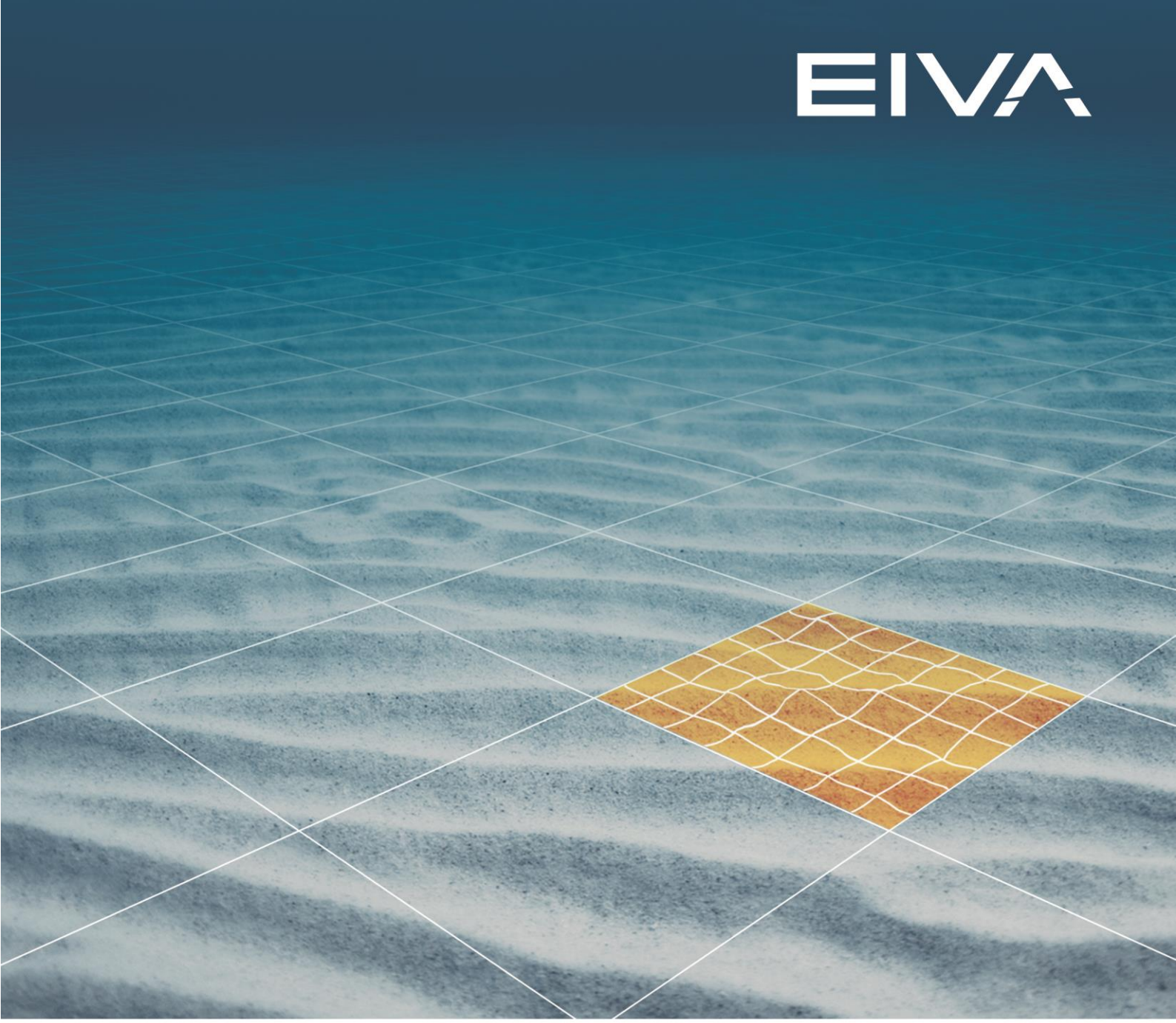


NAVIPAC

A1. USER DEFINED OUTPUTS

Last update: 18/02/2020
Version: 4.2.3

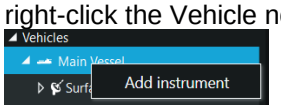
Contents

1 Description of user interface

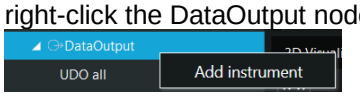
A User defined output is an option to specify up to 15 custom or user defined outputs in NaviPac. The output can be defined in the NaviPac configuration module NaviPac.exe.

To add a **User defined output** instrument the operator has numerous options, ie either

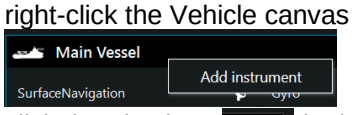
- right-click the Vehicle node in the Project Tree and select *Add instrument*



- right-click the DataOutput node in the Project Tree and select *Add instrument*

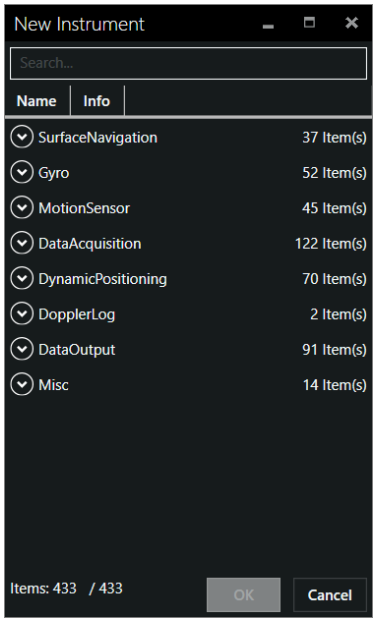


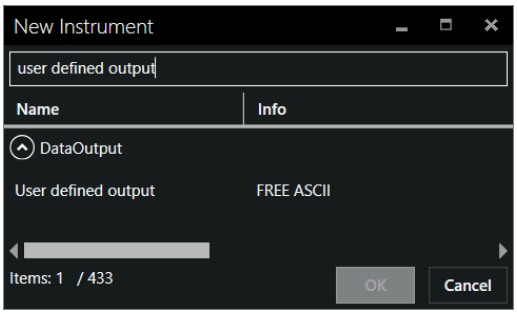
- right-click the Vehicle canvas and select *Add instrument*



- click the plus icon  in the Vehicle title bar

These open the *New Instrument* dialogue. Then manually locate and select the User defined output instrument in the DataOutput category.



