Simulation software for safe, sustainable and future deltas

DELFT3D FM SUITE

Deltares systems



Delta Shell

Deltares

User Manual

Delta Shell

User Manual

Released for: Delft3D FM Suite 2024 D-HYDRO Suite 2024 SOBEK Suite 3.7

Version: 1.2 Revision: 78359

16 July 2024

Delta Shell, User Manual

Published and printed by:

Deltares Boussinesqweg 1 2629 HV Delft P.O. 177 2600 MH Delft The Netherlands telephone: +31 88 335 82 73 e-mail: Information www: Deltares

For sales contact:

telephone: +31 88 335 81 88 e-mail: Sales www: Sales & Support

For support contact:

telephone: +31 88 335 81 00 e-mail: Support www: Sales & Support

Copyright © 2024 Deltares

All rights reserved. No part of this document may be reproduced in any form by print, photo print, photo copy, microfilm or any other means, without written permission from the publisher: Deltares.

Contents

Li	st of T	ables		vi
Li	st of F	igures		vii
1	A gui 1.1 1.2 1.3 1.4 1.5	ide to th Introdu Overvie Manua Typogra Change	nis manual ction ew version and revisions aphical conventions es with respect to previous versions	1 1 1 2 2
2	Back 2.1 2.2 2.3	ground About t Delta S Overall	he Delta Shell framework	3 3 5 5
3	Gene 3.1 3.2	window 3.1.1 3.1.2 3.1.3 3.1.4 3.1.5 3.1.6 3.1.7 3.1.8 3.1.9 Dockate 3.2.1 2.2.2	view of the GUI vs Project Main (central) window Map Chart Region Toolbox Properties Time navigator Messages ble views Docking tabs separately Multiple tabs	8 9 10 11 13 13 13 14 15 16 16 16
	3.3	3.3.2 3.3.3 3.3.3 3.3.4	Image: Table and second sec	18 18 19 20 20 21 21 22 23 23 23 24 24

			3.3.4.2	Hydro object level		. 24
			3.3.4.3	Routes, cross section definitions, roughness sections,	bran	ches,
				catchments		. 25
			3.3.4.4	Editable items		. 26
		3.3.5	Toolbox .			. 27
		3.3.6	Message	S		. 27
	3.4	Ribbon	s and tool	Dars		. 28
	0.1	341	Ribbons	(shortcut keys)		28
		342	File			29
		343	Home			30
		344	View			30
		315				30
		246	Mon			
		0.4.0	Soripting			. 30
		0.4.7	Scripting			
		3.4.0	Outok)		. 32
		3.4.9	QUICK acc			. 33
4	Case	manad	ement			34
	4 1	Case n	nanageme	nt within the Project window		34
	4.2	Case n	nanageme	nt using Python scripting		35
	7.2	Ouse n	nanageme			. 00
5	Worl	king witl	h the map			37
	5.1	Introdu	ction			. 37
	5.2	Map la	vers			. 37
	5.3	Backor	, ound laver			. 39
	5.4	Coordi	nate svster	ms and transformations		. 39
6	Spat	ial edito	or			42
	6.1	Introdu	ction			. 42
	6.2	Genera	al			. 42
		6.2.1	Overview	of spatial editor		. 42
		6.2.2	Import/ex	port dataset		. 43
		6.2.3	Activate (spatial) model quantity		. 44
		6.2.4	Colorscal	e		. 45
		6.2.5	Render n	node		. 46
		6.2.6	Context r	nenu		. 48
	6.3	Quantit	ty selectior	1		. 48
	6.4	Geome	try operati	ons		. 49
		6.4.1	Polygons			. 50
		6.4.2	Lines			. 50
		6.4.3	Points .			. 51
	6.5	Spatial	operations	3		. 51
		6.5.1	Import po	pint cloud		. 52
		6.5.2	Crop .			. 53
		6.5.3	Delete			. 54
		654	Set value			55
		655	Contour			56
		656	Conv to s	amples		58
		657	Conv to s	natial data	• • •	. 50 50
		658	Merge co	patial data	• • •	. 53 60
		650	Gradiont	ana oala		. 00 61
		0.0.9	Internaled	· · · · · · · · · · · · · · · · · · ·	• • •	. 01 . 01
		0.5.10	Smoothin	ال	• • •	. 02
		0.5.11	SHOUTIF	ıy		. 00

	6.6	Operation stack	68 69 69 72 73 73 74
7	Mod	el coupling	75
÷.	7.1	Model coupling using integrated models within the User Interface	75
	7.2	Model coupling using the BMI standard	75
	7.3	Model coupling using the OpenMI standard	76
8	Com	mand-line and scripting	77
	8.1		77
	8.2	Scripting editor overview	77
		8.2.1 Syntax highlighting	//
		8.2.2 Code completion	/8
		8.2.3 Show extra characters	78
		8.2.4 Regions	79
			79
		0.2.0 RIUDUII	01
		8.2.8 Shortcute	82
	83		02 83
	0.0	8.3.1 Exploring the scripting folder	83
		8.3.2 Working with scripts	85
		8.3.3 Working with folders under the scripting folder	85
	8.4	Bunning scripts from within the Delta Shell GUI (Windows only)	86
	8.5	The Delta Shell Console	88
		8.5.1 Testing and running your scripts using the Delta Shell interactive con-	
		sole (Windows/Linux/MacOSX)	89
		8.5.2 Running your script directly from the commandline (Windows/Linux/-	
			89
P	fore	200	01
R	eieren	Ces	91
Α	How	to use OpenDA for Delta Shell models	92

List of Tables

3.1	Functions and their descriptions within the scripting <i>ribbon</i> of the User Inter- face.	31
3.1	Functions and their descriptions within the scripting <i>ribbon</i> of the User Inter-	32
3.2	Shortcut keys within the scripting editor of the User Interface.	32
3.2	Shortcut keys within the scripting editor of the User Interface	33
8.1	Functions and their descriptions within the scripting <i>ribbon</i> of the User Inter- face.	81
8.1	Functions and their descriptions within the scripting <i>ribbon</i> of the User Inter- face.	82
8.2	Shortcut keys within the scripting editor of the User Interface.	82
8.3 8.4	Context menu function descriptions of a script item within the toolbox Context menu function descriptions of the "Scripts" folder (or one of its sub-	85
	folders) within the toolbox.	86

List of Figures

2.1 2.2 2.3 2.4 2.5	Mindmap with key user stories within Delta Shell.The structure of Delta Shell as integrated modelling suite.From real world to model results with Delta Shell.Mindmap with key architecture subjects of Delta Shell.Top-level architecture of Delta Shell.	3 4 5 6 7
3.1 3.2	Overview of the graphical user interface, example for a SOBEK 3 model	9 10
3.3	The central map view.	11
3.4	The chart window view.	11
3.5	The map window.	12
3.6	Example of the chart window.	13
3.7	Example of the region window.	13
3.8	Example of the Toolbox window	14
3.9	Example of a property grid in the properties window of a flow model	15
3.10	Example of the Time navigator window.	15
3.11	The messages window.	16
3.12	Docking windows on two screens within the User Interface	17
3.13	Bringing the Time Navigator window to the front	17
3.14	Docking the Time Navigator window.	17
3.15	Auto hide the Properties window	18
3.16	The context menu on the project level within the project explorer.	18
3.17	The context menu on the integrated model level within the project explorer.	19
3.18	The context menu on the region level within the project explorer.	20
3.19	The context menu of the central man window without any ragion chieste	21
3.20	The context menu of the central map window including region objects.	22
3.21	The context menu of the table editor	22
3.22	The context menu of the chart view	23
3.24	The context menu of the region level within the Begion window	24
3.25	The context menu of the hydro object level within the Begion window	25
3.26	The context menu of the sub levels of the hydro objects within the Region	_0
	window	26
3.27	The context menu of the editable items of the hydro objects within the Region window.	27
3.28	The context menu for the Messages window.	28
3.29	Perform operations using the shortcut keys	28
3.30	The File ribbon.	29
3.31	The User Interface options dialog	29
3.32	The Home ribbon	30
3.33	The View ribbon.	30
3.34	The <i>Tools</i> ribbon contains just the <i>Data</i> item.	30
3.35	The <i>Map</i> ribbon.	30
3.36	The scripting <i>ribbon</i> within the User Interface.	31
3.37	The quick access toolbar.	33
4.1	Example of case management	35
5.1	The layer Properties editor.	38
5.2	Specify the coordinate system for a network.	40
5.3	Choose coordinate system.	40
6.1	Overview of spatial editor functionality in Map ribbon	42

6.2	Importing a point cloud into the project using the context menu on "project" in the project tree	43
6.3	Importing a point cloud into the project using the "Import" drop-down menu in	
	the Spatial Operations ribbon	44
6.4	Activate the imported point cloud in the spatial editor by double clicking it in the project tree	44
6.5	Activate the imported point cloud in the spatial editor by selecting it from the	• •
	dropdown box in the Map ribbon	44
6.6	Legend button in Map ribbon	45
6.7	Activate the colorscale using the Legend in the map ribbon	45
6.8	Edit the colorscale properties using the context menu on the active layer in the	
	Map Tree	46
6.9	Select the rendermode for the active layer in the property grid.	47
6.10	Example of a coverage rendered as colored numbers.	47
0.11	selecting a smoothing operation for a polygon geometry from the context	18
6 12	Activating a spatial quantity by double clicking it in the project tree (in this	40
0.12	example 'Initial Water Level')	49
6.13	Activating a spatial quantity by selecting it from the dropdown box in the 'Spa-	10
	tial Operations' ribbon	49
6.14	Overview of the available geometry operations in the 'Spatial Operations' ribbon	49
6.15	Activating the polygon operation and drawing polygons in the central map	50
6.16	Activating the line operation and drawing lines in the central map	51
6.17	Activating the 'Add points' operation, drawing them in the central map and	
0.10	assigning a value to them.	51
6.18	Overview of the available spatial operations in the 'Spatial Operations' ribbon .	52
6.19	tions' ribbon	50
6.20	Ontion to perform a coordinate transformation on the imported point cloud	52
6.21	Appearance of import point cloud operation in the operations stack	53
6.22	Performing a crop operation on a point cloud with a polygon using 'Crop' from	00
	the 'Spatial Operations' ribbon	53
6.23	Appearance of crop operation in the operations stack	54
6.24	Performing an delete operation on a point cloud with a polygon using 'Delete'	
	from the 'Spatial Operations' ribbon	54
6.25	Appearance of delete operation in the operations stack	55
6.26	Performing a set value operation (e.g. overwrite) on a point cloud with a poly-	
6 07	gon using 'Set value from the 'Spatial Operations' ribbon	56
6.28	Import a nautical chart as a georeferenced tiff file	57
6.29	Set the right map coordinate system for the geotiff	57
6.30	Performing a contour operation on a nautical chart using lines to define the	0,
	contours and 'Contour' from the 'Spatial Operations' ribbon	57
6.31	Bring the sample set to the front if it appears behind the nautical chart	58
6.32	Appearance of contour operation in the operations stack	58
6.33	Applying the copy to samples operation	58
6.34	Copy to samples operation result	59
6.35	Applying the copy spatial data operation	59
0.30	Copy spatial data operation result	59
0.3/	The merge operation requests a pointwise combination method	00
6.39	Resulting arid coverage	61
6.40	Performing a gradient operation on a point cloud with a polygon using 'Gradi-	01
	ent' from the 'Spatial Operations' ribbon	61

6.41 6.42 6.43	Appearance of gradient operation in the operations stack	62 62 63
6.44	Performing an interpolation operation on a single sample set (without using a polyaon) using (Interpolate) from the (Special Operations) ribbon	64
6.45	Appearance of interpolation of 'set1' to the coverage 'bed level' in the opera- tions stack	64
6.46	Performing an interpolation operation on multiple sample sets (without using a polygon) using (Interpolate) from the (Spatial Operations) ribbon	65
6.47	Appearance of interpolation of 'set1' and 'set2' to the coverage 'bed level' in	00
6.48	Performing a smoothing operation on a point cloud with a polygon using 'Smooth-	65
6.49	Appearance of smoothing operation in the operations stack	66 66
6.50	The cursor for the overwrite operation showing the value of the closest cover- age point	67
6.51	Performing an overwrite operation on a coverage point using 'Single Value' from the 'Spatial Operations' ribbon	68
6.52 6.53	Appearance of overwrite operation in the operations stack	68
6.54	(e.g. branches) are used to construct the 'bed level' coverage	69
6.56	output of the operation (bottom panel)	70
	of sync'.	70
6.55	Editing the value or 'Pointwise operation' of a 'Set Value' operation using the properties panel	71
6.57 6.58	Removing an operation from the stack using the cross icon in the stack menu . Removing an operation from the stack using the context menu on the selected	72 72
6.59	Refresh the stack using the 'Refresh' button so that all operation are (re-)	72
6.60 6.61	Quick link to the output after performing all (enabled) operations	73 74
7.1	Example of an integrated model	75
7.1 8.1	Example of an integrated model	75 77
8.2 8.3	Example of code completion within the Delta Shell scripting editor.	78 78
8.4	Showing tabs within the Delta Shell scripting editor.	78
8.5	Showing EOL characters within the Delta Shell scripting editor.	79
8.0 8.7	Closed region within the Delta Shell scripting editor.	79 70
8.8	Local variables within the Delta Shell scripting editor	80
8.9	Properties of variables within the Delta Shell scripting editor.	80
8.10	Watch variables within the Delta Shell scripting editor.	81
8.11	The scripting <i>ribbon</i> within the User Interface.	81
8.12	The Toolbox window within Delta Shell	83
8.13	Expanded view of the Scripts folder.	84
8.14	Context menu of a script item within the toolbox.	85
8.15	Context menu of the "Scripts" folder (or one of its sub-folders) within the toolbox.	86

8.16	Add an item to your Delta Shell project.	87
8.17	Select script to add to your Delta Shell project.	87
8.18	Scripting editor within the Delta Shell GUI.	88
8.19	Create your own auxiliary functions or use the pre-defined library functions	
	using Python.	88
8.20	Run script within Delta Shell GUI	88
8.21	Delta Shell console application options overview.	88
8.22	Test script in interactive mode.	89
8.23	Run script from command line within console application.	90

1 A guide to this manual

1.1 Introduction

This User Manual concerns the Delta Shell framework.

This module is part of several Modelling suites, released by Deltares as Deltares Systems or Dutch Delta Systems. These modelling suites are build with use of the Delta Shell framework. The framework enables to develop a range of modeling suites, each distinguished by the components and — most significantly — the (numerical) modules, which are plugged in. The modules which are compliant with the Delta Shell framework are released as D-*Name of the module*, for example: D-Flow Flexible Mesh, D-Waves, D-Water Quality, D-Real Time Control and D-Rainfall Runoff.

Therefore, this User Manual is shipped with several modelling suites. On the *Start Page* links are provided to all relevant User Manuals (and Technical Reference Manuals) for that modelling suite. Other user manuals can be referenced. In that case, you need to open the specific user manual from the *Start Page* in the central window. Some texts are shared in different user manuals, in order to improve the readability.

1.2 Overview

To make this manual more accessible we will briefly describe the contents of each chapter.

Chapter 2: Background, introduces the Delta Shell framework as an integrated modelling environment to provide users with a single application which acts as a platform to integrate various models and tools.

Chapter 3: General overview of the GUI, gives a brief introduction to all GUI-components, which are shared between applications based on the Delta Shell framework.

Chapter 4: Case management, introduces two ways to work with cases, (model) variants or scenarios.

The map is always a very - if not, the most - usefull window in modelling in Delta Shell. Read more about the map and coordinate systems in Chapter 5: Working with the map.

Chapter 6: Spatial editor, introduces the spatial editor, a generic feature of the Delta Shell for editing spatial data, such as bathymetry, roughness, viscosity, initial conditions, sediment availability.

