```

AFWTF

Atlantic Fleet Weapons Training Facility. Data files from this facility which are to be imported into Debrief should be suffixed with "PRN". The origin of AFWTF is :

```
17º 38.1577' N
065º 4.2065' W
```

Annotation

An annotation is the generic term used to describe the graphic elements added to a plot which do not represent vehicle positions, or bearings recorded on vehicle-mounted sensors. Examples of annotations are rectangles, ellipses and lines.

ASSET

The Advanced Scenario Simulator for the Evaluation of Tactics, a modular simulation suite intended for high-level simulation of maritime tactical scenarios by a relatively inexperienced user (read: uniformed). ASSET was actually the predecessor to Debrief, with the initial Debrief software being created to analyse ASSET simulation results. Quickly it was recognised that Debrief could also be usefully employed in the analysis of real exercise tracks. Aaah, how close we came to not having Debrief at all...

Bearing rate

Bearing rate within the application is calculated using the following formula:

```
Bdot = ((Tspd * sin(Tcrse) - Ospd * sin(Ocrse)) *
cos(brg) - (Tspd * cos(Tcrse) - Ospd * cos(Ocrse))
 *
sin(brg)) / range * 60
```

```
Rdot = Range Rate (yds/min)
Bdot = Bearing Rate (degs/min)
Tspd, Tcrse = Target course and speed (degs & yps)
```

```
Ospd, Ocrse = Ownship course and speed (degs & yps)
brg = Bearing to target from ownship (degs)
range = Range to target from ownship (yds)
```

Positive and negative bearing rates are named Right and Left according to naval convention, abbreviated to R and L in the tote.

Bookmarks

The combination of a DTG, a remark and the name of a plot-file. Debrief NG presents the series of bookmarks allowing you to quickly move through events of interest across a series of plot-files.

Buoyfields

A series of sonar buoys which are laid in a particular pattern during Anti-Submarine Warfare.

Build date

Each copy of Debrief is aware of the date it was built. Find this out by selecting About from the Help menu.

Tutorials

Tutorials guide users through tasks. The task is broken down into steps and presented to the user one step at a time, and the user checks off the steps as he/she completes them. The Cheat Sheets can be accessed from the command in Debrief's Help menu, or the Tutorials section of the Debrief's Welcome page.

CMAP

Core Maritime Analysis Platform a framework of components intended to be reused across a range of maritime analysis applications. The two initial CMAP applications are Debrief and *ASSET*.

Coastline file

Debrief expects to find a coastline file (named World.dat) in its installation directory. Debrief loads this file in the background as soon as it opens; regardless of whether the user has requested to add a coastline to the current plot. Once the coastline is loaded (for the standard 1.2Mb file this takes around 8 seconds) there is no further performance penalty within the application.

The coastline file should be formatted in the following way:

- The coastline consists of a series of coastline segments. Each segment is drawn as a continuous polygon by the application.
- Each segment begins with the # -b separator on a line of its own
- Then there are a series of lines each containing a point in latitude and longitude expressed in decimal degrees (to 6 decimal places in the standard file).

```
# -b
-7.491098    4.257159
-7.523953    4.245425
-9.112761    5.008146
-9.464786    5.339050
-9.807424    5.681688
# -b
-9.807424    5.681688
-10.004558    5.845966
-11.152161    6.606341
-11.131039    6.639197
-11.163895    6.672052
```