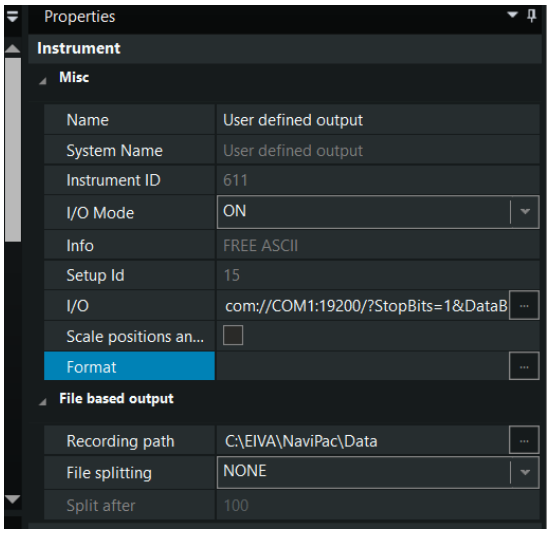


Notice: for a general run-through of the NaviPac.exe configuration module please turn to the EIVA NaviPac 4.0 Configuration Essentials eLearning module regarding this topic.

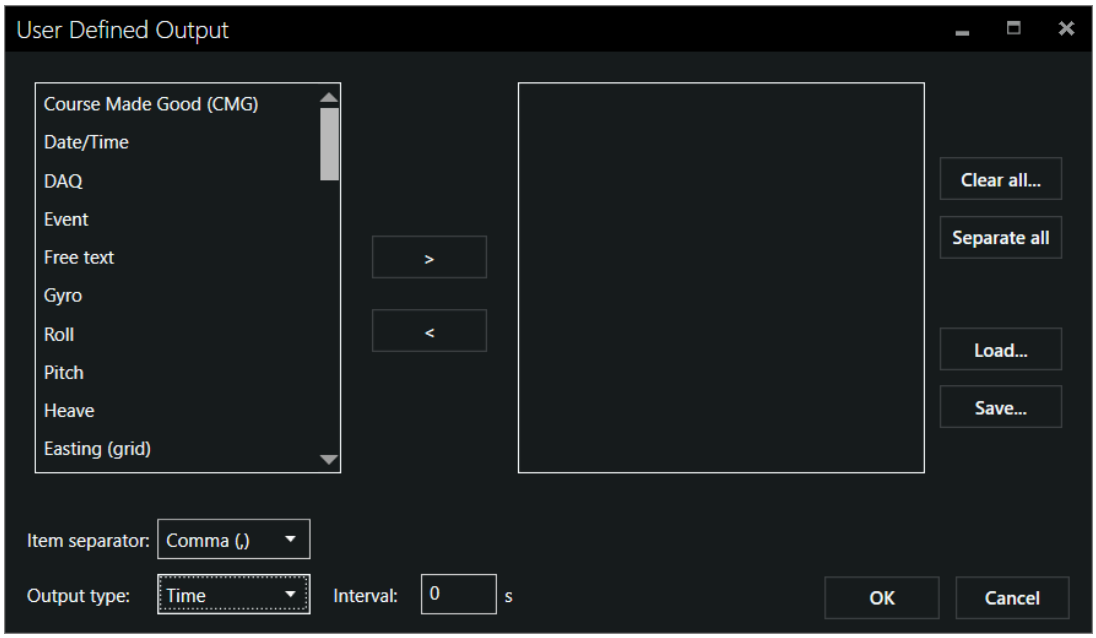
1.1 Edit a User defined output

Select the User defined output instrument, either in the Project Tree or in the Vehicle section. In the Properties section then enter/specify:

- Name
- I/O Mode: On, Off
- I/O Interfacing details
- Output format



Click the property *Format* to open the special dialogue used for defining a User defined output.



The dialogue contains 2 lists with items that can be output on the selected output port or file. The left list contains all optional data items whereas the right one contains the active items selected by the user, thus actually defining the complete user defined output.

The following items can be selected:

- Free text items – ASCII or binary
- Item separators: **comma, space, colon, semicolon and tabulator**
- Terminators: **CR, LF, STX, and ETX**
- Date/Time
- Position items:
  X, Y, KP, DAL, DOL, DCC, LAT, LON, AGE, HEIGHT, Standard deviation (DOP)
- Gyro
- CMG (Course Made Good)
- SMG (Speed Made Good)
- Motion sensor data (Roll, Pitch, Heave)
- DVL (Speed log data) <esp. RDI PD0 information>
- DAQ (eg depths from echo sounders or other data acquisition channels)
- Event
- Raw data – copy of incoming data string
- Geodesy information
- Runline control information (name, logging on/off) and direction
- RTK-based heave bias (heave correction calc.)
- Check sum (2 character exclusive-or)
- Combined data acquisition – scaling and arithmetic
- Runline direction (true or grid)
- Range/bearing to waypoint
- Distance to waypoint

Note: There is practically no limit to the number of times an item type can be included for different objects.

Available list (leftmost list):
This list contains all output items available. An item can simply be drag-and-dropped from the *Available* list to the *Selected* list. Alternatively, an item can be selected in the *Available* list and then moved by clicking the *Include* button (see below) to place a copy in the *Selected* list. To remove (clear) an item from the *Selected* list, chose the item by clicking the item in the *Selected* list and then click the *Exclude* button (left arrow).