Chapter 7: Model coupling, is on models, (timestep based) model coupling and so-called integrated modelling within Delta Shell.

Chapter 8: Command-line and scripting, describes how to run a model without using the GUI. For the advanced user, there is also an introduction to the very powerfull feature of scripting, which enables the user to adjust and run a model without using the GUI, even to setup complete models from scratch.

1.3 Manual version and revisions

This manual applies to SOBEK 3 suite (version 3.5 and higher) and Delft3D Flexible Mesh Suite (version 2016 and higher).

1.4 Typographical conventions

Throughout this manual, the following conventions help you to distinguish between different elements of text.

Example	Description
Module Project	Title of a window or a sub-window are in given in bold . Sub-windows are displayed in the Module window and cannot be moved. Windows can be moved independently from the Mod- ule window, such as the Visualisation Area window.
Save	Item from a menu, title of a push button or the name of a user interface input field. Upon selecting this item (click or in some cases double click with the left mouse button on it) a related action will be executed; in most cases it will result in displaying some other (sub-)window. In case of an input field you are supposed to enter input data of the required format and in the required domain.
<\tutorial\wave\swan-curvi> <siu.mdw></siu.mdw>	Directory names, filenames, and path names are expressed between angle brackets, <>. For Linux environments a forward slash (/) is used instead of the backward slash (\) for Windows environments.
"27 08 1999"	Data to be typed by you into the input fields are dis- played between double quotes. Selections of menu items, option boxes etc. are de- scribed as such: for instance 'select <i>Save</i> and go to the next window'.
delft3d-menu	Commands to be typed by you are given in the font Courier New, 10 points.
	In this User manual, user actions are indicated with this arrow.
[m s ⁻¹] [–]	Units are given between square brackets when used next to the formulae. Leaving them out might result in misinterpretation. Most units will be in SI notation. [m AD] stands for 'meter Above Datum', which denotes a level relative to the vertical reference system in the model.

1.5 Changes with respect to previous versions

In this edition changes have been made to:

- ♦ the chapter 6: Spatial editor
- ♦ the Appendix A: How to use OpenDA for Delta Shell models

2 Background

2.1 About the Delta Shell framework

After its first release in 2012, the Delta Shell framework will gradually be extended to become a full-fledged framework for the integration of a variety of environmental models.

The Delta Shell framework is an integrated modelling environment to provide users with a single application which acts as a platform to integrate various models and tools. This is achieved by making use of a software framework specifically focused to provide a set of components which can be reused by all kinds of environmental models. The overall picture of what functionality Delta Shell needs to provide has been proposed by a variety of (potential) users and has been captured in so-called user stories. These user stories have been gathered in a so-called mind map which is presented in Figure 2.1. As can be seen from the mind map,



Figure 2.1: Mindmap with key user stories within Delta Shell.

Delta Shell has to comply with a broad range of wishes defined by the various users and user groups. Main topics (depicted in the main branches) are:

- ♦ The definition of the integrated modelling framework,
- ♦ Data management: structuring, importing/exporting, storage.
- ♦ Dealing with environmental domain libraries (hydro, geo).
- Application interfaces (APIs) such as a graphical user interface or a command line interfce using scripts for batch operation.
- ♦ Documentation and support

Delta Shell provides a user-friendly and open framework for environmental applications. Figure 2.2 shows the principles of the Delta Shell framework. Delta Shell allows to combine different modules (plug-ins), such as for example, D-Flow 1D, D-RTC, D-Water Quality, or D-RR, which results in the SOBEK 3 suite. In this fashion, it is possible to compose various dedicated software suites within a single framework and preserving the same look-and-feel at

the same time. Furthermore, Delta Shell contains common plug-ins with generic functionality which may be used by all model plug-ins that have been integrated within the framework. Finally, Delta Shell contains tools for setting up or importing different types of models, perform simulations of the different models or combinations of models and analyse model results.



Figure 2.2: The structure of Delta Shell as integrated modelling suite.

A model is by definition a simplified version of (a possible) reality. A generic model is a mathematical description of physical processes provided as a computer program, for example D-Flow 1D, which is a model of surface water flow that solves the Saint-Venant equations in one dimension. D-Flow 1D is a numerical model, because the governing flow equations are solved numerically. In combination with data for a specific part of the world the model becomes a site-specific model, for example the D-Flow 1D-model for the river Rhine from Maxau to Lobith. Within a site-specific model a set of boundary conditions forms a scenario.

Delta Shell follows a layered concept to get from 'real world' to 'numerical model result' (Figure 2.3). Real objects like rivers, lakes, buildings, hills, bridges, pumps, culverts, etc. usually are available on maps or other (digital) format. Based on such data Delta Shell helps to create a schematisation, i.e. a network with model objects that correspond to the real objects. The schematisation makes the model site-specific. A computational grid is added, which is part of the discretisation of real world processes like surface water flow into a numerical model. This model can be run under different sets of boundary conditions (scenarios). Numerical solution of the flow equations with a Delta Shell plugin finally produces model results.



Figure 2.3: From real world to model results with Delta Shell.

2.2 Delta Shell: Vision Statement

For users:

The framework should facilitate modellers in: schematization (with/without user interface), calibration of models, running of models in standardized modelling environments (ex. OpenMI), batch execution (from command line) and parallel execution (on clusters). All modellers should be able to use the Delta Shell Framework combined with the modelling tools of their preference (ex. D-Flow 1D/2D, D-Flow FM, Simona, Mike11, etc.).

For developers:

The Delta Shell Framework should provide a generic, transparent and simple way of developing new or coupling existing modelling tools (plug-ins). The emphasis should be on reuse and expansion of existing code and features.

2.3 Overall Architecture

This section gives an overview of the Delta Shell architecture and a description of its main components. Delta Shell has been designed using a flexible architecture which can easily facilitate the use of external applications. The concept of the overall architecture is similar to existing commercial products like Eclipse or Microsoft Visual Studio Shell. The most important subjects considering the architecture of Delta Shell are presented in Figure 2.4. These are the subjects that need to come together within the architecture of Delta Shell.

The subjects within the main branches of the mindmap that grasp the overall architecture of Delta Shell are:

♦ Interoperability and/or Model Coupling. Define an architecture that is able to couple a



Figure 2.4: Mindmap with key architecture subjects of Delta Shell.

variety of models in a correct fashion, both numerical as physics-wise.

- Applications and plug-ins. Provide a flexible framework architecture such that it becomes possible to build applications (such as a command-line runner) or integrate (model) plugins within the framework in a generic fashion.
- Data. The Delta Shell framework needs to provide functionality to deal with the storage, import/export, transformation of data, as well as to provide support for the scientific character of (most of) these data.
- Domains. Currently, Delta Shell needs to provide functionality for the Hydro and Geospatial domains. Support for other domains is foreseen in the future.
- Scripting. To be able to set-up a Delta Shell project from the command-line, or perform batch simulations using Delta Shell, it needs to support scripting.

To grasp all of the above subjects within the Delta Shell framework, the architecture of Delta Shell consists of multiple layers that will all be described in this document. The resulting overall architecture of Delta Shell is depicted in Figure 2.5.

The highest level of the Delta Shell architecture is shown in Figure 2.5. The central component of the system is the modeling framework. The framework consists of a set of class libraries which can be used throughout the system by the common and application plug-ins and standalone applications such as the Graphical User Interface (GUI) or command-line runner.

To integrate new plug-ins within Delta Shell the framework provides interfaces for the plug-in to comply with via its Core and GUI APIs (for example the IPlugin, IModel, IModelProvider, IView or IViewProvider interfaces). The same holds for adding stand alone top level applications (using the IApplication interface). Currently Delta Shell offers 2 of these applications, e.g. the Delta Shell GUI and the Delta Shell command line runner, both of which combine the available plug-ins within a single environment. It should be noted that other extensions of the Delta Shell environment, for example an ArcGIS Extension, or Web Service, can easily be developed as an alternative to existing GUI and console applications. Only the top-level part



Figure 2.5: Top-level architecture of Delta Shell.

of the system needs to be replaced and the rest of the components will work as before without making any changes.

The plug-ins have been implemented as extensions to Delta Shell and can be divided into two groups. The first group consists of the so called common plug-ins which provide common functionality such as data types (network data types, OGC-compliant GIS data types, multidimensional data types etc.) or which provide generic functionality such as the import/export of data, the editing of data using various custom controls, or the visualization of data. The functionality implemented in these common plug-ins is made available to the second group of application plug-ins implicitly via the framework, and explicitly by direct use of .NET assemblies such as the GeoAPI or NetCDF assemblies (dll's). This second group consists of specific plug-ins which integrate some computational model (currently D-Flow 1D, D-Water Quality, D-RTC, and D-RR) into the system.

3 General overview of the GUI

As shown in Figure 2.5, one of the interfaces within the User Interface makes it possible to extend the available (model) plug-ins with Graphical User Interfaces (GUIs). In this fashion, all user interfaces of model plug-ins integrated within Delta Shell have the same look-and-feel.

This chapter introduces all GUI-components, which are shared between applications based on the Delta Shell framework.

3.1 Windows

As Delta Shell is an integrated modelling suite, the application is project-based. Within a project several models may be run and combined.

The main user interface is organized in a set of tool and document windows. An example is given in Figure 3.1. The tool windows show properties of the current project, whereas document windows are used to visualize or edit a specific data type. Tool windows can be docked where you prefer — even at a second display. Document windows are, when placed within the framework, always in the central area but may also be docked stand-alone (on a second display, for example). Examples of tool windows are:

- ♦ Project
- ♦ Map
- ♦ Region
- ♦ Properties
- ♦ Chart
- ♦ Data
- ♦ Time navigator
- ♦ Messages
- ♦ Toolbox

Examples of document windows are:

- ♦ Map(s)
- ♦ Editor(s)



Figure 3.1: Overview of the graphical user interface, example for a SOBEK3 model.

In this chapter all tool windows, menus, dockable views, context menus, and ribbons and toolbars will be described. Map functionality and the spatial editor are treated in separate chapters. The specific editors for the different models are described in the user manuals belonging to those model plug-ins.

3.1.1 Project

The **Project** window is the main navigation window for the project data, showing the total workspace in a tree view (Figure 3.2). In the **Project** all project components are shown. For a project containing a one-dimensional flow model and a map these may look like:

- ♦ The model (in this case a 1-dimensional flow model)
 - The model input
 - The network(s)
 - The computational grid(s)
 - The initial conditions
 - o The boundary conditions
 - o Lateral data
 - o Roughness
 - o Wind data
 - □ The model output/results
- ♦ The map

All project items with sub-levels can be collapsed by a mouse-click on the '-' sign in the tree view. Project data can be sorted by adding new folders to the project tree view and moving models or movable items to designated folders. By clicking on the top left icon in Figure 3.2 the active item in the central Map is located in the tree view.



Figure 3.2: The project tree window.

Several possibilities exist to work with the tree view:

- ♦ Left mouse-click to select
- ♦ Right mouse-click gives a context menu with available actions
- Double-click to show a map or editor in the main (central) window, depending on the parameter

3.1.2 Main (central) window

The main window (Figure 3.3) is by default always placed in the middle of the screen. It can also be docked separately, for example on a second display. It is used to present a map for all geo-referenced modeldata, the editors for other data, and results in charts. The editors for other data are model-specific and therefore described in the manuals for the various model plug-ins.

All items with a geo-reference will be presented on the central map, for example: network, computational grid and output data as layers, comparable to a geographical information system (GIS). Working with these layers is described in section 5.2.



Figure 3.3: The central map view.

When working in the central map, for example on a network, it is possible to add, adjust or delete network components, which is described thoroughly in the D-Flow 1D manual.

Results in charts windows (Figure 3.6) are presented by combining a table view and a 'xy'-plot. The user can visualize separate data points by selecting rows in the table view. The data that is presented in the chart window may also be exported to a *.csv file by clicking the *Csv* export button.

Clipboard import 🖉 Csv i	mport 🖳 Csv export							
Fime [yyyy/MM/dd HH:mm:ss]	Discharge at (Zandmaas13, 276) [m³/s]					(=)		
2000/01/01 00:00:00	84.026				Discharge at	(Zandma	as13, 276) [m³/s]
2000/01/01 01:00:00	567.71		Г					
000/01/01 02:00:00	513.14		550					
000/01/01 03:00:00	415.28		E00					
000/01/0104:00:00	354.91		300					
000/01/0105:00:00	312.8		450					
000/01/01 06:00:00	278.53		400					
000/01/0107:00:00	246.99		250					
000/01/01 08:00:00	212.16		350					
000/01/01 09:00:00	184.89		300					
000/01/01 10:00:00	163.78		250					
000/01/01 11:00:00	137.06		200					
000/01/01 12:00:00	105.97		200					
000/01/01 13:00:00	74.292		150	1 1				
000/01/01 14:00:00	44.274		100					
000/01/01 15:00:00	18.512		50					
000/01/01 16:00:00	-0.087847		30					
000/01/01 17:00:00	-5.1014		0 -					
000/01/01 18:00:00	2.1799		Ļ	1/3/2000	1/6/200	0 1	1/9/2000	1/12/200
000/01/01 19:00:00	1.5041	•			Time [yyyy/l	M/dd HH	:mm:ss]	

Figure 3.4: The chart window view.

3.1.3 Map

The **Map** window (Figure 3.5) manages the active map. In this window layers within the active map can be shown, hidden or adjusted. With the four icons in the top left of the window new \langle shp \rangle - or \langle wms \rangle -layers can be added, removed or exported. With the icons in the top right of the window, the window can be removed or hidden. The window can be retrieved by clicking on *Map* in *View* ribbon.

Map	▼ ↓ ×
£ € ≋ ≤	
→ Map (WGS 84 / Pseudo-Mercator) → Maas_j17_5-v1_JD4_4000 → Ø Region → Ø Nodes → Ø Deservation Points → Ø Observation Points → Ø Culverts → Ø Extra Resistances → Ø Culverts → Ø Composite Structure → Ø Composite Structure → Ø Routes ⊕ Ø Input ↓ Ø Output ⊕ Ø Input ↓ Ø Output ⊕ Ø Nodel ⊕ Ø Roughness data ⊕ Initial water flow ⊕ Ø Output	
Map Chart Region Operations Toolbox	

Figure 3.5: The map window.

3.1.4 Chart

The **Chart** window (Figure 3.6) can be used to stack/unstack chart series of the same type or to convert chart series to a certain type (area, line, point, or bar series), Furthermore, series can be selected or deselected within this window.

Chart	• •
<u>100</u>	Stack serie
🗹 🥥 🕍 Discharge at (Zandmaas13, 276) [m³/s] Left Axis	
Map Chart Region Operations Toolbox	

Figure 3.6: Example of the chart window.

3.1.5 Region

In the **Region** window (Figure 3.7) all hydrological objects are shown, like river-networks, -catchments or waterbodies. The **Region** window is related to the selected model. While in the central map the model objects are arranged according to their spatial location, the **Region** sorts the model objects by category and chainage. A component that is selected in the **Region** window is also automatically selected in the central map and vice versa.



Figure 3.7: Example of the region window.

3.1.6 Toolbox

In the **Toolbox** window (Figure 3.8), users can quickly add a variety of items (models, standalone items) to the project tree, by selecting 'Add new item' or 'Add new model' after a rightmouse-click on the desired item. Furthermore, users may manage their Python scripts within the **Toolbox** window.



Figure 3.8: Example of the Toolbox window.

3.1.7 Properties

The **Properties** window shows properties for an active selection of the graphical user interface. When a model object is selected in the **Region** window it shows the properties of this object. Accordingly, the **Properties** window of an item selected in the **Project** shows data related to the selected item, for example the simulation time of a <flow model> or a list with output parameters when clicking on the <output> entry. Figure 3.9 shows an example for the properties of a flow model.

In the **Properties** window data can also be edited. If the property grid is insufficient to display the information, for example in case of time series, an additional editor can be opened.