```
-11.307052    6.761232
-11.351642    6.803475
```

Dead reckoning	Dead reckoning (DR) is the process of estimating one's current position based upon a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time, and course. Debrief plots tracks in DR mode by plotting the positions as a series of course/speed legs from an origin. Debrief plots lats/longs unchanged when in *Over the Ground* mode.
DIS Standard	Distributed Interactive Simulation (DIS) is an IEEE standard for conducting real-time platform-level wargaming across multiple host computers and is used worldwide, especially by military organizations but also by other agencies such as those involved in space exploration and medicine.
Display mode	Each of the display modes used on the Debrief plot represents a different method of stepping through the plot. When in the normal mode each track is shown in full, with a highlight drawn over the current point (as indicated in the Tote). When in snail mode only the current point plus an optional back-track is plotted.
DTG	Acronym representing Date Time Group
Earth Model	The Earth Model used by the application is modular and interchangeable. In the initial release of the application the calculations use the Rhumb-Line Formulae, as described in 'Admiralty Manual of Navigation, Volume 2, 1973'. Short-distance sailing is defined as "the following of a rhumb-line track for a distance not greater than 600"
ETOPO	Standard for digital topographic data, predominantly distributed by the NOAA
GitHub	GitHub.com [http://github.com] is a web-site providing a range of software development capabilities. Its name comes from its use of the *Git* software version control system. *Git* allows multiple developers to work simultaneously on a project, preventing conflicts and helping to merge divergent development paths.
Highlighter	The highlighter is the graphic used to indicate the current point on a track. Use of different highlighters allows range rings or a vessel-specific symbol to be plotted at the current point
Layer	A collection of objected plotted on the Debrief Plot. Each layer can be switched on and off individually using the Outline View (see Section 4, “Layer management”). When written to a plot-file, each layer is stored separately - making it quite easy to copy individual layers out of a plot-file using a text-editor and collating them into a new file. (An example of this would be drawing exercise areas into one session, then moving them all into one layer, save the file to disk, copy this layer to a file of its own, then dropping this file of exercise areas into new files - an example of this is in the VPF best-practice, Section 4.3, “How to do it - 1”)
Lightweight Tracks	A simpler, faster version of a conventional track. These are quicker to render in Debrief and can be placed into Layers - unlike normal tracks, which all sit at the top level. Fewer formatting

	and manipulation operations are available for lightweight tracks, particularly those relating to TMA.
MWC	[Extracts taken from MWC 2000 Flyer] Under CinCFleet, the Maritime Warfare Centre (MWC) is a 'one-stop shop' for the evolution and dissemination of maritime/joint doctrine and concepts through teaching, tactical development, operational analysis, force development and wargaming. It also provides the focus for the development and practice of operational level warfighting, planning and decision making. The MWC was formed on 1 October 1995 merging the activities of the Maritime Warfare Development Centre at HMS Dolphin and the Maritime Tactical School at HMS Dryad to create a focal point for doctrine and tactical development.
Narrative	A series of time-related text messages. Typically these may represent the narrative recorded in a control room during an exercise, but alternatively they may contain a series of status messages retrieved from a sensor or weapon. All that is required is that the message have a DTG attached and that it may be represented in text form.
Natural Earth	In 2015 support for Natural Earth data was added to Debrief. Natural Earth is a public domain, freely available dataset of Vector and Raster data. To support Debrief analysts a customized version of Natural Earth has been produced that is compliant with the Mercator Projection, and styled suitably for the maritime domain.
Over the ground	Plotting a Debrief track using the recorded sensor positions. Whilst Debrief stores course and speed data, and uses their values in calculations, they do not contribute to how the positions are plotted. Debrief plots a track using course and speed data when in *Dead Reckoning* mode.
Overview	A zoomed out plot showing the full dataset currently loaded. Double clicking on this plot forces the main plot to re-centre on the selected point, and dragging an area on this plot forces the main plot to zoom in on the selected area. The formatting on the overview chart is identical to (and unchangeable from) that on the main plot, with the exception that text is not plotted - to reduce clutter.
Perspective	Each Workbench window contains one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. For example, the Tactical Analysis perspective combines views that you would commonly use while editing analysing tactical files, while the Contact Ork perspective would help aliens (typically named Mork) in contacting their mother planet - together with the VOIP software link direct to Orson. As you work in the Workbench, you will probably switch perspectives frequently, especially if you're having Mindy trouble. Perspectives control what appears in certain menus and toolbars. They define visible action sets, which you can change to customize a perspective. You can save a perspective that you build in this

manner, making your own custom perspective that you can open again later.

You can use the General > Perspectives preference page to open perspectives in the same window or in a new window.

PFWTF

Pacific Fleet Weapons Training Facility (see AFWTF). The origin of PFWTF is :