Selected list (rightmost list):
This list contains all output items selected to be output to a port or file. It can be removed by selecting the item and then either clicking the embedded *Delete* button  (the trash bin icon) or click the *Exclude* button (see below).
If an entry is included in the list with reference to an unknown item (eg position of an object that no longer exists or reference to a gyro that has been removed) then the line will include the text: **NOT FOUND !**

Include button (right arrow ):
A chosen output item in the *Available* list is copied to the *Selected* list, which indicates that

the output item is to be a part of the output. An item can also be drag-and-dropped from the *Available* list to the *Selected* list.

Exclude button (left arrow [redacted]):
Remove a selected output item from the *Selected* list, ie do not output it. A selected item can also be cleared by via the *Delete* button.

Clear all… button:
Erase the entire included list. The operator will be prompted to accept.

Separate all button:
Put the currently selected separator between all the selected items. A quick way to finalise the string.

Below the 2 lists the following can be set up:

Item separator:
How to separate fields/columns (output items in the selected list): **comma, space, colon, semicolon and Tab.** These will apply to all items.

Output type:
Specify how often the output is generated: **All updates, Time, Event**

Interval:
If Time is selected in *Output type* then the frequency to output the output string can be specified.

Example:

If the user were to select and define the following:

- Free text item (time)
- <Separator>
- Date/time item
- <Separator>
- Free text item (position (X,Y))
- <Separator>
- Position (X)
- <Separator>
- Position (Y)
- <Separator>
- text item (depth)
- <Separator>
- DAQ (echo sounder channel)
- <CR>
- <LF>

The resulting output string could be:
Time, Date/Time, Position (X, Y), X (grid), Y (grid), NaviSound 2000:1<CR><LF>

OK button:
Accept last changes in the *Selected* list and save changes.

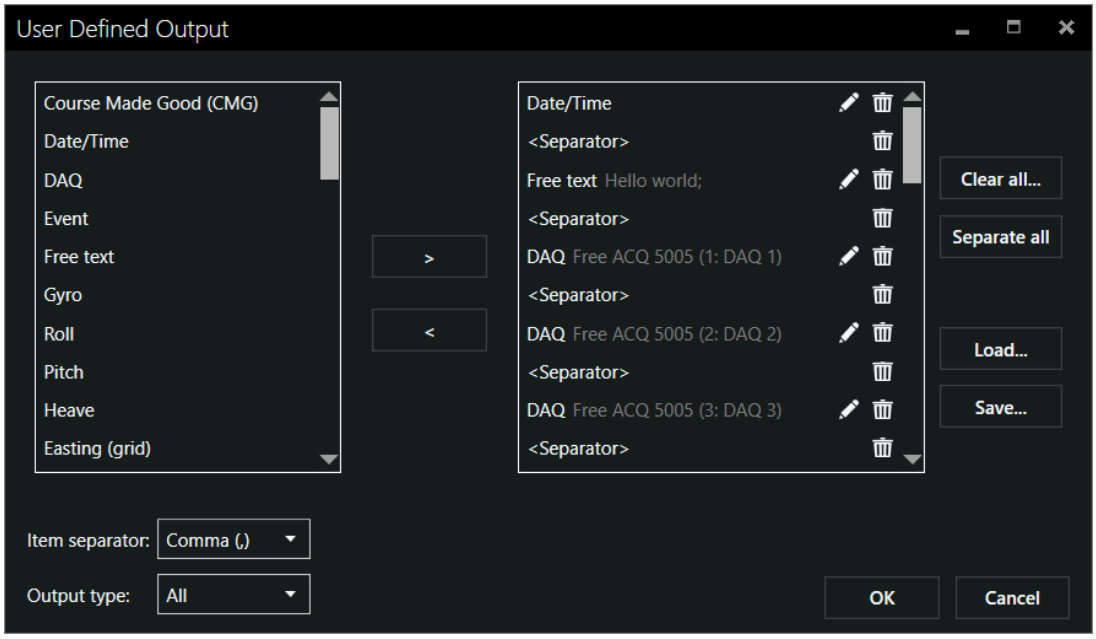
Cancel button:
Close the dialogue without saving changes.

Save… button:
Save a copy of the current defined output to an external file (.out2). This file can then be used at a later stage.

Load… button:
Load output definitions from an external file (.out2) – current setup will be erased!

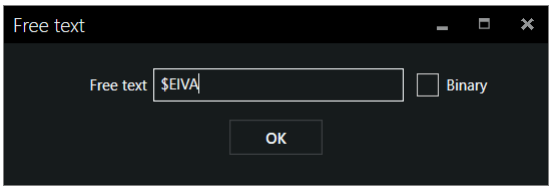
1.2 Item definition – the edit item icon

It is possible (and often required) to edit an item for the majority of output items. If the item can be edited, then a small pen icon is shown next to the trash bin icon. Either double-click the item itself or click the pen icon to edit the item.

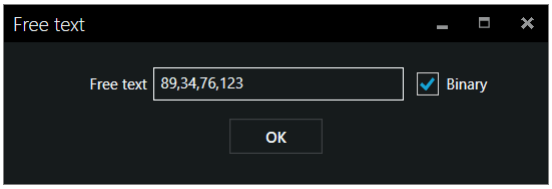


The various properties to edit will depend on the item type selected. Some of these will be described in the following sections.