]⊉↓							
4	General							
	Name	Flow1D						
	Status	None						
	Current time	2016-11-10 00:00:00						
Þ	Initial conditions							
Þ	Model settings							
4	Output parameters							
	Model output time step	0d 01:00:00:000						
4	Run parameters							
Þ	Model parameters	ModelApiParameterProp						
	Start time	2016-11-10 00:00:00						
	Time step	0d 01:00:00:000						
	Stop time	2016-11-11 00:00:00						
	Use restart	True						
	Write restart	True						
	Use save state time ran	False						
	Save state time range:	0001-01-01 00:00:00						
	Save state time range: t	0d 00:00:00						
	Save state time range:	0001-01-01 00:00:00						
_								

Figure 3.9: Example of a property grid in the properties window of a flow model.

3.1.8 Time navigator

The **Time navigator** is used to navigate through time(steps) of any time-dependent variable. An example is given in Figure 3.10. Each map or side view that shows time-dependent variables has its own time navigator. Users may also play results of time-dependent variables continuously.

Time Navigator													•	ņх
07/19/2013 00:00:00	-													-
Delay: 0.1 sec	-7-20 (Frid	13 0:00 ay 19 juli 21	19-7-201 013 till Monda	.3 12:00 ay 22 juli 20	20-7-20: 013)	13 0:00	20-7-20:	13 12:00	21-7-20	13 0:00	21-7-201	3 12:00	22-7-1	2013 0:

Figure 3.10: Example of the Time navigator window.

3.1.9 Messages

The **Messages** window (Figure 3.11) is a logging window. Messages sent from models or different parts of the system are shown here. When a message is too large to fit within the **Messages** window, the user can open a single message (**??**) separately by right-mouse-clicking the message and selecting the 'Show details' option.

N	essag	es		• 🗆 X
		16:12:04.6086	** INFO : 6(+)Deltares, SOBEXSIM Version 4.08.007.37207, Nov 27 2014, 14.09:03 ** INFO : Simulation stated ** WARNING Solutorian initial: Crest level adjusted to bed level in structure Weir, id = ST_Amerongen_zom##1 ** INFO : iopt : 1 grds = 4 dhrs : 1 kis 5 m = 4 sc 1 ** INFO : iopt : 1 grds = 7 dhrs : 3 kis : 3 dhrs : 3 dis :	DeltaShell.Plugins.DelftModels.WaterFlowModel.WaterFlowModel
	0	16:12:04.6086	Sobel Log	DeltsShell.Plugins.DelftModels.WaterFlowModel.WaterFlowModel
	(16:12:04.5866	Wall clock * Rijn-instance-24': Sant time: 15-0e: 2014 1612:04. End time 15-0e: 2014 1612:04. Ron Duration: 00:000:00187	DelftTools.Shell.Core.Workflow.TimeDependentModelBase
Ш	0	16:12:04.5866	Model "Rijn-instance-24" has finished in 5 steps.	DelftTools.Shell.Core.Workflow.TimeDependentModelBase
	(16:12:04.5866	Wall clock ' hjn-flow-model': Start time: 15-0e: 2014 161:204. End time 15-0e: 2014 161:204. Ron Duration: 00:00000185	DelftTools.Shell.Core.Workflow.TimeDependentModelBase
Ш	(0 16:12:04.5856	Model "rijn-flow-model" has finished in 5 steps.	DelftTools.Shell.Core.Workflow.TimeDependentModelBase
Ľ	L. Messa	ges Time Navi	pator	

Figure 3.11: The messages window.

Within the **Messages** window the user may select the verbosity of the shown messages, ranging from 'Info messages' to 'Warning messages' to 'Error messages'. It is also possible to clear all messages by clicking **X**.

Furthermore, a run report is shown in the output in the **Project** for each model simulation. This run report contains all the messages (from Delta Shell and the model plug-ins) that occur during a simulation.

Finally, an application log is kept for each session of Delta Shell in the project database. In this log-file, which can be accessed through the *File/Help* or *Home* menus, all messages are stored.

3.2 Dockable views

The User Interface offers lots of freedom to customize dockable views, which are discussed in this section.

3.2.1 Docking tabs separately

Within the User Interface the user can dock the separate windows according to personal preferences. These preferences are then saved for future use of the framework. An example of such preferences is presented in Figure 3.12, where windows have been docked on two screens.



Figure 3.12: Docking windows on two screens within the User Interface.

3.2.2 Multiple tabs

In case two windows are docked in one view, the underlying window (tab) can be brought to the front by simply selecting the tab, as is shown here.

Messages X Time Navigator	
1 New channel command: <back> remove last point, <s> toggle snapping, L decrease step, R increase step</s></back>	11/1/2016 11:47:07 AM 🔺
Adding welcome page	11/1/2016 11:32:28 AM 📃
Could not load the requested dock layout. The settings are invalid and will be reset to the default state.	11/1/2016 11:32:28 AM
1 Main window created.	11/1/2016 11:32:27 AM
Hiding splash screen	11/1/2016 11:32:26 AM
1 Started in 1943 sec	11/1/2016 11·32·26 AM



By dragging dockable windows with the left mouse button and dropping the window left, right, above or below another one the graphical user interface can be customized according to personal preferences. Here an example of the **Time Navigator** window being docked to the right of the **Messages** window.

Mes	age	S	₩ џ	х	Time Navigator
0	0	New channel command: <back> remove last point, <s> toggle snap</s></back>	11/1/2016 11:47:07 AM	-	▶ II 144 44 9∌ 144
	0	Adding welcome page	11/1/2016 11:32:28 AM		
0	4	Could not load the requested dock layout. The settings are invalid an	11/1/2016 11:32:28 AM		01/01/0001 00:00:00
-	0	Main window created.	11/1/2016 11:32:27 AM		Delay: 0.1 sec
<u> </u>	0	Hiding splash screen	11/1/2016 11:32:26 AM		
	0	Started in 19.43 sec	11/1/2016 11:32:26 AM	-	
	-			1	

Figure 3.14: Docking the Time Navigator window.

Additional features are the possibility to remove or (auto) hide the window (top right in Figure 3.14). In case of removal, the window can be retrieved by two left mouse-clicks on *Time Navigator* in the *View* ribbon. Hiding the **Properties** window results in:



Figure 3.15: Auto hide the Properties window

3.3 Context menus

Depending on the active window, different context menus are present when right-mouseclicking on items within this window. This section will treat these context menus per active window.

3.3.1 Project

Within the **Project** window, a variety of levels and/or items with different context menus are present. These will be described here.

3.3.1.1 Project level

The context menu of the project level within the project explorer is shown in Figure 3.16. It contains the following choices:



Figure 3.16: The context menu on the project level within the project explorer.

- ♦ Add ⇒ Option to add models and/or items to project (depending on installed model plugins)
- ♦ Import ... ⇒ Opens selection window for a variety of available importers (if present)
- ♦ Export ... ⇒ Opens selection window for a variety of available exporters (if present)
- \diamond *Cut* \Rightarrow Cuts current project for pasting elsewhere
- \diamond Copy \Rightarrow Copies current project for pasting elsewhere
- \diamond Paste \Rightarrow Pastes current project available on the clipboard

- ♦ *Rename* ⇒ Rename the current project
- ♦ Run All Models ⇒ Runs all models available in the project
- ♦ Stop All Models ⇒ Stops running of all models currently running within the project
- ♦ Clear All Models Output ⇒ Clears all model output of models available within the project
- \diamond *Expand All* ... \Rightarrow Expands all project items
- \diamond Collapse All ... \Rightarrow Collapses all project items
- ♦ Properties ⇒ Switches to Properties window of active project

3.3.1.2 Integrated model level

The context menu of the integrated model level within the project explorer is shown in Figure 3.17. It contains the following choices:



Figure 3.17: The context menu on the integrated model level within the project explorer.

- ◇ Open With ⇒ Opens a dialog with choices to open the integrated model (for example with 'Hydro Model Settings')
- \diamond *Run Model* \Rightarrow Runs the integrated model (including all its sub models)
- \diamond Stop Model \Rightarrow Stops running of the integrated model
- ♦ Clear Model Output ⇒ Clears the model output of the integrated model (including all its sub models)
- \diamond Cut \Rightarrow Cuts the integrated model for pasting elsewhere
- \diamond Copy \Rightarrow Copies the integrated model for pasting elsewhere
- \diamond Delete \Rightarrow Deletes the integrated model from the project
- \diamond *Rename* \Rightarrow Rename the integrated model
- \diamond Import... \Rightarrow Opens selection window for a variety of available importers (if present)
- \diamond *Export*... \Rightarrow Opens selection window for a variety of available exporters (if present)
- \diamond Expand All ... \Rightarrow Expands all integrated model items
- ♦ Collapse All ... ⇒ Collapses all integrated model items
- ♦ *Properties* ⇒ Switches to **Properties** window of active integrated model

3.3.1.3 Region level

The context menu of the region level within the project explorer is shown in Figure 3.18. It contains the following choices:

	Import
	Export
do	Cu <u>t</u>
	<u>С</u> ору
12	Paste
\times	<u>D</u> elete
	Re <u>n</u> ame
Š	<u>U</u> nlink
60	<u>L</u> ink
P	<u>P</u> roperties
2	Add sub region

Figure 3.18: The context menu on the region level within the project explorer.

- \diamond Import... \Rightarrow Opens selection window for a variety of available importers (if present)
- \diamond *Export*... \Rightarrow Opens selection window for a variety of available exporters (if present)
- \diamond *Cut* \Rightarrow Cuts the region for pasting elsewhere
- \diamond Copy \Rightarrow Copies the region for pasting elsewhere
- \diamond Paste \Rightarrow Pastes current region available on the clipboard
- \diamond *Delete* \Rightarrow Deletes the region model from the project
- \diamond *Rename* \Rightarrow Rename the region
- $\diamond \quad Unlink \Rightarrow Unlink region from model$
- \diamond Link \Rightarrow Link the region to model
- ♦ Properties ⇒ Switches to Properties window of active integrated model
- \diamond Add sub region \Rightarrow Opens selection window to add sub regions to region

3.3.1.4 Model level

The context menu of the model level within the project explorer is shown in Figure 3.19. It always contains the following choices:

	Open With
	Run Model
	Stop Model
1	Clear Model Output
d	Cut
	Сору
×	Delete
	Rename
	Import
	Export
	Expand All
	Collapse All
P	Properties
	Validate
¢	Use Previous Output as Initial Condition 🔹 🕨

Figure 3.19: The context menu on the model level within the project explorer.

- ◊ Open With ⇒ Opens a dialog with choices to open the model (for example with 'Validation report')
- \diamond *Run Model* \Rightarrow Runs the model
- ♦ Stop Model ⇒ Stops running of the model
- \diamond Clear Model Output \Rightarrow Clears the model output
- \diamond *Cut* \Rightarrow Cuts the model for pasting elsewhere
- \diamond Copy \Rightarrow Copies the model for pasting elsewhere
- \diamond *Delete* \Rightarrow Deletes the model from the project/integrated model
- \diamond *Rename* \Rightarrow Rename the integrated model
- \diamond Import... \Rightarrow Opens selection window for a variety of available importers (if present)
- ♦ Export ... ⇒ Opens selection window for a variety of available exporters (if present)
- \diamond *Expand All* . . . \Rightarrow Expands all model items
- \diamond Collapse All . . . \Rightarrow Collapses all model items
- ♦ Properties ⇒ Switches to Properties window of active integrated model
- \diamond Validate \Rightarrow Opens the validation report of the model

As can be seen in Figure 3.19, context menus on the model level can also contain model specific choices. These choices are described in the user manuals for the specific models.

3.3.1.5 Other items

For the other items in the project explorer the following holds:

- Folder items have the same context menu as the project level context menu depicted in Figure 3.16.
- All other project items have context menus similar to the context menu on the region level as shown in Figure 3.18

3.3.2 Main (Central Map window)

The **Main** or **Central Map** window may consist of multiple tabs, ranging from the region editor to table or chart editors or views. A description of the various context menus is given in this section.

3.3.2.1 Region editor

Without any region objects the region editor context menu only consists of choices regarding the underlying map layer, see Figure 3.20:



Figure 3.20: The context menu of the central map window without any region objects.

- ♦ Export map as image
- ♦ Zoom to extents

When the region objects are also projected on the **Central Map** window, the context menu is expanded with the more choices, see for example Figure 3.21:



Figure 3.21: The context menu of the central map window including region objects.

- ♦ Edit ⇒ Edit objects of region
- \diamond Select \Rightarrow Select objects on region
- \diamond Copy \Rightarrow Copy objects on region for pasting elsewhere
- ♦ Delete selection
- ♦ Query time series
- ♦ Insert Node
- ♦ Remove Node
- ♦ Reverse direction
- ◊ Import cross section(s) from csv
- ♦ Import selected features to branch layer

3.3.2.2 Table editor

The context menu of the table editors within the **Main** window as shown in Figure 3.22 contains the following choices:

	Time [yyyy-MM-dd	flow
	2010-11-01 00:00:00	500
Þ	2010-11-02 23:59:00	500
	2010-11-03 00:00:00	Сору
	2010-11-04 00:00:00	🖺 Paste
	2010-11-04 00:01:00	X Delete
	2010-11-06 00:00:00	500
*		

Figure 3.22: The context menu of the table editor.

- ♦ Copy
- ♦ Paste
- ♦ Delete

3.3.2.3 Charts

The context menu of the charts within the **Main** window as shown in Figure 3.23 contains the following choice:



Figure 3.23: The context menu of the chart view.

♦ Export as Image ...

3.3.3 Map

The context menus for the Map window are described in ??.

3.3.4 Region

Within the **Region** window a variety of context menus is present on different levels within the region tree.

3.3.4.1 Region level

The context menu of the region level within the **Region** window as shown in Figure 3.24 contains the following choice:



Figure 3.24: The context menu of the region level within the Region window.

♦ Add sub region

3.3.4.2 Hydro object level

The context menu of the hydro object level within the **Region** window as shown in Figure 3.25 contains the following choices:



Figure 3.25: The context menu of the hydro object level within the Region window.

- \diamond *Paste* \Rightarrow Paste items onto hydro object
- *◇ Rename*
- ♦ Add Branch
- ♦ Properties

3.3.4.3 Routes, cross section definitions, roughness sections, branches, catchments

The context menu of the various sub levels of the hydro objects within the **Region** window as shown in Figure 3.26 contains the following choices:


Figure 3.26: The context menu of the sub levels of the hydro objects within the Region window.

- ♦ Zoom to branch
- ♦ Copy
- ♦ Paste ...
- ♦ Delete
- ♦ Rename
- \diamond Add ... \Rightarrow Adds items onto hydro object
- ♦ Properties

3.3.4.4 Editable items

The context menu of the editable items of the hydro objects within the **Region** window as shown in Figure 3.27 contains the following choices:

Region		• 🗆 X			
🖃 🖗 Region					
🗄 💥 network					
🖨 式 Routes		=			
	jn-Lek	-	17.00		
	jn-Waal		R. A.		
Bovenri	jn-Ussel				
🖶 📆 Shared Cro	ss Section Definitions		和自己在自己		
🕀 📊 Sections (ro	oughness)		No.		
🕀 🔧 Bovenrijn					
🗎 🗎 🖓 Waal4	🖶 🔆 Waal4				
	elWaal: 0.00		- C		
	Open				
	Open With				
	Zoom to feature				
	Cut				
	Сору	C	trl+C		
	Paste into	C	trl+V		
- @ 91 ×	Delete				
Chart Region Tool	Rename				
	Properties				

Figure 3.27: The context menu of the editable items of the hydro objects within the *Region* window.

- ♦ Open
- ♦ Open With ...
- ♦ Zoom to feature
- ♦ Cut
- ♦ Copy
- ♦ Paste into
- ♦ Delete
- ♦ Rename
- ♦ Properties

3.3.5 Toolbox

The context menus for the **Toolbox** window is described in **??**.