```
22º 7.16646' N
159º 55.17' W
```

Plot

A graphic *God's-eye* representation of the current dataset. By default the plot has a black background but this may be altered by the user.

Plot-File

A file containing the following:

- The data originally loaded from the Replay file
- Any formatting applied to the data originally loaded
- The details of any features added to the plot from the toolbars
- The coordinates of the current view of the data
- The settings of any controls used in Debrief (time on the Tote, primary/secondary tracks, etc)

Plot-Files end with an `DPF` suffix and may be viewed in Internet Explorer or edited using Notepad.

Properties window

A view containing a list of all of the editable properties for an object within Debrief. Where applicable, custom editors are supplied (Color, Location, DTG, etc)

Range rate

Range rate within the application is calculated using the following formula:

```
Rdot = (Tspd * cos (Tcrse - brg) -
Ospd * cos(Ocrse - brg) /*60
```

```
              Rdot = Range Rate (yds/min)
Tspd, Tcrse = Target course and speed (degs & yps)
Ospd, Ocrse = Ownship course and speed (degs & yps)
brg = Bearing to target from ownship (degs)
```

Replay

Replay is the name of the Unix application used for viewing tracks back in the early 90's at MWC

Serial

A "block" of exercise time, typically a sub-section of an exercise. An analyst will normally analyse and exercise one serial at a time, and it is usual for the exercise data to be broken down into serials.

Sensor

A sensor is defined as source of bearing-related information. As such, it could clearly be an acoustic sensor such as a sonar, but

	could also be a periscope or radar. Debrief makes no specific assumptions regarding what type of sensor is being represented.
Sensor contact	This is an individual contact recorded on a sensor, a single bearing line reaching from the sensor location (origin) along the contact bearing to the contact range.
Session	The layers, projection details, and settings of any GUI-elements for the current view
Slant range	The line of sight distance between two points, not at the same level relative to a specific datum. Normally in Debrief range is just calculated in two-dimensions (at the surface), but slant ranges can be requested via the Preferences window. Slant ranges are particularly useful in Debrief when analysing the proximity of two entities that are very close when measured at the surface but who possess a significant depth separation.
Snail trail	A mode within Debrief where only the current vessel position plus a short back-track of previous positions is shown.
Stepper control	The Stepper Control is the collection of controls at the top of the Tote panel. The Stepper Control provides controls to move the current time backwards and forwards, controls to edit the stepper itself (edit properties and change display mode).
Symbology	This pair of text characters contained in an REP file indicate the formatting to be applied to this particular track/fix/annotation, although they can be over-ridden once the data is open in Debrief.
T-Zero	A specific time of interest, particularly the start of a particular event. Contexts typically have their own convention of t-zero. Lightweight torpedo launches use weapon-splash time.
TMA solution	Warships (submarines in particular) use Target Motion Analysis (TMA) to produce an estimate of target range. course and speed when the target is held on a bearing/frequency only sensor. TMA solutions frequently represent uncertainty over target location by representing the location as an ellipse - given by a centre-point, an orientation and dimensions for the maximum and minimum axis (as diameter, not radius).
Track	A series of positions recorded for a particular vehicle (ship, submarine, helo, etc). A track has its own characteristics such as colour, label and symbol frequency, and the symbol used to represent it when the symbol highlighter is in use (see Chapter B.5, *Symbol sets*).
Tote	A GUI panel located by default at the lower-left hand side of the Debrief window. The Tote contains the Time Stepper, beneath which are shown the primary and secondary tracks, when assigned. When stepping forward through a *serial* the Tote contains data calculated from the current vessel positions.
VPF	The Vector Product Format is the format of vectored data which may be viewed by Debrief. The main type of VPF data is the Vector Map Level 0, an unclassified global database which includes coastlines, national borders and depth contours. Its supplier (WWW.NIMA.MIL [http://www.nima.mil]) describes it as: “The Vector Product Format (VPF) is a standard format, structure, and organization for large

	geographic databases that are based on a georelational data model and are intended for direct use. ”
XML	The eXtensible Markup Language, as recommend by the World Wide Web consortium.

Index

Colophon

Debrief has been authored in DocBook XML version 5.0 using Oxygen V12. Version 6.5.2 of Saxon used version 1.76 of the Docbook XSL stylesheets and version 1.0 of fop to publish the document to PDF and HTML.

The majority of the text was originally imported from previous Debrief help-files, but the reviews conducted in the transfer to DocBook have resulted in many corrections and clarifications. Sharp Words [http://sharp-words.co.uk/] provided a block of tidying/reformatting in Summer 2011, including an update of the sources to Docbook 5.0.