1.1.1 Free text item

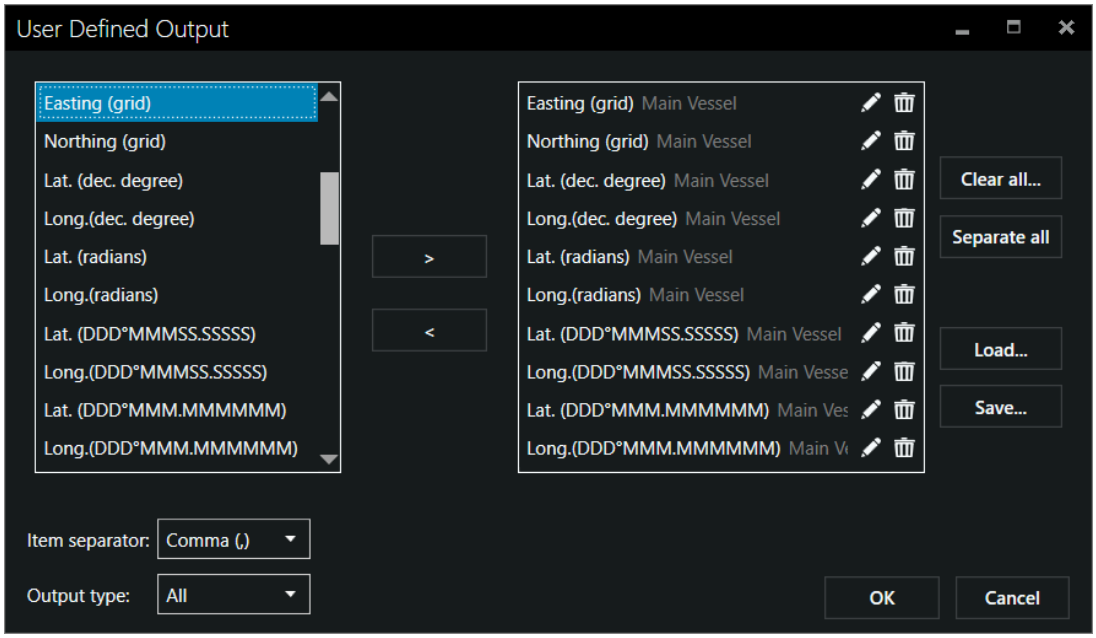


A dialogue with an ASCII text field is shown. To output binary data, select Binary and enter the ASCII codes separated by a comma.

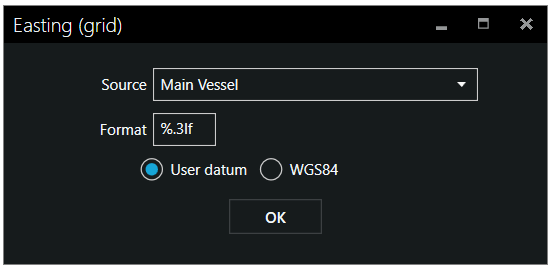


1.1.2 Position items

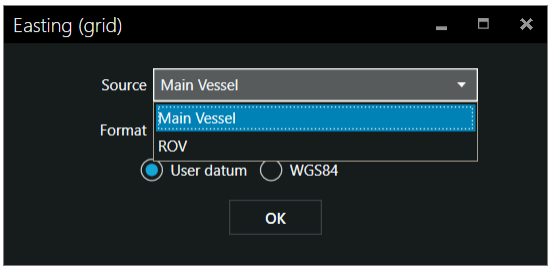
Positions can have either a default format, a user defined output format, eg nnnnnnn.nn, or a fixed format, eg DDD°MMMSS.SSSSS.



If a fixed item type is selected a default format is used. If not, the user is allowed to enter a format string. The format field for positions (and other data items) follows the ANSI C syntax – see section 2 Custom output format strings.

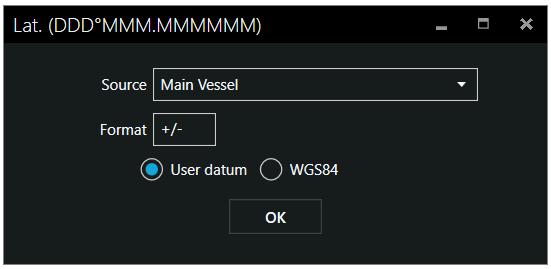


Select which source (dynamic object, POI) is to be the grid position provider. Choose between current geodetic datum (User Datum), eg ED50, or convert to WGS84.



1.1.2.1 Latitude/Longitude items

The latitude/longitude comes in 4 different formats.



- Signed (default):
  The output will be prefixed with a minus to indicate west or south.
  Sample: -000°01.146058'
  Definition: Enter +/- in the format field

- Append indicator:
  An E/N/S/W will be appended to the string.
  Sample: 000°01.146058'W
  Definition: Enter E or N or S or W in the format field

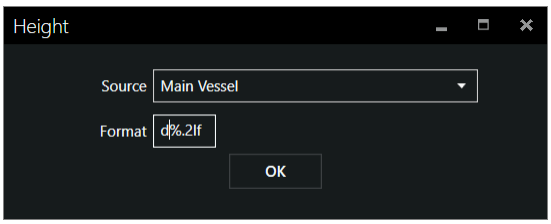
- Separator appended indicator:
  A separator plus E/N/S/W will be appended to the string.
  Sample: 000°01.146058',W

Definition: Enter the separator (NOT E or N or S or W) in the format field – eg ‘,’ in the above sample

- Special case:
  Data without degree signs with 4 decimals.
  Sample: Lat <DD MMM.MMMM> N Long <DDD MMM.MMMM> E
  Definition: Enter X

1.1.2.2 Height (and Depth) item

NaviPac uses positive upward for heights, which means that the height output of eg a ROV will become negative. This can be overruled by inserting a ‘d’ character in the format field indicating that you want to have it output as Depth.

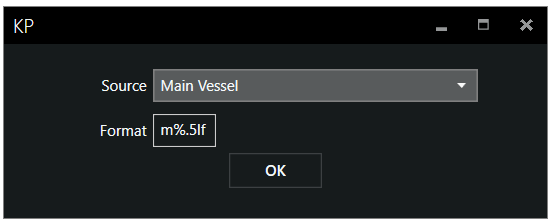


1.1.3 KP item

The unit of KP (kilometre point of a runline) is similar to the unit of the position output. The format unit is default in kilometre.

1.1.3.1 Forced to metre

Often the format unit is wanted in metres. This can be done by prefixing the format string with an ‘m’ character (or ‘M’).



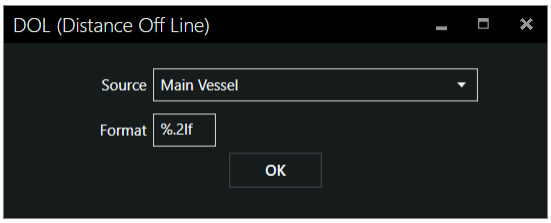
The above example will output KP in metre with at least 6 digits including leading zero. The default KP setting outputs in km with 5 digits including leading zero.