3.3.6 Messages

The context menu for the **Messages** window as shown in Figure 3.28 contains the following choices:

Messages	- ↓ ×
1 Dimr [2017-10-05 15:48:09.518] Info >> 1036200.0: flow model.Update(600.0)	10/5/2017 3:48:09 PM 🔺
10/5/2017 3:48 PM Dim [2017-10-05 15:48:09.518] Info >> 1036200.0: flow model.Update(600.0)	
Dimr [2017-10-05 15:48:09.518] Info >> 1036200.0: real-time control.Update(-1.0)	10/5/2017 3:48:09 PM
Dimr [2017-10-05 15:48:09.456] Info >> 1035600.0: flow model.Update(600.0)	10/5/2017 3:48:09 PM
Dimr [2017-10-05 15;48:09.456] Info >> 1035600.0; real-time control.Update(-1.0)	10/5/2017 3:48:09 PM
Dimr [2017-10-05 15;48:09.409] Info >> 1035000.0: flow model.Update(600.0)	10/5/2017 3:48:09 PM
Dimr [2017-10-05 15;48:09.409] Info >> 1035000.0: real-time control.Update(-1.0)	10/5/2017 3:48:09 PM
Dimr [2017-10-05 15:48:09.346] Info >> 1034400.0: flow model.Update(600.0)	10/5/2017 3:48:09 PM 🖵

Messages Time Navigator



- ♦ Copy
- \diamond Show details \Rightarrow Open separate window with detailed message information
- ♦ Clear all

3.4 Ribbons and toolbars

The user can access the toolbars arranged in *ribbons*. Model plug-ins can have their own model specific *ribbon*. The *ribbon* may be auto collapsed by activating the *Collapse the Ribbon* button when right-mouse-clicking on the *ribbon*.

3.4.1 Ribbons (shortcut keys)

The User Interface makes use of ribbons, just like Microsoft Office. You can use these ribbons for most of the operations. With the ribbons comes shortcut key functionality, providing shortcuts to perform operations. If you press Alt, you will see the letters and numbers to access the ribbons and the ribbon contents (i.e. operations). For example, Alt + H will lead you to the *Home*-ribbon (Figure 3.29).

Note: Implementation of the shortcut key functionality is still work in progress.



Figure 3.29: Perform operations using the shortcut keys

3.4.2 File

The left-most *ribbon* is the *File* ribbon. It has menu-items comparable to most Microsoft applications. Furthermore, it offers users save and open functionality, as well as the *Info* and *Options* dialogs, as shown in Figure 3.30 and Figure 3.31.

E	Info		Project1 - Delft3D FM Suite 2020.01 HMWQ (1.6.0.	45780)	- 0 ×
Info New Open Save	Product name Dr. Version LL Copyright C Telephone 23	elf3D FM Suite 2020.01 HMWQ User name 6.0.45780 Machine name Deltares 2018 Operating syste Hild Jauponf@deltares.nl Processor count 31 (0) 68 335 8100 Process type	rajen L01907 m Microsoft Windows 10 Enterprise (Microsoft Windo 8 64 bit	ws 117 6-2.9000.0	Deltares
Save As Close Options	Application plugins Delta Shell Common Tools Plugin Plugin for common views and data.		Version : 1.4.0.45777 File version : 1.4.0.4577	User-startics plages Der configuration Plagin 000 Provides possibilities to configure DMR settings Provides possibilities to configure DMR settings Plants and Compare Table Names (201)	Version : 1.6.0.45780 File version : 1.0.0
	Provides implementation of the data. Provides inducementation Provides functionality to create and in Provides functionality to create Provides functionality to create Provides functionality Provides functionality Provides functionality Provides functionality Provides functionality Provides functionality for grade Provides functionality for grade	access to the Defa Shell project files using Whilewarde libery in integrated models. for models, such as D-Flow TD. ers and channels, and channels, lakes, estuarine & castal areas and see data. metrical information in the hydro region.	 File Version 12.02 (Version 15.4.0210) Version 15.4.02100 Version 15.1.10 Version 12.7.7.42100 Version 12.0.2.17.42100 Version 12.0.2.17.42100 Version 15.0.42100 Version 15.0.42100 Version 15.0.42100 Version 15.0.42100 Version 11.2.0.42100 Version 12.0.2.17.42100 	Plays the common Views and data. Plays the control Views (0) Plays	File retrien 11.000 Weissin 27.0.5997 File retrien 13.00 File retrien 13.00 File retrien 13.00 File retrien 13.00 File retrien 13.00 Weissin 12.07.4970 File retrien 13.00 Weissin 13.06.49700 File retrien 13.00 File retrien 13.00 File retrien 13.00 File retrien 13.00
	Data Shell Scripting Plagin Prosider scripting Encidencity. Starphage GS Plagin Provider common GIS functionality. Provider common GIS functionality. Toblox containing code snippets, mc	sdels, data items and scripts.	Vension : 1.4.045777 File vension : 1.1.0.045777 File vension : 1.4.045777 File vension : 1.4.045777 File vension : 1.4.0.045777	Deta Santi Scriptog Purgle (UI) Provides compting functionality Provides common GS functionality Provides common GS functionality Deta Santi Toobere Plugie (U) Deta Santi Toobere Plugie (U)	Version : 1.40.45777 File version : 1.1.0.0 Version : 1.4.0.45777 File version : 1.4.0.45777 File version : 1.4.0.45777 File version : 1.1.00
Optior : Ger	ns neral	User Settinas	igure 3.30: The		×
	neral ipting • Syntax highlighting	 Show start screet Show documenta Auto save project Color Theme: Gen Working directory Color Theme: Gen Significant digit 	en at startup ation page after start et on model run eric v ::\Users\vaj_pn\AppDat at s: 5 + 1.	ta\Local\Temp\DeltaShell_Working_Directory Sample 0012346 2346 2346 2346 2346 2346 2346E+07	
				ОК	Cancel

Figure 3.31: The User Interface options dialog.

3.4.3 Home

The second *ribbon* is the *Home* ribbon (Figure 3.32). It harbours some general features for clipboard actions, addition of items, running models, finding items within projects or views, and help functionality.





3.4.4 View

The third ribbon is the View ribbon (Figure 3.33). Here, the user can show or hide windows.

÷	File	H	lome	View		Tools	N	1ap	Sp	oatial Op	erations
_	Peret lavout		🄑 Prop	erties		🔛 Chart Cont	ents	X	Region C	ontents	
			🖵 Messages		👫 Map Contents		Project Explorer				
	After Rest	tart	🏝 Time	Navigato	or	C Operations	5	â	Toolbox		
	Layout					Show/Hid	e				

Figure 3.33: The View ribbon.

3.4.5 Tools

The fourth *ribbon* is the *Tools* ribbon (Figure 3.34). By default, it contains only the *Open Case Analysis View* tool. Some model plug-ins offer the installation of extra tools that may be installed. These are documented within the user documentation of those model plug-ins.



Figure 3.34: The Tools ribbon contains just the Data item.

3.4.6 Map

The last *ribbon* is the *Map* ribbon (Figure 3.35).

File	Home Vie	ew Tools	Map	Spatial Operations						
民 118 🖳	North Arrow	🔉 🔀 🗲 Zoon	n Previous	🚼 Map Coordinate System	XX	🐹 Show Profile	📥 坐 🔪 🗋	🗎 Link	🚊 Add Load	🖉 Set Label
1	📃 Legend	🚯 🔍 🤿 Zoon	n Next	T Export As Image	٩,		👝 😣 🔔 🚅		Add Observation Point	Overwrite Label
₩ <i>2</i> ,	15 Scale Bar	🖑 🛋 🚺 Quer	y Features	Query Time Series	۲		👁 🔺 🎘 🖱		🔎 Find Grid Cell	
FM Region	Decorations		То	ols	Edit	Grid Profile	Area	Region	Water Quality	Spatial Operations

Figure 3.35: The Map ribbon.

This will be used heavily, while it harbours all Geospatial functions, like:

- ♦ Decorations for the map
 - North arrow
 - Scale bar
 - Legend
 - ...
- ♦ Tools to customize the map view
 - Select a single item
 - Select multiple items by drawing a curve
 - Pan
 - Zoom to Extents
 - Zoom by drawing a rectangle
 - Zoom to Measure distance
 - □ ...
- ♦ *Edit* polygons, for example within a network, basin, or waterbody
 - Move geometry point(s)
 - Add geometry point(s)
 - Remove geometry point(s)
- ♦ Addition of Area elements, such as
 - Add 2D Cross-section
 - Add 2D Structure
 - Add 2D Pump
 - □ ...

Still, all functions of the category can be activated as they will appear in the drop-down panel.

3.4.7 Scripting

When you open the scripting editor in the User Interface, a Scripting *ribbon* category will appear. This ribbon has the following additional options (see also Figure 8.11), which are described in Table 8.1:

File	Home	View	Tools Scripting						
Run C Script	lear cached variables	Show spaces Show tabs Show end of lir	Python private variables Highlight current line Word wrap	Text color • Background color • Font size 14	Local variables	 Insert spaces Keep tabs Tab size 	⊡= Convert to space indenting □= Convert to tab indenting □= Convert to tab indenting □ Gomment selection	Python	✓ Save before run
R	un		View				Editing	Docum	Settings

Figure 3.36: The scripting ribbon within the User Interface.

Table 3.1: Functions and t	eir descriptions	within the s	s <i>cripting</i> ribbon	of the User	Inter-
face.					

Function	Description
Run script	Executes the selected text. If no text is selected then it will execute the entire script
Clear cached variables	Clears all variables and loaded libraries from memory

Function	Description
Debugging	Enables/Disables the debug option. When enabled you can add breakpoint to the code (using $F9$ or clicking in the mar- gin) and the code will stop at this point before executing the statement (use F10 (step over) or F11 (step into) for a more step by step approach)
Python variables	Show or hide python variables (like _var_) in code comple- tion
Insert spaces/tabs	Determines if spaces or tab characters are added when pressing tab
Tab size	Sets the number of spaces that are considered equal to a tab character
Save before run	Saves the changes to the file before running
Create region	Creates a new region surrounding the selected text
Comment selection	Comments out the selected text
Convert to space indenting	Converts all tab characters in the script to spaces. The num- ber of spaces is determined by Tab size
Convert to tab indenting	Converts all x number of space characters (determined by Tab size) in the script to tabs
Python (documentation)	Opens a link to the python website, showing you the python syntax and standard libraries

Table 3.1: Functions and their descriptions within the scripting ribbon of the User Interface.

3.4.8 Shortcuts

The shortcut keys of the scripting editor within the User Interface are documented in Table 8.2.

Shortcut	Function
Ctrl + Enter	Run selection (or entire script with no selection)
Ctrl + Shift +	Run current region (region where the cursor is in)
Enter	
Ctrl + X	Cut selection
Ctrl + C	Copy selection
Ctrl + V	Paste selection
Ctrl + S	Save script
Ctrl + -	Collapse all regions
Ctrl + +	Expand all regions
Ctrl + "	Comment or Uncomment current selection
Ctrl + W	Add selection as watch variable
Ctrl + H	Highlight current selection in script (press esc to cancel)
F 9	Add/remove breakpoint (In debug mode only)

Table 3.2: Shortcut keys within the scripting editor of the User Interface.

Shortcut	Function
F5	Continue running (In debug mode only — when on breakpoint)
Shift + F5	Stop running (In debug mode only — when on breakpoint)
F10	Step over current line and break on next line (In debug mode only - when on breakpoint)
F11	Step into current line if possible, otherwise go to next line (In debug mode only — when on breakpoint). This is used to debug functions declared in the same script (that have already runned)

Table 3.2: Shortcut ke	evs within the scriptin	na editor of the Us	ser Interface.
	<i>yo waann aro oonpan</i>	ig builde of the be	for micoriado.

3.4.9 Quick access toolbar

Note: The user can make frequently used functions available by a single mouse-click in the *Quick Access Toolbar*, the top-most part of the application-window. Do this by right-mouse-clicking a ribbon item and selecting *Add to Quick Access Toolbar*.



4 Case management

Delta Shell currently offers two ways of case management: through the *Project* window or by using the Python scripting functionality (which is described in more detail in chapter 8).

4.1 Case management within the *Project* window

Within a Delta Shell project multiple models can be run and combined. For example multiple runs from a single model, or a model run for a flow model combined with a RTC model or a water quality model. A project may contain different Delta Shell models (see also Figure 4.1), for example one or multiple flow models and a rainfall-runoff model. Model features of one model can be copied or linked to a second model in the project. Complete networks can also be copied or linked. In this way it is easy to create several models with the same basic characteristics, but small differences. In this way the impact of different possible measures on the water system can be analyzed.

Results from different models can be viewed next to each other by docking maps and graphs. Results of different models can also be combined in one single map or graph.

To enhance transparency the **Project** can be sorted by adding folders and moving networks, models and data. It is advised to group several models using one network in a folder to separate them from other models. Figure 4.1 gives an example of a case management in Delta Shell. In this project an original model was used for a sensitivity study and a study for different measures (in this case weirs with or without RTC). The original network was used in the models for the sensitivity study and only the roughness was adjusted. Then, for the measures a new network ("measure1:weirs") was created which was used to model the situation with and without RTC.



Figure 4.1: Example of case management

4.2 Case management using Python scripting

The Python scripting toolbox as provided within the Delta Shell framework can be used for a variety of purposes. One of these is case management. It may be used for sensitivity analyses, batch runs, urban development studies etc.

An example of using Python scripting for case management is presented below. This example script takes a base case existing in the Delta Shell project, then varies the boundary condition on a specific location, runs the model using the new boundary condition, and finally saves the results in a new Delta Shell project.

```
flow = CurrentProject.RootFolder["Base case"]
# Case 1: Batch run
# Changing constant boundaries and save them to completely new projects
constantBoundaries = [1.5, 1.75, 2.0]
```

for x in constantBoundaries: EditConstantBoundaryCondition(flow, "benedenrand_ZwarteWater", x) Application.RunActivity(flow) timeseries = GetWaterLevelResultsAtLateralSource(flow, "Zwolle") Gui.CommandHandler.OpenView(timeseries) projectName = ComposeProjectName("SW_max", x) Application.SaveProjectAs(projectName)

SOMETHING ABOUT CASE MANAGEMENT AND PYTHON SCRIPTING

5 Working with the map

5.1 Introduction

By default all geo-referenced data will be presented in one and the same (tab in the) main window: the Central Map.

It is also possible to use an extra map. The user can add a map to the **Project** by a rightmouse-click on <project1/Add/New item> or a double-click in the **Toolbox** window on *Items/-General/Map*. This map can be customized by the user by dragging individual model features (such as the computational grid, initial conditions, output results etc.) onto the map.



💱 🗲 Zoom Previous i 🔍 🔿 Zoom Next

With the left-most part of the *Map* ribbon Decorations, the user can show or hide the *Scale* bar, *Legend* and *North Arrow*.

The user can zoom and pan by activating one of the buttons 2 Course Cour

By selecting ¹/_m distances between two points on the map are shown.

5.2 Map layers

Each layer can be made (in)visible by (un)checking it in the **Map** window. This can be done for the network as a whole for example, but also for the layers within the network, such as *Nodes* or *Weirs*.

The order in which layers are presented in the map is always top-down. The user can adjust this by a right-mouse-click on a layer in **Map** window and selecting for example *Bring to front*.