1.1.3.2 US Stations

If used in the US it's often required to get the KP value as Station instead. This can be achieved by inserting a 'S' or 's' character.

1.1.4 DOL item

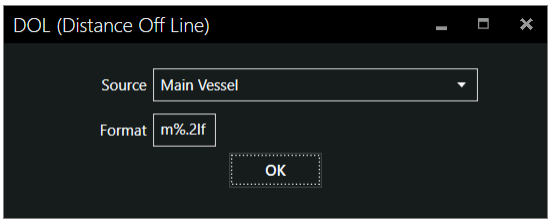
The User defined output offers an item called DOL (Distance Off Line) measured in metre (or current NaviPac distance unit).



It may also output the value called DCC (Distance Cross Course) instead. To do so, insert the character c (lower case) in front of the data format. The output is default in selected data unit (eg metric) – but you may select other formats if needed.

1.1.4.1 Forced to centimetre

Enter an 'm' in front of the format string.

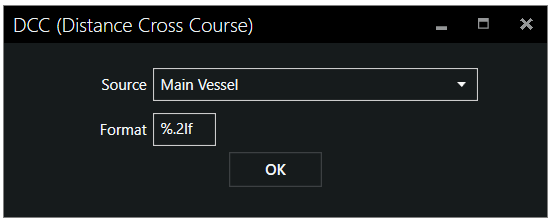


1.1.4.2 Forced to US Survey Feet

Enter an 'f' in front of the format string.

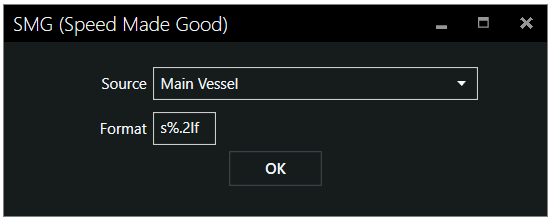
1.1.5 DCC item

The field called DCC (Distance Cross Course) is a special (and more stable) variant of the above DOL, as the definition depends on KP ascending direction (left-/+right) rather than on sailing (design) direction (left-/+right).



1.1.6 SMG item

The calculated speed SMG (Speed Made Good) can be output for a selected object.



The output will be knots per default, but you may choose other units by inserting a special character in front of the format.

- Km/h Type k in front – eg k%.2lf
- m/s Type s in front – eg s%.5lf
- m/min Type m in front – eg m%.5lf

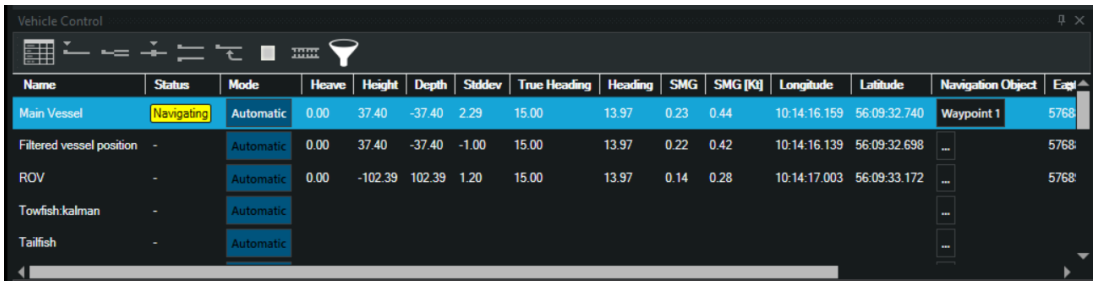
1.1.7 Range/Bearing to Waypoint item

The output includes two fields:

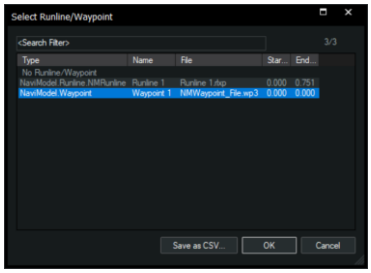
- Range to Waypoint
- Bearing to Waypoint (grid)

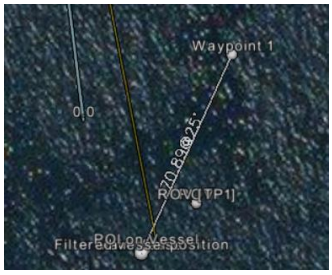
They are both position-based – meaning that the operator selects a vehicle to control the output.

An object must be controlling a waypoint in the Vehicle Control View. This is done by opening a Vehicle Control View in the master Helmsman's Display, and then selecting a waypoint in the Navigation Object column.



This applies for the items listed next regarding waypoints as well.





1.1.8 Distance (E/N) to Waypoint item

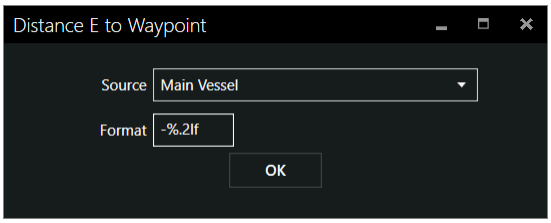
The output includes two fields:

- Distance Easting to Waypoint
- Distance Northing to Waypoint

They are both position-based – meaning that the operator selects an object to control the output.

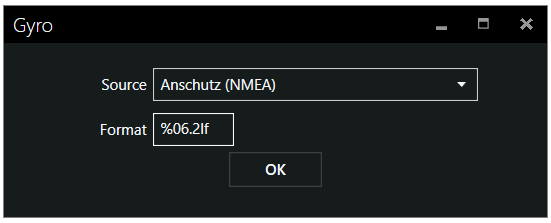
An object must be controlling a waypoint in the Vehicle Control View, as described in Range/Bearing to Waypoint.

The distance is calculated as position of waypoint minus position of object. To calculate the inverse, that is object position minus waypoint position, enter a minus in the format field.