With the four icons in the top left of the window $\stackrel{\texttt{F}}{=} \stackrel{\texttt{F}}{=} \stackrel{\texttt{F}}{=} \text{new} <.shp>- or <.wms>-layers can be added, removed or exported.}$

By right-mouse-click on a layer in the **Map** window, the following functions are available within the context menu:

- ♦ Open Attribute Table (of GIS-shape)
- ♦ Cut, Copy, Delete, Rename
- ♦ Remove Theme
- ♦ Import legend
- ♦ Export legend
- ♦ Order
 - □ Bring to front
 - Send to back
 - Bring forward
 - Send backward
- ♦ Zoom to Extend
- Synchronize zoom level with map

- ♦ Show labels
- ♦ Show in Legend
- ♦ Hide all Layers but this one
- ♦ Properties

By selecting *Properties* ... a window pops up in which symbols, sizes, colors etc. can be adjusted to preference, see also Figure 5.1.

Style	Layer information	Labels					
1-3	Single Symbol	Genera	ate theme				
1 2 3	Category	Attribu	te: water	r depth	•		
1	Gradient	Co	olors:		•	Classes: 12	
3 0-1 1-2	Quantity	🗸 Siz	ze: Min:	12 💂	Max: 25	Gene	rate
2-3		Cu	istom range:	0.00	to 0.00		
		Them	e items				
			Symbol	Value		Label	
		•	•	0		0	
				0		0	
		4 St	yle				
		▲ Str Co	yle lor		25	5.0.0	
		▲ Sty Col Ou Ou	yle lor tline color tline dash stv	le	■ 255 ■ 0,0	5. 0. 0 0. 0	
		Sty Co Ou Ou Ou	yle lor tline color tline dash sty tline width	le	■ 255 ■ 0.0 ■ So 1	5. 0. 0 0. 0 lid	
		▲ Sty Col Ou Ou Poi	yle lor tline color tline dash sty tline width int shape	le	255 ■ 0, (= So 1 Ellipse	5. 0. 0 0. 0 lid	
		Style S	yle lor tline color tline dash sty tline width int shape ape size ow outline	le	■ 255 ■ 0.0 ■ So 1 Ellipse 12 True	5. 0. 0 0. 0 lid	
		▲ Sty Co Ou Ou Po Sh Sh	yle lor tline color tline dash sty tline width int shape ape size ow outline mbol	le	255 ■ 0,0 = 0,0 1 Ellipse 12 True ■ Sy	5, 0, 0), 0 lid stem.Drawing.Bitmap	

Figure 5.1: The layer Properties editor.

By selecting *Open Attribute Table*, a table-view will emerge below the Central Map. With the help of filters the user can search for model objects and edit attribute data. Below the table are buttons to add or delete objects. The table supports the following context menu actions:

- ♦ Copy
- ♦ Paste
- ♦ Delete
- ♦ Add attribute
- ♦ Zoom to item
- ♦ Open View ...

Selecting *Open View*... will present the editor in a different tab in the main window. Doubleclicking on an item in the **Project** window which is not geo-referenced, will also open the editor.

5.3 Background layer

For modelling and presentation a map in the background is very useful, for example a <.shp>-file or <.wms>-layer.

As an example, to set OpenStreetMap as background, the user must:

- right-mouse-click on <project1> in the **Project** window, and select "Add" and "New Item ..."
- ♦ select General and Map(world)
- and finally right-mouse-click on the map in the **Project** window, and select Use as default background layer

Note: Mark that automatically the coordination system for the map is set on <WGS 84 / Pseudo-Mercator>. More on coordinate systems in the next paragraph (section 5.4).

5.4 Coordinate systems and transformations

The map and all spatial objects in your model, like a network, basin or waterbody, may have a coordinate system associated with it. When presenting the objects in the map, Delta Shell performs the appropriate coordinate transformation (from the coordinate system of the layer to the coordinate system of the map).

The user can specify or adjust the coordinate system for the map by selecting ^{SMAP Coordinate System} in the *Map* ribbon or by a right-mouse-click on *map* in the **Map** window and selecting *Change Map Coordinate System*

The user can also specify or adjust the coordinate system for model objects by selecting the object in the **Project** window and select *CoordinateSystem* in the **Properties** window, as in Figure 5.2.



Figure 5.2: Specify the coordinate system for a network.

For both ways, the following window will pop up:

amersfoort PRO.J.4. +projesteere altg.0=2521661065555555 +lon_0=53875388888889.+0.9990797.4, 0=155000 +y_0=463000 +elipesp-bessel +towgsd4=656.417.50.3319.4651 0.398957.0.343988.1.8774.0725 + units=m +no_defs WKT: PRO.JCS[^Amersfoort] /RD New [EPSG.28992] Amersfoot/ RD New [EPSG.28992] Amersfoot/ RD New [EPSG.28992] BY HEROID[************************************		
AUTHORITYEFSG".9001"]]. AXIS["X":EAST]: AXIS["Y":NORTH]: AUTHORITY["EPSG".28992"]]	amersfoort Geographic Coordinate Systems Projected Coordinate Systems Amersfoort / RD Old [EPSG.22991] Amersfoort / RD New [EPSG.28992]	PROJ.4. + proj-sterea + alc. 0-52. 1561 (605555555 +ton. 0-53 3763888888 +40 - 999079 × 0-155000 +v_0-453000 + ellips-bessel + towg84+555.417.50.313,45552. 0.399957.0.34388.1 8774.4 0725 + units=m + no_defs WKT: PROJ.CS[*Amerstoort, DATUM[*Amerstoort, DATUM[*Amerstoort, SPHEROID[*Bessel 141*6377397.155.299.1528128, AUTHORITY[*EPSG*7004*]], TOWG84[656.417.50.3319.465.552.0.389957.0.343988. 18774.40725] AUTHORITY[*EPSG**2091]], PRIMEM[*Greenwach*0. AUTHORITY[*EPSG**2091]], AUTHORITY[*EPSG**2091]], AUTHORITY[*EPSG**2091]], AUTHORITY[*EPSG**2091]], PROJECTION[*Obios_State_37], AUTHORITY[*EPSG**2091]], PRAME TER[*atitude_0_forgephic*], PARAME TER[*atitude_0_forg93078], PARAME TE

Figure 5.3: Choose coordinate system.

Here, the user can select from a very extensive list of coordinate systems. Delta Shell works with open source software provided by the Open Source Geospatial Foundation (http://www.gdal.org/).

By typing in the selection-box the list will be filtered and selection made easier. In the box on the right hand side of the window characteristics of the coordinate system are presented.

6 Spatial editor

6.1 Introduction

The spatial editor is a generic feature of the User Interface for editing spatial data (e.g. bed level, roughness, viscosity, initial conditions, sediment availability). The spatial editor supports both point clouds and coverages (e.g. data on a grid or network). Therefore, you can use the spatial editor both to edit spatial data in general and to prepare model input. This Chapter describes the general features of the spatial editor (section 6.2), (spatial) quantity selection (section 6.3), geometry operations (section 6.4), spatial operations (section 6.5) and the operation stack (section 6.6). The examples given below are typically focusing on use of the spatial editor in the D-Flow FM User Manual.



Figure 6.1: Overview of spatial editor functionality in Map ribbon

6.2 General

6.2.1 Overview of spatial editor

The spatial editor functionality can be accessed from the "Spatial Operations" ribbon (Figure 6.1). Typically, you first select which data set or quantity (either a point cloud or a coverage) to work on (e.g. bed level, roughness, viscosity, initial conditions, sediment availability), then a geometry (e.g. point, line, polygon) and finally which spatial operation to perform (e.g. crop, delete, set value, contour, interpolate, smoothing). Typical workflows are as follows:

Working on a point cloud dataset:

- 1 Import the dataset as point cloud (section 6.2.2)
- 2 Activate/select the dataset (quantity) in the spatial editor (section 6.3)
- 3 Select a geometry to perform an operation on (section 6.4)
- 4 Select the spatial operation for this geometry (section 6.5)
- 5 Repeat steps 3 and 4 until you are satisfied with the data
- 6 Export the dataset (section 6.6.7)

Working on a coverage (e.g. model input):

- 1 Activate/select the spatial quantity to work on in the spatial editor (section 6.3)
- 2 Optional: import a dataset as point cloud (section 6.5.1)
- 3 Select a geometry to perform an operation on (section 6.4)
- 4 Select the spatial operation for this geometry (section 6.5)
- 5 Repeat steps 3 and 4 until you are satisfied with the data
- 6 Interpolate the point cloud to the grid or network (section 6.5.10)
- 7 Optional: export the dataset (section 6.6.7)

Upon performing a spatial operation, the 'Operations' panel will open (see Figure 6.53) with the operations stack. This stack keeps track of the workflow of spatial operations that you performed. This helps you to make transparent how you arrived at your 'final' dataset without

having to save all the intermediate datasets (steps) separately. Moreover, the stack is reproducible and easily editable without having to start all over again. When working on a coverage (e.g. the second workflow), point clouds can be used to construct the coverage. In this case the coverage (for example 'Bed level') is the 'trunk' of the workflow and the point clouds (appearing as sets in the stack) are 'branches' or subsets of this trunk (see Figure 6.53). By selecting the set or coverage in the 'Operations' panel you determine on which dataset you are working. The interpolate operation (section 6.5.10) allows you to bring data from the point cloud (branch) to the coverage (trunk). For more information on the stack you are referred to section section 6.6.

6.2.2 Import/export dataset

To import a (point cloud) dataset use the context menu on "project" in the "Project Tree", select "import" from the context menu and select the option "Points from XYZ-file" (Figure 6.2). There is yet another method to import the point cloud. Click on "Import" icon under the Home ribbon to obtain a drop-down menu with the list of importers. Select the option "Points from XYZ-file" under Spatial data section (Figure 6.3) to launch the import wizard.

After the import, the point cloud will be added to the project tree. To activate the point cloud in the spatial editor, either double click the dataset in the project tree (Figure 6.4) or select it from the drop down box in the spatial editor ribbon (Figure 6.5).

pe:				
Data				
🏟 Time Series (*.csv)				
Water Quality				
Hydrodynamics (*.hyd)				
D-Flow FM 2D/3D				
🗑 Flexible Mesh His File	🚼 Flexible Mesh Map File	👿 Flow Flexible Mesh Model	😥 Unstructured Grid (UGRID)	
Wave Output (WAVM)	C Waves Model			
Spatial data				
DF	🏐 NetCDF Regular 2D Grid	Points from XYZ-file	🏐 Raster File	
Time-Dependent Grid				
Data Import				
🐇 Model features from GIS				

Note: Exporter still to be implemented

Figure 6.2: Importing a point cloud into the project using the context menu on "project" in the project tree

Data									
8	Time Series (*.csv)								
Wate	r Quality								
*	Hydrodynamics (*.hyd)								
D-Flo	w FM 2D/3D								
*	Flow Flexible Mesh Model	*	Flexible Mesh Map File	*	Flexible Mesh His File	*	Unstructured Grid (UGRID)	*	Waves Model
*	Wave Output (WAVM)								
Spati	al data					_			
	NetCDF Regular 2D Grid		Raster File	Ę	Time-Dependent Grid		Points from XYZ-file		IDF
Data	Import								
*	Model features from GIS								
									.:

Figure 6.3: Importing a point cloud into the project using the "Import" drop-down menu in the Spatial Operations ribbon



Figure 6.4: Activate the imported point cloud in the spatial editor by double clicking it in the project tree



Figure 6.5: Activate the imported point cloud in the spatial editor by selecting it from the dropdown box in the Map ribbon

6.2.3 Activate (spatial) model quantity

Similar to activating an imported dataset in the spatial editor, you can also activate a (spatial) model quantity (e.g. bed level, initial conditions, roughness, viscosity) in the spatial editor by double clicking the quantity in the project tree or selecting it from the dropdown box in the "Spatial Operations" ribbon.

6.2.4 Colorscale

In the spatial editor the colorscale for a spatial quantity can be made visible in the map window by clicking on the "Legend" button in the "Spatial Operations" ribbon (Figure 6.6). Then, in the Map window a colorscale is activated (Figure 6.7). You can (de-)activate the color scale by clicking on the "Legend" button again. By default the colorscale is ranging from the minimum to the maximum value of the active dataset.



Figure 6.7: Activate the colorscale by using the Legend in the map ribbon and the colorscale will become visible in the map window.

You can adjust the colormap and classes of the colorscale using the context menu on the spatial quantity in the "Map tree" and selecting "Legend Properties" (Figure 6.8 left panel). A layer properties editor will open in which you can set the colormap to your own preferences (Figure 6.8 right panel).

Man 👻 🏾 X	Properties	– 🗆 ×
年 國 (2) 今	Theme Labels Information	
▲ ⊕ Map	Theme type	v
🔺 🗹 har		
🕨 🗹 Area	Feature attribute value	
Boundary Conditions	value	
✓ Boundaries	Color scale	v.
✓ Sources and Sinks	Number of classes	12
Estimated Grid-snapped features		12 🗸
✓ Unstructured Grid	 Auto scale 	
Initial Water Level	O Custom range min : 0 max : 0	Determine
Roughness		
Viscosity		
Diffusivity		Generate
Initial Salinity	Value Min Max Symbol Label	
Bed Lev Bed Lev Show attribute table	4.166 4.166	
4.16 4.16	2.732 0.732	
2.731 Zoom to extends	1.298 🥥 1.298	
Legend Properties	-0.1356 -0.1356	
-0.13 Import Legend	-1.569 -1.569	
-1.0 Export Legend	-4.437 -4.437	
	-5.87 -5.87	
-5.87 Order +	-7.304	
-7.30 🚓 Export laver	-8.738 -8.738	
-8.73 Delete laver	-10.17	
-10.17		
-11.61		
	Ok Ca	incel Apply

Figure 6.8: Edit the colorscale properties using the context menu on the active layer in the Map Tree

6.2.5 Render mode

By default point clouds are rendered as (colored) points and coverages as shades (e.g. 'FillSmooth'). The render mode can be edited using the properties of the active layer Figure 6.9. The User Interface offers the following render modes:

- ♦ Points
- ♦ Lines (only for coverages)
- Shades or 'FillSmooth' (only for coverages)
- ♦ Colored numbers
- ♦ Mono colored numbers

An example of a coverage rendered as colored numbers is given in Figure 6.10.

Prop	perties	₩ ф)
	Grid layer		
•	A ↓		
~	Coordinates		
	Map coordinate syste	<empty></empty>	
	Layer contents coordi	Amersfoort / RD New	
¥	General		
	Opacity	0.8	
\sim	Misc		
	RenderMode	FillSmooth	1
~	Rendering	Point	
	Render asynchronous	Line	
	Render technology	FillSmooth	
	Optimize Rendering	ColoredNumbers	
		MonoNumbers	
		L	
Re	nderMode		

Figure 6.9: Select the rendermode for the active layer in the property grid.



Figure 6.10: Example of a coverage rendered as colored numbers.

6.2.6 Context menu

In addition to the selection of spatial operation from the 'Spatial Operations' ribbon (see section 6.5), you can also select spatial operations using the context menu (e.g. context menu). After drawing a geometry and clicking the context menu all spatial operation available for the geometry will pop-up (see Figure 6.11). The spatial operation will become active upon selecting it from the context menu.



Figure 6.11: Selecting a smoothing operation for a polygon geometry from the context menu (using context menu)

6.3 Quantity selection

A spatial quantity can be activated/selected either by double clicking it in the project tree (Figure 6.12) or by selecting it from the dropdown box in the "Spatial Operations" ribbon (Figure 6.13). Upon selecting the spatial quantity it will be shown as a point cloud (for a dataset) or coverage (for model input) on the central map. Typically, you start from a point cloud (either obtained from import or by generating samples yourself) which will eventually be interpolated to a grid or network (e.g. coverage). The spatial editor will keep track of both the changes made to the point cloud(s) and coverage of the selected spatial quantity. The information will be saved in the User Interface project and available the next time you open the project. **Note:** The operations are not yet saved in a human-readable/editable file



Figure 6.12: Activating a spatial quantity by double clicking it in the project tree (in this example 'Initial Water Level')

Initial Water Level	-
Initial Water Level	
Roughness	5
Viscosity	
Initial Salinity	
Bathymetry	

Figure 6.13: Activating a spatial quantity by selecting it from the dropdown box in the 'Spatial Operations' ribbon

6.4 Geometry operations

The spatial editor supports three types of geometry operations: (1) polygons, (2) lines and (3) points (see also Figure 6.14). The following sub-sections subsequently describe how these geometries can be selected and edited. If you do not select any of these three geometry operations, the spatial operation automatically applies to all the data.

Note: Please note that the drawn geometries are not yet persistent, implying that the geometries once drawn cannot be edited yet. Upon pressing the "Esc" button while in editing mode all drawn geometries will disappear.



Figure 6.14: Overview of the available geometry operations in the 'Spatial Operations' ribbon

6.4.1 Polygons

Upon selecting "Polygon" from the "Map" ribbon you can draw one or multiple polygons (Figure 6.15). Each polygon point is defined by a single click with the LMB. The polygon is closed by double clicking the LMB. After drawing the (first) polygon, the available spatial operations for polygons are enabled in the "Spatial Operations" ribbon. The following spatial operations are available for polygons:

- ♦ Crop (section 6.5.2)
- ♦ Delete (section 6.5.3)
- ♦ Set Value (section 6.5.4)
- ♦ Contour (section 6.5.5)
- ♦ Gradient (section 6.5.9)
- ♦ Smoothing (section 6.5.11)
- ♦ Interpolate only in case samples and a grid/network are available (section 6.5.10)
- ♦ Copy to samples (section 6.5.6)
- ♦ Copy to spatial data (section 6.5.7) only for grid coverages

Мар Home View Tools Spatial Operations DIMR File TImport 🕒 Set Value 🖉 Interpolate -Copy to samples 🛛 🖑 Copy to spatial data Polygon Ξ Layer Initial Water Level 🐣 (日 Line 🕂 Crop 🖉 Contour 🥒 Smoothing 1. Merge spatial data Legend Merge Points ◆ Delete III Gradient 📅 Overwrite Value Operations Add Spatial Operations Decora. Edit Project 🔵 har 🗙 • 🛛 🗙 4 🖃 📆 Harlingen 🗄 - ठ har 🗈 General 🗄 😽 Area 🕞 Grid 🗑 Bed Leve 🕀 Time Frame 📲 Processes 🗃 Initial Conditions 🗄 🕞 Boundary Conditions 🖶 🚽 Physical Parameters Sources and Sinks Numerical Parameters Output Parameters 🗄 🕞 Output

The selected spatial operation applies to all the drawn polygons.

Figure 6.15: Activating the polygon operation and drawing polygons in the central map.

6.4.2 Lines

Upon selecting "Line" from the "Map" ribbon you can draw one or multiple lines (Figure 6.16). Each line point is defined by a single click with the LMB. The line is completed by double clicking the LMB. After drawing the (first) line, the available spatial operations for lines are enabled in the "Map" ribbon. The following spatial operations are available for lines:

- ♦ Contour (section 6.5.5)
- ♦ Copy to samples (section 6.5.6)
- ♦ Copy to spatial data (section 6.5.7) only for grid coverages

The selected spatial operation applies to all the drawn lines.



Figure 6.16: Activating the line operation and drawing lines in the central map.

6.4.3 Points

Upon selecting "Add points" from the "Map" ribbon you can draw one or multiple points and assign a uniform value to them (Figure 6.17). Each point is defined by a single click with the LMB. The group of points is completed by double clicking the LMB. Upon double clicking a popup appears in which you can assign a value to the points.



Figure 6.17: Activating the 'Add points' operation, drawing them in the central map and assigning a value to them.

6.5 Spatial operations

The spatial editor supports the following spatial operations (see also Figure 6.18):

♦ Import (section 6.5.1) - only for point clouds

- ♦ Crop (section 6.5.2)
- ♦ Delete (section 6.5.3)
- \diamond Set Value (section 6.5.4)
- ♦ Contour (section 6.5.5) only for point clouds
- ♦ Gradient (section 6.5.9)
- ♦ Interpolate (section 6.5.10) only for point clouds
- ♦ Smoothing (section 6.5.11)
- ♦ Change single value (section 6.5.12) only for grid coverages
- ♦ Merge spatial data (section 6.5.8) only for grid coverages
- ♦ Copy to samples (section 6.5.6) only for grid coverages
- ♦ Copy to spatial data (section 6.5.7) only for grid coverages



Figure 6.18: Overview of the available spatial operations in the 'Spatial Operations' ribbon

The sections below provide a description of each operation.

6.5.1 Import point cloud

With the import operation you can import a point cloud for the selected spatial quantity. For this operation no geometry is required. The import operation is activated from the 'Spatial Operations' ribbon (Figure 6.19). Upon importing a point cloud you are asked whether a coordinate transformation should be applied to the imported dataset (Figure 6.20). By default it will be assumed that the imported data is in the same coordinate system as the model. If not, you can indicate from which to which coordinate system the data should be transformed. After import the point cloud is added to the operations stack (Figure 6.21). The difference between this importer and importing a point cloud on the project level in the project tree (section 6.2.2) is that for this importer the point cloud is directly assigned to the selected spatial quantity (e.g. model input) instead of being treated as a separate dataset.



Spatial Operations

Figure 6.19: Importing a point cloud using the 'Import' operation from the 'Spatial Operations' ribbon

Apply coordinate transformation on da —							
 Import without transformation (as-is) 							
O Transform from:	Amersfoort / RD New						
to:	Amersfoort / RD New						
	ОК	Cano	;el				

Figure 6.20: Option to perform a coordinate transformation on the imported point cloud



Figure 6.21: Appearance of import point cloud operation in the operations stack

6.5.2 Crop

The crop operation crops a point cloud or coverage (depending on which one is active). The crop operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries. You can control which part of the data should be deleted by using polygons. If you provide a polygon outside the point cloud or coverage, all data will be deleted. For an example see Figure 6.22. After cropping (part of) the point cloud of coverage the operation is added to the operations stack (Figure 6.23).



Figure 6.22: Performing a crop operation on a point cloud with a polygon using 'Crop' from the 'Spatial Operations' ribbon



Figure 6.23: Appearance of crop operation in the operations stack

6.5.3 Delete

The delete operation deletes (part of) a point cloud or coverage (depending on which one is active). The delete operation is activated from the 'Spatial Operations' ribbon. You can control which part of the data should be deleted by using polygons. If no polygons are provided, the total dataset will be deleted. For an example see Figure 6.24. After erasing (part of) the point cloud or coverage the operation is added to the operations stack (Figure 6.25).



Figure 6.24: Performing an delete operation on a point cloud with a polygon using 'Delete' from the 'Spatial Operations' ribbon

Operations	- ↓ ×
🍖 Refresh 🔊 🔎 🗮 🗙 🔛 📄	
Bed Level Bed Level Bed Level Bed Level Frase 1 Image: Second Seco	
I	

Figure 6.25: Appearance of delete operation in the operations stack

6.5.4 Set value

The set value operation assigns a value to a point cloud or coverage (depending on which one is active). The set value operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries or for the total data set if no polygon is provided. By assigning a value, the user can choose from the following operations:

- Overwrite : overwrites all existing points within the polygon (excluding no data values) with the uniform value
- Overwrite where missing (only for coverages) : overwrites all missing values within the polygon with the uniform value
- Add : Adds the uniform value to all existing points within the polygon (excluding no data values)
- Subtract : Subtracts the uniform value from all existing points within the polygon (excluding no data values)
- Multiply : Multiplies all existing points within the polygon (excluding no data values) with the uniform value
- Divide : Divides all existing points within the polygon (excluding no data values) by the uniform value
- Maximum : Sets all existing points within the polygon (excluding no data values) to the maximum of its current value and the uniform value
- ♦ Minimum : Sets all existing points within the polygon (excluding no data values) to the minimum of its current value and the uniform value

For an example see Figure 6.26. After performing a set value operation to (part of) the point cloud or coverage the operation is added to the operations stack (Figure 6.27).



Figure 6.26: Performing a set value operation (e.g. overwrite) on a point cloud with a polygon using 'Set Value' from the 'Spatial Operations' ribbon



Figure 6.27: Appearance of set value operation in the operations stack

6.5.5 Contour

The contour operation creates a point cloud with a uniform value along a line or polygon (depending on which one is active). The contour operation is activated from the 'Spatial Operations' ribbon. After drawing the lines or polygons you have to assign the uniform value (argument) and the sampling interval in m. This spatial operation can be useful to digitalize information from nautical charts for example. In this case you first have to import the nautical chart as a geotiff (Figure 6.28), set the right map coordinate system (Figure 6.29) and then use the contour operation Figure 6.30. Sometimes the samples are created behind the geotiff. Then you can use the context menu on the samples layer in the Map tree to bring the samples to the front (Figure 6.31). After applying the contour operation it is added to the stack (Figure 6.32).

Project	- ↓ × (Samples.xyz 🗙
∎ . ⊡ ² 73) Open	
	Organize Vew folder	Contwikkeling > D-Flow-FM_GUI > 20140820_Demo > • • + + Search 20140820_Demo > >
Map S	 J Music E Pictures E Subversion E Videos I Videos 	test.dsproj_data 1969_2008_mixed _data_medium_s cale_003_modifie _d.tif
	GS (C:)	
	homes (\\home *	
	File nam	e: 1969_2008_mixed_data_medium_scale_003_modified.tif GeoTIFF (*.tif) Cancel

Figure 6.28: Import a nautical chart as a georeferenced tiff file



Figure 6.29: Set the right map coordinate system for the geotiff



Figure 6.30: Performing a contour operation on a nautical chart using lines to define the contours and 'Contour' from the 'Spatial Operations' ribbon

	ĚĚ	Open Attribute Table	10	134 M. F.
Мар	do	Cut		2 Ans
Z X Z Z		Сору	13	20,50
🖃 🥘 map	×	Delete		1000
🔽 1969		Rename		ELEL D 18e
testo	12	Remove Theme	L'	TELD 1
	ŝ	Import legend	EN	T AREA FL5s
		Export legend	Ľ.	183 A Fairway
	þ	Order +	đ	Bring to front
	15 JR 12 SI	Zoom to Extent		Send to back
	000	Synchronize zoom level with map		Bring forward
٠ III		Show labels		Send backward
Map Data	~	Show in Legend	-	
Messages Time		Hide all Layers but this one		
	r	Properties		

Figure 6.31: Bring the sample set to the front if it appears behind the nautical chart

⊡ 🗊 test.xyz
···· 🔲 test.xyz
🗄 🖋 Contour 1
🕅 💕 input
🕅 💕 mask
📖 🔽 🧲 output

Figure 6.32: Appearance of contour operation in the operations stack

6.5.6 Copy to samples

This operation converts the currently selected grid coverage to a sample set, which becomes that starting point of a new subset. If polygons have been drawn, the operation will confine the copy to the interiors. The operation will not convert missing values to samples. Note that the operation does keep a reference to the original copied grid coverage; if it is changes by a re-evaluation of the stack, the changes will affect the output point cloud.

File	Home View	Tools	Map	Spatial Operations DIMR	
	Layer Bed Level Y	Polygon Line	{ ⊟	∑Import	
Legend		Points	Operations	🔹 Delete 📲 Gradient 🔓 Overwrite Value Copies values and locations within polygons to samples	
Decora	Edit	Add		Spatial Operations	

Figure 6.33: Applying the copy to samples operation



Figure 6.34: Copy to samples operation result

6.5.7 Copy to spatial data

This operation simply clones the grid coverage, starting a new subset with a snapshot of the currently selected operation output. Similarly to 6.5.6, it keeps a reference to the input data and will perform the clone again upon re-evaluation.



Figure 6.35: Applying the copy spatial data operation



Figure 6.36: Copy spatial data operation result

6.5.8 Merge spatial data

Whenever a subset contains a grid coverage as its editing data (after applying e.g. 6.5.7) on the same grid as the main operation set, its result can be combined with the main set by applying this operation, similarly to the interpolation operation for sample sets, discussed below. Combining the grid coverages can be achieved with the usual point-wise methods.



Figure 6.37: Activating the merge spatial data tool from the ribbon



Figure 6.38: The merge operation requests a pointwise combination method



Figure 6.39: Resulting grid coverage

6.5.9 Gradient

The gradient operation applies a gradient to a point cloud or grid coverage (depending on which one is active). The gradient operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries or for the total data set if no polygon is provided. You have to assign the initial (start) value, the final (end) value and the going to angle (according to the Cartesian convention with 0 degrees is East, 90 degrees is North, etc **Note:** This is not working properly yet). For an example see Figure 6.40. After applying a gradient to (part of) the point cloud or coverage the operation is added to the operations stack (Figure 6.41).



Figure 6.40: Performing a gradient operation on a point cloud with a polygon using 'Gradient' from the 'Spatial Operations' ribbon

|★


Figure 6.41: Appearance of gradient operation in the operations stack

6.5.10 Interpolate

The interpolate operation is the way to get sample set(s) to a grid or network (e.g. coverage). In the operation stack this means that we are actively switching from working on a point cloud (helping to construct the coverage) to working on the selected coverage (or spatial quantity). The interpolation is performed on the data within a polygon or polygons (if provided) or all the data (if no polygons are provided). The interpolate operation can be performed on a single (selected) sample set or on multiple sample sets. Both methods are discussed below. The methods for interpolation are either

- Triangulation: performs a Delauney triangulation on the sample point set before projecting onto the grid.
- Averaging: combines sample points within a possible enlarged cell according to an algorithm of choice. The user can set the search cell expansion factor (rel. search cell size) and a threshold for the number of sample points within a cell (minimum sample points), see Figure 6.42.

Interpolation operation	>	×
Please specify the operati	ion parameters	
Pointwise operation	Overwrite where miss $ \smallsetminus $	
Interpolation method	Averaging \sim	
Cell averaging algorithm	SimpleAveraging \sim	
Minimum sample points	1	
Rel. search cell size	1	
	OK Cancel	
		-

Figure 6.42: Interpolation Operation options

Seven Cell averaging algorithms can be chosen by the user; see Figure 6.43. These algorithms are explained below:

- \diamond SimpleAveraging: the arithmetic mean¹ of the samples inside the search area is taken.
- ♦ ClosestPoint: the value of the closest sample inside the search area is taken.
- ♦ MaximumValue: the maximum value of the samples inside the search area is taken.

¹The arithmetic mean is given by:

 $u(\mathbf{x}) = \frac{1}{N} \sum_{k=1}^{N} u_i$

- ♦ MinimumValue: the minimum value of the samples inside the search area is taken.
- InverseWeightedDistance: the inverse-distance-weighted mean² of the samples inside the search area is taken. Contrary to the arithmetic mean, in which all samples contribute equally to the mean, when using the inverse-distance-weighted average, the samples close to the query point have a larger contribution to the mean.
- MinAbs: the minimum of the absolute values of the samples inside the search area is taken.
- ♦ KdTree: This is an obsolete option, which will be removed in a future release.



Figure 6.43: Averaging options

By default, the interpolation will only overwrite missing values in the gridded data set. However, if the grid coverage already contains values, the user may choose to overwrite or combine the data by a pointwise arithmetic operation.