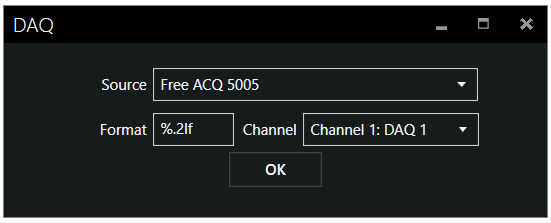
1.1.9 Gyro item

Which gyro to use and how the gyro values should be formatted can be selected.



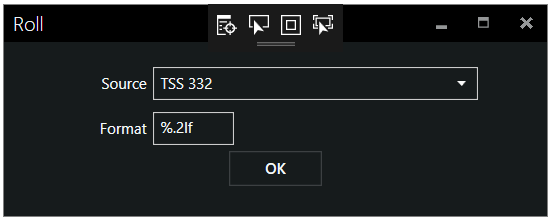
1.1.10 DAQ item

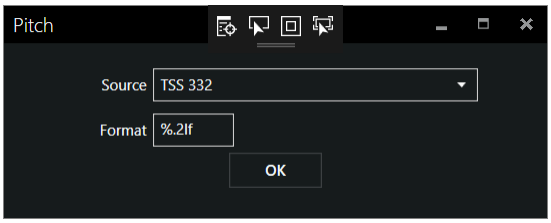
Which echo sounder depth or data acquisition channel should be output and in what format can be selected.

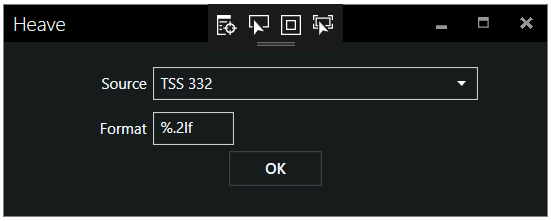


1.1.11 Motion (Roll/Pitch/Heave) item

If more than one motion sensor is available, a specific sensor can be selected along with which items (roll, pitch, heave) should be output by that sensor and how they should be formatted.

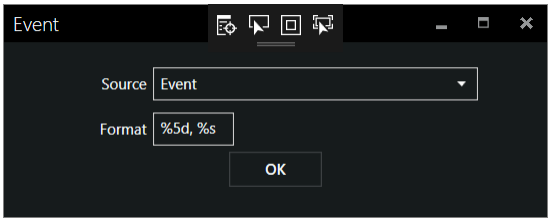






1.1.12 Event item

How the event number should be formatted can be selected.

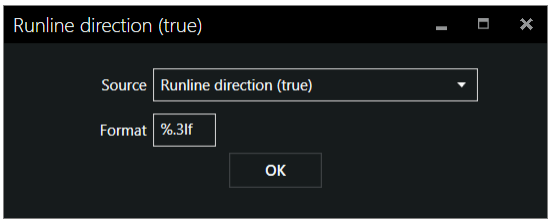


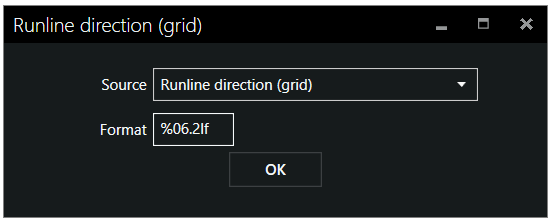
To output the event number enter eg %05d to use 5 digits with leading zeros. It is also possible to include event text (manual event and user defined events) by entering %05d,%s where %s represents the string with the event text.

1.1.13 Runline direction (true/grid) item

The current bearing of the active line segment (or tangent of curve intersection point) can be output as either true heading or grid heading (the latter is compensated for meridian convergence).

How many digits the line direction (heading) must contain can be specified.

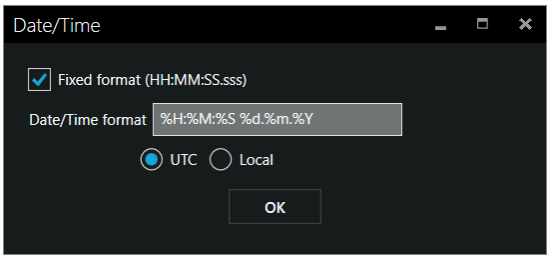


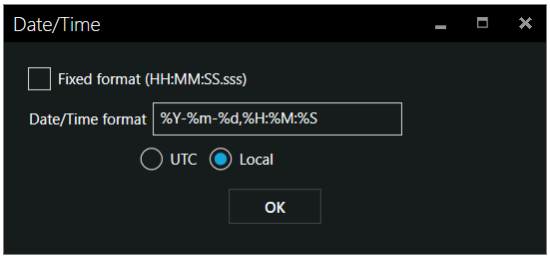


To output headings with leading zeros and two digits after the decimal point just set the format to %06.2lf.

1.1.14 Date & Time item

Defines the date and time of the observation. Note: This is not the data timestamp – but the timestamp of the NaviPac output cycle.





The system can be defined to output local time or UTC.

The date/time format can be defined using ANSI C format as explained in section 2.6. The timestamp will then be given in resolution of 1 second.

You may also select a timestamp in the fixed format of hh:mm:ss.sss which allows you to get time at a resolution of 1 ms.

Example:
Specifying: *%H:%M:%S %d.%m.%Y*

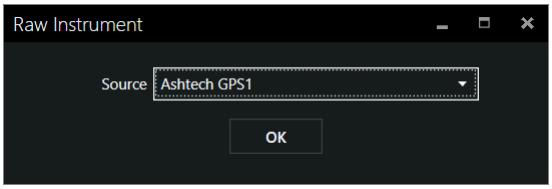
Will result in the following output string:

10:06:01 12.01.2020

If *Fixed* is selected the format will be HH:MM:SS.sss

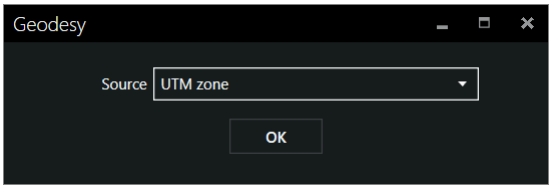
1.1.15 Raw instrument item

To output a copy of raw instrument data just select the instrument in action. This will not work for binary instruments.