Interpolate single (selected) set

To perform interpolation on a single sample set, select the sample set (i.e. 'set1') in the operation stack and press 'Interpolate' in the 'Map' ribbon (Figure 6.44). Since no polygon is provided in this example, all the samples will be interpolated to the grid. Use polygons if you would like to have more control over the interpolation. After the interpolation the operation is added to the operations stack (Figure 6.45). Please note that after performing the interpolation the workflow in the stack is shifting from the sample set (i.e. 'set1' - which was a side step to construct the coverage) to the coverage (i.e. 'bed level').

```
<sup>2</sup>The inverse-distance-weighted mean is given by:
u(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) u_i}{\sum_{i=1}^{N} w_i(\mathbf{x})}, & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all } i, \\ u_i, & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some } i, \end{cases}w_i(\mathbf{x}) = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)}
```



Figure 6.44: Performing an interpolation operation on a single sample set (without using a polygon) using 'Interpolate' from the 'Spatial Operations' ribbon



Figure 6.45: Appearance of interpolation of 'set1' to the coverage 'bed level' in the operations stack

Interpolate multiple sets

To perform interpolation on multiple sample sets, select the active coverage (i.e. 'bed level') in the operation stack and press 'Interpolate' in the 'Map' ribbon (Figure 6.46). In the popup you can select which sample sets to include in the interpolation (in this example both). Since no polygon is provided, all the samples (from the two sets) will be interpolated to the grid. Use polygons if you would like to have more control over the interpolation. After the interpolation the operation is added to the operations stack (Figure 6.47). Again note that after performing the interpolation the workflow in the stack is shifting from the sample set (which was a side path to construct the coverage) to the coverage (i.e. bed level).

Note: Please note that interpolation of multiple sample sets can also be achieved by importing/combining different sample sets into the same set in the stack instead of using two separate sets. In this case you can just interpolate the single (selected) set.



Figure 6.46: Performing an interpolation operation on multiple sample sets (without using a polygon) using 'Interpolate' from the 'Spatial Operations' ribbon



Figure 6.47: Appearance of interpolation of 'set1' and 'set2' to the coverage 'bed level' in the operations stack

6.5.11 Smoothing

The smoothing operation smooths out (steep) gradients in a point cloud or coverage (depending on which one is active). The smoothing operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries or for the total data set if no polygon is provided. You have to assign the smoothing exponent and number of smoothing steps. The higher the exponent and the number of smoothing steps, the heavier the smoothing. For an example see Figure 6.48. After applying smoothing to (part of) the point cloud or coverage the operation is added to the operations stack (Figure 6.49).



Figure 6.48: Performing a smoothing operation on a point cloud with a polygon using 'Smoothing' from the 'Spatial Operations' ribbon



Figure 6.49: Appearance of smoothing operation in the operations stack



Figure 6.50: The cursor for the overwrite operation showing the value of the closest coverage point

6.5.12 Overwrite (single) value

The 'overwrite (single) value' operation allows you to edit single values on the active coverage after the interpolation. The 'overwrite (single) value' operation is activated from the 'Spatial Operations' ribbon. There is no geometry required for this operation. Upon selecting the operation from the ribbon a cursor will become active showing the coverage value closest to the cursor in a tooltipstring (Figure 6.50). Upon clicking LMB a popup appears in which you can overwrite the value of this coverage point Figure 6.51. After applying the overwrite operation it is added to the operations stack (Figure 6.52).



Figure 6.51: Performing an overwrite operation on a coverage point using 'Single Value' from the 'Spatial Operations' ribbon



Figure 6.52: Appearance of overwrite operation in the operations stack

6.6 Operation stack

The operation stack keeps track of the workflow of spatial operations that you performed. This helps you to make transparent how you arrived at your 'final' dataset without having to save all the intermediate datasets (steps) separately. Moreover, the stack is reproducible and easily editable without having to start all over again. This section describes the stack workflow (section 6.6.1), how to edit operation properties (section 6.6.2), how to enable/disable (section 6.6.3), delete (section 6.6.4), refresh(section 6.6.5) operations, quick links (section 6.6.6) and import/export functionality (section 6.6.7).

Note: Currently, the stack is saved in the User Interface project upon saving the project. The next time you open the project, the stack will reappear. The stack is not (yet) saved in a human readable/editable file.

ſ	
l	/• J

6.6.1 Stack workflow

Upon performing a spatial operation, the 'Operations' panel will open (see Figure 6.53) with the operations stack (tree). The stack first shows on which point cloud or coverage you are working (in this example 'bed level'). Subesquently, all the operations on this dataset are listed. For each operation you can inspect what the input, mask (e.g. the geometry used for the operation) and output are for the operation (Figure 6.54). By default the stack continues from the last operation that you performed. If you wish to work on a different dataset or operation within a dataset, you have to select that dataset or operation in the 'Operations' panel with the LMB.

When working on a coverage, point clouds (or sets) can be used to construct the coverage. In that case the stack jumps from the 'trunk' to a 'branch' and the subsequent operations are performed on the point cloud (see Figure 6.53). By selecting the set or coverage in the 'Operations' panel you determine on which dataset you are working. The interpolate operation (section 6.5.10) allows you to bring data from the point cloud (branch) to the coverage (trunk). See also Figure 6.53.





6.6.2 Edit operation properties

For each operation that you performed the properties (such as the value or 'Pointwise operation' of a 'Set Value' operation) can be edited using the 'Properties' window (Figure 6.55). **Note:** Please note that the mask of an operation cannot (yet) be edited. By editing the operation properties the operation stack becomes 'out of sync' and has to be refreshed for the changes to become active (see section 6.6.5).



6.6.3 Enable/disable operations

You can (temporarily) enable/disable operations by selecting the operation and pressing boxed cross icon in the stack menu (Figure 6.56). Upon disabling an operation the operation will be made grey in the stack and the operation is not taken into account anymore upon evaluation of the overall result. The result of disabling an operation is not directly activated. This is indicated in the stack with the 'out of sync' exlamation mark (Figure 6.56). You need to refresh the stack (see section 6.6.5) for the changes to become active.



Figure 6.54: Input for the operation (top panel), mask for the operation (middle panel) and output of the operation (bottom panel)



Figure 6.56: Disabling an operation using the boxed cross icon in the stack menu. The operation will become grey. Note the exlamation marks marking the stack 'out of sync'.

Ope	ratio	ons			•	д	>
Refresh 🔊 🔎 🗙 🔀 📄 📑				9			
		Bed l	level				
	• ···	4 Crop	1				
	<u> </u>	Set v	alue 1 :	-10			
			nput 🕈 mask				
			outpu	ıt			
Ch	art	Region	Map	Operations	Toolbox		
Properties •					д	;	
	Inte	rpolation	operati	on		·	
		1					
•	A Z	1					
•	A Z Ge	neral					
~	A Ger Nar	neral me		Set value	: 1		
*	£ ↓ Ger Nar Ena	neral me bled		Set value True	:1		
* *	A Ger Nar Ena	neral me bled eration p	aramet	Set value True ers	e 1		
×	Agen Nar Ena Ope Poi	neral me bled eration p	aramet peration	Set value True ers Overwrit	e 1 te		

Figure 6.55: Editing the value or 'Pointwise operation' of a 'Set Value' operation using the properties panel

6.6.4 Delete operations

To delete an operation permanently you have to select the operation and either press the cross icon (Figure 6.57) or use the context menu and select delete (Figure 6.58). The operation will be removed from the stack. The result of deleting an operation is not directly activated. This is indicated in the stack with the 'out of sync' exlamation mark. You need to refresh the stack (see section 6.6.5) for the changes to become active.



Figure 6.57: Removing an operation from the stack using the cross icon in the stack menu

Operations			•	џ	×
🗄 🔂 Refresh 🔊 🔎	×E	3 2 4			
🖃 🗐 Bed Level					
Bed Level					
🖨 🗐 set 1					
🖶 🛃 Import sam	nples	1			
🕀 📲 Add points	1				
🗄 📗 Gradient 1	1	Import			
🖮 🗐 set 2	4	Export			
🗄 📄 Import san	×	Delete			
	\sim	Delete			

Figure 6.58: Removing an operation from the stack using the context menu on the selected operation

6.6.5 Refresh stack

When the stack is marked 'out of sync' by exclamation marks, you can refresh the stack by pressing the 'Refresh' button for the changes to become active (Figure 6.59). Upon refreshing the stack all the (enabled) operations in the stack will be (re-)evaluated. **Note:** Please note that refreshing the stack can take some time when large datasets are (re-)evaluated!





Figure 6.59: Refresh the stack using the 'Refresh' button so that all operation are (re-) evaluated

6.6.6 Quick links

The stack menu contains two quick links to quickly show the original dataset (e.g. where you started from, Figure 6.60) and the end result of the spatial operations (Figure 6.61).



Figure 6.60: Quick link to the original dataset before performing any spatial operations



Figure 6.61: Quick link to the output after performing all (enabled) operations

6.6.7 Import/export

Note: Importing and exporting data into or from the stack is still under construction



7 Model coupling

7.1 Model coupling using integrated models within the User Interface

Besides adding stand-alone models to a project, it is also possible to create integrated models or upgrade stand-alone models to integrated models. A flow model and a rainfall-runoff model can be independent, but they may also be combined in an integrated model. An integrated model consists of a set of models which may run sequentially or in parallel. The first means that the output of the first sub-model is generated and used as input for the second model within the integrated model without any feedback from the second model to the first. The second means that the output of one sub-model is used as input of the second sub-model within the integrated model and vice versa. In this way the models are coupled externally (Morita and Yen, 2002) on a time step basis.

An example of an integrated model is a D-RTC model (a Real Time Control model) combined with a D-Flow FM model (Figure 7.1). When the integrated model is run, each timestep both the D-Flow FM model and the D-RTC model are run sequential or parallel. The output of the D-Flow FM model is input for the D-RTC model. In case the models are run parallel the inand output are exchanged every time step: the output of the D-RTC model in the previous timestep is used as input for the D-Flow FM model, and vice-versa. The models are coupled on the structures which are controlled.



Figure 7.1: Example of an integrated model

7.2 Model coupling using the BMI standard

Most computational cores available within the User Interface can not only be coupled using the integrated model functionality within the D-Flow FM GUI, but can also be coupled outside the Delta Shell GUI for running on clusters. For such model runs, which usually require a high performance, the models are coupled using the Basic Model Interface (BMI). For a description of the BMI the user is referred to http://csdms.colorado.edu/wiki/BMI_Description.

7.3 Model coupling using the OpenMI standard

Applications that are based on the Delta Shell framework, like SOBEK 3 and Delft 3D Flexible Mesh (FM), are so-called OpenMI-compliant with version 1.4 and 2.0 of the OpenMI-standard. The OpenMI-standard, the documentation and the supporting tools can be found at http://www.openmi.org.

SOBEK 3- and Delft 3D FM-models can be used in an OpenMI-configuration by providing a so-called <*.omi>-file in which the project (<*.dsproj>) is specified. An example follows here.

8 Command-line and scripting

8.1 Introduction

Setting up and running models using (ASCII) scripts can be a very useful work flow for modellers that would like to perform batch studies where a single model parameter needs to be changed, to perform batches of calculations on large computational clusters, or for simply reproducing the model setup procedure (before extending the model to a more complex version for example) over and over again. To support the here-fore mentioned work flows, the Delta Shell framework supports the IronPython scripting language encapsulated within the Delta Shell scripting toolbox. It may be used from within the Delta Shell GUI (only on Windows machines), as well as from a console/terminal on any platform. In the following sections, the use of the Delta Shell command-line and scripting functionality will be explained based on the use of existing D-Flow 1D projects.

8.2 Scripting editor overview

The scripting editor within Delta Shell contains the following features:

- ♦ Syntax highlighting section 8.2.1
- ♦ Code completion section 8.2.2
- ♦ Show extra characters section 8.2.3
- ♦ Regions section 8.2.4
- ♦ Local variables section 8.2.5
- Ribbon section 8.2.7
- ♦ Shortcuts section 8.2.8

8.2.1 Syntax highlighting

With syntax highlighting all the words and types that are known in Python will be shown as colored text. To configure the color schema go to the Delta Shell installation folder and navigate to the plugins/Delta Shell.Plugins.Scripting.Gui folder. Here the XML-file <Python.xshd> contains all the information used for syntax highlighting. An example is presented in Figure 8.1.

<u> </u> N	Лу script * 🗙
1	CurrentProject.Name = "abc"
2	
3	print "Changed name"

Figure 8.1: Example of syntax highlighting within the Delta Shell scripting editor.

8.2.2 Code completion

Code completion is used to give the you quick access to all properties, events and methods of the variable that you are working on. When activated a list will be shown showing all the options for this variable and the documentation for the selected element. To filter the list continue typing, and the list will filter out all non matching elements. To select an element press the up and down keys and enter to confirm. If an empty list is shown, then the type of the variable can not be determined or the variable has no elements to show. Code completion is activated when a '.' is typed or when Ctrl + Space is pressed. An example is shown in Figure 8.2.

My script * 🗙			
1 CurrentProject.	N = "abc"		
2 3 print "Changed	DefaultName	*	Property : String Name
o primo onangea	Description		Description : Gets or sets a readable name of the project
	EditWasCancelled		Maria Garana Gurbara
	EndEdit	Π	Namespace :system
	FileFormatVersion		
	GetDirectChildren	=	
	GetEntityType		
	IsChanged		
	IsEditing		
	IsNestedEditingDone		
	Name	•	

Figure 8.2: Example of code completion within the Delta Shell scripting editor.

8.2.3 Show extra characters

These options are added to show you the space, tab and line end characters. This can be important because pythons logic uses indenting for its statements. Figure 8.3 to Figure 8.5 show examples of these extra characters.

🗾 My	y script * 🗙
1 2 3	<pre>def ChangeProjectName (newName) :</pre>
5	ChangeProjectName("abc")



🗾 М	y script * 🗙
1 2 3	<pre>def ChangeProjectName (newName): CurrentProject.Name = newName print "Changed name to " + newName</pre>
5	ChangeProjectName("abc")

Figure 8.4: Showing tabs within the Delta Shell scripting editor.



Figure 8.5: Showing EOL characters within the Delta Shell scripting editor.

8.2.4 Regions

Regions are used to mark code blocks, and gives you the ability to collapse these blocks to a single line with a title. Figure 8.6 and Figure 8.7 shows an examples of such a region.

🛛 🗾 My	v script * 🗙
1 🗄	#region Functions
2 3 4	<pre>def ChangeProjectName(newName): """Changes name of project to newName""" CurrentEroject Name = newName</pre>
6	print "Changed name to " + newName
8 L 9	-#endregion
10	ChangeProjectName("abc")





Figure 8.7: Closed region within the Delta Shell scripting editor.

8.2.5 Local variables

The local variable option gives you an overview of the declared variables in your script. The variables are declared when you run the part of your script that declares the variables. It will show the name of the variable followed by a string representation of the variable value and the type. An example is presented in Figure 8.8. To get more detail about the variable, select it and the property window will show you all the details about the variable as shown in Figure 8.9.

📕 My script (read-only)	×				
1 newName = ' 2	'abc"				
3 project = 0 4	project = CurrentProject				
5 project.Nam 6	5 project.Name = newName 6				
7 print "Char	nged name to " +	newName			
💼 Local variables 🔒	Watch variables				
Name	Value	Туре			
newName	abc	System.String			
project	abc	DelftTools.Shel			

Figure 8.8: Local variables within the Delta Shell scripting editor.

	₽↓		
4	Misc		Ľ
	Name	abc	
	CreateTime		
	ChangeTime		
	FileFormatVersion	1.1.0.0	1
	RootFolder	<root></root>	
	Description		
	Size	0	1
	IsChanged	True	
	IsTemporary	True	
	ViewContextManao	DelftTools.Shell.Gui.G	1

Figure 8.9: Properties of variables within the Delta Shell scripting editor.

There is also an option called watch variables that allows you to add your own variables that you want to observe. This works exactly the same as the local variables, with one exception. The variables are not automatically reloaded after running a part of the script. Reloading can be done by pressing the *refresh values* button below the watch variables. An example of a watch variable is shown in Figure 8.10.

Local variables	🏜 Watch variables	
Statement	Value	Туре
project.Name	abc	System.String

Figure 8.10: Watch variables within the Delta Shell scripting editor.

8.2.6 Ribbon

8.2.7 Scripting

When you open the scripting editor in the User Interface, a Scripting *ribbon* category will appear. This ribbon has the following additional options (see also Figure 8.11), which are described in Table 8.1:

File	Home	View	Tools Scripting						
Run C Script	Clear cached variables	Show spaces Show tabs Show end of	Python private variable: Highlight current line	Text color • Background color • Font size 14	Local variables	 Insert spaces Keep tabs Tab size 		Python	✓ Save before run
F	Run		View				Editing	Docum	Settings



Function	Description			
Run script	Executes the selected text. If no text is selected then it will execute the entire script			
Clear cached variables	Clears all variables and loaded libraries from memory			
Debugging	Enables/Disables the debug option. When enabled you can add breakpoint to the code (using F9 or clicking in the margin) and the code will stop at this point before executing the statement (use F10 (step over) or F11 (step into) for a more step by step approach)			
Python variables	Show or hide python variables (like _var_) in code comple- tion			
Insert spaces/tabs	Determines if spaces or tab characters are added when pressing tab			
Tab size	Sets the number of spaces that are considered equal to a tab character			
Save before run	Saves the changes to the file before running			
Create region	Creates a new region surrounding the selected text			
Comment selection	Comments out the selected text			
Convert to space indenting	Converts all tab characters in the script to spaces. The num- ber of spaces is determined by Tab size			
Convert to tab indenting	Converts all x number of space characters (determined by ab size) in the script to tabs			

Table 8.1: Functions and their descriptions within the scripting ribbon of the User Interface.

Table 8.1: Functions and their descriptions within the scripting ribbon of the User Interface.

Function	Description				
Python (documentation)	Opens a link to the python website, showing you the python syntax and standard libraries				

8.2.8 Shortcuts

The shortcut keys of the scripting editor within the User Interface are documented in Table 8.2.

Shortcut	Function
Ctrl + Enter	Run selection (or entire script with no selection)
Ctrl + Shift + Enter	Run current region (region where the cursor is in)
Ctrl + X	Cut selection
Ctrl + C	Copy selection
Ctrl + V	Paste selection
Ctrl + S	Save script
Ctrl + -	Collapse all regions
Ctrl + +	Expand all regions
Ctrl + "	Comment or Uncomment current selection
Ctrl + W	Add selection as watch variable
Ctrl + H	Highlight current selection in script (press esc to cancel)
F 9	Add/remove breakpoint (In debug mode only)
F 5	Continue running (In debug mode only — when on breakpoint)
Shift + F5	Stop running (In debug mode only — when on breakpoint)
F10	Step over current line and break on next line (In debug mode only - when on breakpoint)
F11	Step into current line if possible, otherwise go to next line (In debug mode only — when on breakpoint). This is used to debug functions declared in the same script (that have already runned)

Table 8.2: Shortcut keys within the scripting editor of the User Interface.

8.3 Toolbox overview

The toolbox in Delta Shell consists of three folders, see Figure 8.12. The first two folders "Models" and "Items" contain lists of models and items that can be added to your project. The third folder "Scripts" shows all the sub-folders and (python) script files that are in the scripting folder.

Toolbox	•	×
Settings		
Toolbox		
Models		
🕨 🔚 Items		
Scripts		
Chart Toolbox		

Figure 8.12: The Toolbox window within Delta Shell.

The scripting folder is a folder on your file system that is used for storing scripts that you would want to share between projects. This folder is synchronized with the file system so that you can manage the scripts in the toolbox and with your file browser (windows explorer). To change the location of the scripting folder right click on the toolbox node and select *Change scripting folder*.

8.3.1 Exploring the scripting folder

The toolbox gives you an easy way to work and manage the scripts in the scripting folder. Figure 8.13 shows an expanded view of the scripting folder. By using the + sign next to the file you can see all the declared functions and classes that are in that script. By selecting the script or one of the functions you can also see the documentation of the selection in the property grid. By double clicking one of the functions you will open the scripting editor at the start of the declaration of the function.

	Toolbox	,	џ	x
	✓ Settings			
	🔺 💼 Toolbox			•
	Models			
	Items			
	Scripts			
	Examples			
	4 🚞 Libraries			
	ChartFunctions			
	AddValuesToSeries (list, se (line 12)	rie	:s)	
	CreateAreaSeries (list)			
	CreateBarSeries (list) (line 26)			
	(ine 26) CreateChart (series) (line 38)			
	CreateLineSeries (list)			
	CreatePointSeries (list) (line 30)			
	Conversions			
	FlowFlexibleMeshFunctions			
	MapFunctions			
	NetCdfFunctions			
	NetworkFunctions			
	RainfallRunoffModelFunctions			
	RtcModelFunctions			
	Sobek2Functions			
	SobekFunctions			
	SobekWaterFlowFunctions			
	D SpatialOperations			•
I	Toolbox Chart			
	Properties .	•	џ	×
	Toolbox			•
	₽ ₽ ↓ ©			
	▲ General			
	Name CreateAreaSeries (li	ist)	
	Description Creates an area serie	25	for	the
	Name			

Figure 8.13: Expanded view of the Scripts folder.

8.3.2 Working with scripts

After selecting a script in the toolbox and right clicking it, you get a context menu with the options as shown in Figure 8.14. The options are described in Table 8.3.

	Edit script
	Run file
9	Insert code
a	Add as import
æ	Add as import (using alias)
a	Open location with windows explorer
Ŧ	Rename
×	Delete



Function	Description			
Edit script	Opens the scripting editor for this script			
Run file	Runs the whole script without opening the editor			
Insert code	Inserts the code of the selected script into the current script (the script open in the scripting editor)			
Add as import	Adds the selected script as import to the current script (the script open in the scripting editor)			
Open location with windows explorer	Opens an instance of windows explorer for the location of the selected script			
Rename	Rename the script (also on file system)			
Delete	Deletes the selected script (also on file system)			

 Table 8.3: Context menu function descriptions of a script item within the toolbox.

8.3.3 Working with folders under the scripting folder

After selecting a folder under the scripting folder "Scripts" in the toolbox and right clicking it, you get a context menu with the options as shown in Figure 8.15. The options are described in Table 8.4.

a	Open location with windows explorer
	Add new folder
	Add new script
2	Import script from existing
A	Add as import
A	Add as import (using alias)
P	Find in folder
Ŧ	Rename
×	Delete

Figure 8.15: Context menu of the "Scripts" folder (or one of its sub-folders) within the toolbox.

Function	Description			
Open location with windows explorer	Opens an instance of windows explorer for the location of the selected folder			
Add new folder	Adds a new folder under the current selected folder and adds ainitpy file to that folder (to make it work as import)			
Add new script	Adds a new (python) script to the folder			
Import script from existing	Adds a new (python) script to the folder with the content of an existing script (like a file copy)			
Add as import	Adds the folder as import to the current script			
Rename	Rename the folder (also on file system)			
Delete	Deletes the selected folder (also on file system)			

Table 8.4: Context menu function descriptions of the "Scripts" folder (or one of its subfolders) within the toolbox.

8.4 Running scripts from within the Delta Shell GUI (Windows only)

After starting Delta Shell, add a script item to a new project or an existing project by rightmouse clicking on your project, and selecting $Add \rightarrow New Item \dots$ shown in Figure 8.16, or add a script to your scripting folder by selecting *Add new script* from the context menu of the scripting folder within the **Toolbox** window as shown in Figure 8.15.

When adding a script to your **Project** window, a selection dialog for adding items to your project will appear. Now select the *Script* item from the various types as shown in Figure 8.17. A script item is now shown within the project. Double-click on the script item to open the script editor within the Delta Shell GUI.

Start editing the script (either in your project or in your scripting folder) using Python syntax, as shown in Figure 8.18. As shown in the figure, it is possible to start a script by importing functions from pre-defined routines developed within provided Python scripts with the Toolbox plug-in or creating your own Python libraries and adding them to your Scripts folder. These functions can than be used within in the main script. For example, in Figure 8.18, the functions SetDefaultRoughness, and AddRoughnessAtLocation, have been imported from

2 2	-	2 2 -		
Project Explorer		4	×	Script1
4				date time [yyyy/MM Discharge
B SW max B				▶ 2007/01/15 01:00:00
🗄 🍈 Base ca		Add	•	New Item 0.2347
Script1		Import		🖓 New Model 0.4695
	-0			New Folder 0.7042
		Export		0.9390
	6	Paste Ctrl + V		2007/01/15 06:00:00 1.173
		Rename		2007/01/15 07:00:00 1.408
		Kendine		2007/01/15 08:00:00 1.643
	•	Run All Models		2007/01/15 09:00:00 1.878
		Stop All Models		2007/01/15 10:00:00 2.230
	_			2007/01/15 11:00:00 2.582
		Properties		2007/01/15 12:00:00 2.934
	_			2007/01/15 13:00:00 3.286
				2007/01/15 14:00:00 3.638
Man Contents		Д.	×	2007/01/15 15:00:00 3.990
				2007/01/15 16:00:00 4.34

Figure 8.16: Add an item to your Delta Shell project.

General			
😁 Мар	i (World)	Script	Text Document
® Web Link			
Hydro			
Basin	🧏 Network	💥 Region	

Figure 8.17: Select script to add to your Delta Shell project.

the pre-defined toolbox libraries as shown in the first six lines of the script.

An example of an auxiliary function as available in the pre-defined Toolbox plug-in libraries is given in Figure 8.19. The function shown needs an existing flow model and boundary name as input arguments to be able to return the desired boundary condition data to the user.

To test the script regularly during editing, or after finishing the script, the user can press the *Run script* button (within the Scripting *ribbon* in the Delta Shell GUI) shown in Figure 8.20 to see if the script is working properly. If there is something wrong within the script the messages tab within the Delta Shell GUI shows the error messages encountered during the running of the script. The messages tab will also show the messages that the user has defined himself by using rint \normalfont Python commands within the script.



Figure 8.18: Scripting editor within the Delta Shell GUI.



Figure 8.19: Create your own auxiliary functions or use the pre-defined library functions using Python.



Figure 8.20: Run script within Delta Shell GUI.

8.5 The Delta Shell Console

Besides setting up scripts and running them from the Delta Shell GUI, users can set up scripts and run them using the Delta Shell console application. The Delta Shell console application options can be obtained from a console/terminal by typing *DeltaShell.Console.exe -?*. This will result in the options as presented in Figure 8.21.

Administrator: C:\Windows\System32\	cmd.exe	
d:\putten_hs\Downloads\Delta Usage: DeltaShell.Console [0	Shell\bin\Release>DeltaShell.Console.exe -? PTION]	
Valid options include:		
-h, $-?$, $help$	Show usage and exits.	
-p,project-file-OHLOE	additional options below.	
-r,run-activity=VALUE	Run activity or model available in the project.	
-f,run-file=VHLUE	Run script from file Run specified command (Ruthen)	
-iinteractive	Start interactive console	
-s,skip-default-libs	Do not load standard python libraries (faster	
	start-up)	
		-

Figure 8.21: Delta Shell console application options overview.

8.5.1 Testing and running your scripts using the Delta Shell interactive console (Windows/Linux/MacOSX)

The Delta Shell interactive console application can be started from a console/terminal by typing *DeltaShell.Console.exe -i* within the console (make sure the Delta Shell bin directory has been added to your path or start the application within the Delta Shell bin directory of the installation) or by clicking the *Deltares* \rightarrow *<Name of Software Suite>* \rightarrow *Interactive Console* within the Windows start menu. Within the console application, the user can test the script command by command without losing information that has been gathered thus far, and thus can easily test these commands on a step by step basis without having to worry about making errors. The commands of the script shown in Figure 8.18 can be tested line by line in this fashion as well. After finishing the script (and saving all commands to a *.py file), run the complete script within the Delta Shell console application using the following command: *Application.ScriptRunner.RunScript((open('scriptname.py', 'r').read())).*

Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe			
PS_D:\delta-shell\src\DeltaShell\DeltaShell.Loader\bin\Debug>\DeltaShell.Conso			
le.exe -i	<u> </u>		
Starting Delta Shell			
Loading plugin: CommonToolsPlugin			
Loading plugin: NHibernateDaoPlugin			
Loading plugin: XmlDataAccessPlugin			
Loading plugin: RainfallRunoffPlugin			
Loading plugin: RealTimeControlPlugin			
Loading plugin: WaterFlowModel1DPlugin			
Loading plugin: WaterYualityModellDPlugin			
Loading plugin: Developeriooisriugin			
Loading plugin. rewsrlugin			
Loading plugin: Habitatingin			
Loading plugin. Namical Importer i lugin			
Loading plagin: Sobel moort Plagin			
Loading plugin: MorphanPlugin			
Loading plugin: MorphAnDemosPlugin			
Loading plugin: NetCdfPlugin			
Loading plugin: OpenMIPlugin			
Loading plugin: ProjectExplorerPlugin			
Loading plugin: ScriptingPlugin			
Loading plugin: SeriesPlugin			
Loading plugin: NetworkEditorPlugin			
Loading plugin: SharpMapGisPlugin			
Loading plugin: loolboxPlugin			
Loading plugin: WPDExplorerriugin			
Loading plugin. wrDExplorerDemosringin			
Loading plugin. Wheekplurerrottypelbobringin			
Douling plugin. Abeachilugin			
27 plagints were du			
IronPython 2.6.2 (2.6.10920.0) on .NET 2.0.50727.5456			
Type "help", "copyright", "credits" or "license" for more information.			
	T		

Figure 8.22: Test script in interactive mode.

8.5.2 Running your script directly from the commandline (Windows/Linux/MacOSX)

A finished or already existing Python script can easily be run using the Delta Shell console application as well, by passing the script to the application directly as an input argument. The command to do this from a console/terminal is: *DeltaShell.Console.exe –run-file=script.py* as is also shown in Figure 8.23.

Finally, note that when running scripts using the Delta Shell GUI, the user may also use Delta Shell GUI components within the scripts. When running scripts from the Delta Shell console application or from the command-line, these components are not available, and the user needs to fall back on GUI components (for example for plotting) available within Python or develop them himself.



Figure 8.23: Run script from command line within console application.

References

Morita, M. and B. Yen, 2002. "Modeling of Conjunctive Two-Dimensional Surface-Three-Dimensional Subsurface Flows." *Journal of Hydraulic Engineering* 128 (2): 184–200. DOI: 10.1061/(ASCE)0733-9429(2002)128:2(184).

A How to use OpenDA for Delta Shell models

OpenDA is an open-source software tool distributed by the OpenDA Association (see www. openda.org). It enables the user to calibrate and Ensemble Kalman Filter (EnKF) simulation models, such as Delft3D Flexible Mesh Suite and SOBEK3. In this document we will speak of Delta Shell (models).

Both the calibration of Delta Shell models and running them in EnKF-mode is done by using OpenDA. To run an OpenDA calibration or EnKF-simulation, a so called OpenDA application (.oda) file is needed, in which the application to be performed is specified. This oda file is the top of a hierarchy of configuration files that is organized in a directory structure that is usually setup as indicated below.

- ♦ topDir, containing e.g. for EnKF of a SOBEK3 model textitSimpleModel:
 - SimpleModelEnKF.oda>)
 - algorithm contains the configuration file(s) for the calibration or EnKF algorithm
 - stochObserver contains the configuration file(s) and measurement data for the so called 'stochastic observer', the set of measurements and the specification of their uncertainty
 - stochModel contains the configuration file(s) for the so called 'stochastic model factory', that specify how model instances can be created. For Delta Shell models, this is described in
 - template contains the base model (<SimpleModel.md1d> and the associated model files)
 - **bbWrapperConfig.xml** describes the way the model can be addressed. For Delta Shell models the so-called *black-box* wrapper is used.
 - bbModelConfig.xml specifies the <*.md1d> model file and some other optional settings for repeatedly running the model.
 - bbStochModelConfig.xml describes which items can be calibrated, and specifies the relation between the measurement series and the related observation point in the model

For the all over structure and the content of the various files, the user is referred to the documentation of OpenDA on www.openda.org. Up-to-date examples can be found there.





+31 (0)88 335 81 88 software@deltares.nl www.deltares.nl/software