1.1.16 Geodesy information

It is possible to output various information of the selected geodesy.

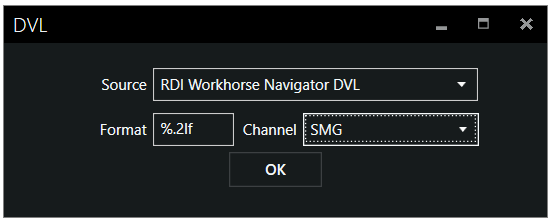


The following parameters will be available:

- Ellipsoid name
  Character string
- Semi major axis
  Floating point number
- Inverse flattening
  Floating point number
- Projection name
  Character string
- Scale at Origin /* Point scale factor at the ellipsoid origin */
  Character string
- First parallel /* First parallel in Lambert's conical - degree */
  ddd mm ss.ssss
- Second Parallel /* Second parallel in Lambert's conical - degree */
  ddd mm ss.ssss
- Longitude at Origin /* Origin lambda (lon) on the reference ellipsoid [degree] */
  ddd mm ss.ssss
- Latitude at Origin /* Origin phi (lat) on the projection ellipsoid [degree] */
  ddd mm ss.ssss
- False Easting /* Origin easting on the projection plane [metre] */
  Floating point
- False Northing /* Origin northing on the projection plane [metre] */
  Floating point
- UTM zone /* Derived from "Original Longitude" [1..60] */
  Integer

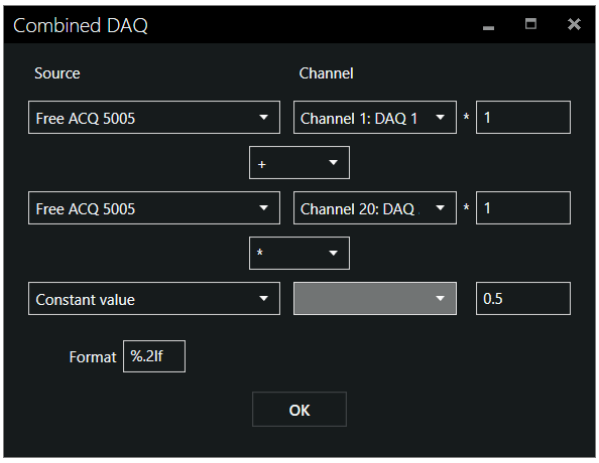
1.1.17 DVL (Speed log) item

If a Doppler log, eg an RDI Doppler log, is attached, then the system can output various parameters from the PD0 data from the Doppler log.



1.1.18 Combined Data Acquisition item

Via the Combined Data Acquisition item, you may combine two or three DAQ values such as depth and altitude or depth plus constant for a single output.



Select the instruments (the lists to the left) and channels for each of them. Finally, scale the inputs if they are in different data units. The special “instrument” called Constant value is used for entering a fixed number.

The following operators are available:

+	Plus
-	Minus
*	Multiply
/	Divide by
NO	None – the second source is ignored

The order of calculation will be from left to right (see below), defined as:

(value1 <operator> value2) <operator> value 3.

The above example gives the average value of DAQ1 plus DAQ20 (DAQ1+ DAQ20) *0.5.

1.2 Definition file

It must be noted that the final definition of user defined output is handled in some separate files that are outside the normal definitions of the NaviPac project file (.npp) file. These files are located on \EIVA\NaviPac\Setup and named userDef<index>.out (eg userDef611.out).

2. Custom output format strings

The format field for positions, depths etc follow the ANSI C syntax:

Format Specification Fields: printf and wprintf functions

A format specification, which consists of optional and required fields, has the following form:

%[flags] [width] [**.**precision] [{**h** | **l** | **I64** | **L**}]type

Each field of the format specification is a single character or a number signifying a particular format option. The simplest format specification contains only the percent sign and a *type* character (for example, `%s`). If a percent sign is followed by a character that has no meaning as a format field, the character is copied to **stdout**. For example, to print a percent sign character, use `%%`.

The optional fields, which appear before the *type* character, control other aspects of the formatting, as follows:

Type
Required character that determines whether the associated *argument* is interpreted as a character, a string, or a number (see Table R.3 printf type field characters).

Flags
Optional character or characters that control justification of output and printing of signs, blanks, decimal points, and octal and hexadecimal prefixes (see Table R.4 Flag characters). More than one flag can appear in a format specification.

Width
Optional number that specifies the minimum number of characters to be output (see printf width specification).

Precision
Optional number that specifies the maximum number of characters printed for all or part of the output field, or the minimum number of digits printed for integer values (see Table R.5 Precision values).

h | **l** | **I64** | **L**

Optional prefixes to *type* that specify the size of the *argument* (see Table R.6).

2.1 printf type field characters

The *type* character is the only required format field; it appears after any optional format fields. The *type* character determines whether the associated argument is interpreted as a character, string, or number. The types **C** and **S**, and the behaviour of **c** and **s** with **printf** functions are Microsoft extensions and are not ANSI-compatible.

Table R.3 printf type field characters

Character	Type	Output Format
c	int or wint_t	When used with printf functions, specifies a single-byte character; when used with wprintf functions, specifies a wide character.
C	int or wint_t	When used with printf functions, specifies a wide character; when used with wprintf functions, specifies a single-byte character.
d	int	Signed decimal integer.
i	int	Signed decimal integer.
o	int	Unsigned octal integer.
u	int	Unsigned decimal integer.
x	int	Unsigned hexadecimal integer, using “abcdef.”
X	int	Unsigned hexadecimal integer, using “ABCDEF.”
e	double	Signed value having the form [–]*d.dddd* e [*sign*]*ddd* where *d* is a single decimal digit, *dddd* is one or more decimal digits, *ddd* is exactly three decimal digits, and *sign* is + or –.
E	double	Identical to the e format except that E rather than e introduces the exponent.
f	double	Signed value having the form [–]*dddd.dddd*, where *dddd* is one or more decimal digits. The number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision.
g	double	Signed value printed in f or e format, whichever is more compact for the given value and precision. The e format is used only when the exponent of the value is less than –4 or greater than or equal to the precision argument. Trailing zeros are truncated, and the decimal point appears only if one or more digits follow it.
G	double	Identical to the g format, except that E, rather than e, introduces the exponent (where appropriate).
n	Pointer to integer	Number of characters successfully written so far to the stream or buffer; this value is stored in the integer whose address is given as the argument.
p	Pointer to void	Prints the address pointed to by the argument in the form *xxxx*:*yyyy*, where *xxxx* is the segment and *yyyy* is the offset, and the digits *x* and *y* are uppercase hexadecimal digits.

s	String	When used with printf functions, specifies a single-byte character string; when used with wprintf functions, specifies a wide character string. Characters are printed up to the first null character or until the *precision* value is reached.
S	String	When used with printf functions, specifies a wide character string; when used with wprintf functions, specifies a single-byte character string. Characters are printed up to the first null character or until the *precision* value is reached.

2.2 Flag directives

The first optional field of the format specification is *flags*. A flag directive is a character that justifies output and prints signs, blanks, decimal points, and octal and hexadecimal prefixes. More than one flag directive may appear in a format specification.

Table R.4 Flag characters

Flag	Meaning	Default
–	Left align the result within the given field width.	Right align.
+	Prefix the output value with a sign (+ or –) if the output value is of a signed type.	Sign appears only for negative signed values (–).
0	If *width* is prefixed with 0, zeros are added until the minimum width is reached. If 0 and – appear, the 0 is ignored. If 0 is specified with an integer format (i, u, x, X, o, d) the 0 is ignored.	No padding.
blank (' ')	Prefix the output value with a blank if the output value is signed and positive; the blank is ignored if both the blank and + flags appear.	No blank appears.
#	When used with the o, x, or X format, the # flag prefixes any non-zero output value with 0, 0x, or 0X, respectively.	No blank appears.
	When used with the e, E, or f format, the # flag forces the output value to contain a decimal point in all cases.	Decimal point appears only if digits follow it.
	When used with the g or G format, the # flag forces the output value to contain a decimal point in all cases and prevents the truncation of trailing zeros. Ignored when used with c, d, i, u, or s.	Decimal point appears only if digits follow it. Trailing zeros are truncated.

2.3 printf width specification

The second optional field of the format specification is the width specification. The *width* argument is a non-negative decimal integer controlling the minimum number of characters printed. If the number of characters in the output value is less than the specified width, blanks are added to the left or the right of the values, depending on whether the – flag (for left alignment) is specified, until the minimum width is reached. If *width* is prefixed with 0, zeros are added until the minimum width is reached (not useful for left-aligned numbers).

The width specification never causes a value to be truncated. If the number of characters in the output value is greater than the specified width, or if *width* is not given, all characters of the value are printed (subject to the precision specification).

If the width specification is an asterisk (*****), an **int** argument from the argument list supplies the value. The *width* argument must precede the value being formatted in the argument list. A nonexistent or small field width does not cause the truncation of a field; if the result of a conversion is wider than the field width, the field expands to contain the conversion result.

2.4 How precision values affect type

Table R.5 Precision values

Type	Meaning	Default
c, C	The precision has no effect.	Character is printed.
d, i, u, o, x, X	The precision specifies the minimum number of digits to be printed. If the number of digits in the argument is less than the *precision*, the output value is padded on the left with zeros. The value is not truncated when the number of digits exceeds the *precision*.	Default precision is 1.
e, E	The precision specifies the number of digits to be printed after the decimal point. The last printed digit is rounded.	Default precision is 6; if *precision* is 0 or the period (.) appears without a number following it, no decimal point is printed.
f	The precision value specifies the number of digits after the decimal point. If a decimal point appears, at least one digit appears before it. The value is rounded to the appropriate number of digits.	Default precision is 6; if *precision* is 0, or if the period (.) appears without a number following it, no decimal point is printed.
g, G	The precision specifies the maximum number of significant digits printed.	Six significant digits are printed, with any trailing zeros truncated.
s, S	The precision specifies the maximum number of characters to be printed. Characters in excess of *precision* are not printed.	Characters are printed until a null character is encountered.

If the argument corresponding to a floating-point specifier is infinite, indefinite, or NaN, **printf** gives the following output.

Value	Output
+ infinity	**1.#INF***random-digits*
– infinity	**–1.#INF***random-digits*
Indefinite (same as quiet NaN)	*digit.#***IND***random-digits*
NAN	*digit.#***NAN***random-digits*

2.5 Size prefixes for printf and wprintf format type specifiers

Table R.6 Size prefixes

To Specify	Use Prefix	Width Type Specifier
long int	l	d, i, o, x, or X
long unsigned int	l	u
short int	h	d, i, o, x, or X
short unsigned int	h	u
int64	l64	d, i, o, u, x, or X
Single-byte character with printf functions	h	c or C
Single-byte character with wprintf functions	h	c or C
Wide character with printf functions	l	c or C
Wide character with wprintf functions	l	c or C
Single-byte – character string with printf functions	h	s or S
Single-byte – character string with wprintf functions	h	s or S
Wide character string with printf functions	l	s or S
Wide character string with wprintf functions	l	s or S

Thus, to print single-byte or wide characters with **printf** functions and **wprintf** functions, use format specifiers as follows:

To Print Character As	Use Function	Width Format Specifier
single byte	printf	**c**, **hc**, or **hC**
single byte	wprintf	**C**, **hc**, or **hC**
wide	wprintf	**c**, **lc**, or **lC**
wide	printf	**C**, **lc**, or **lC**

To print strings with **printf** functions and **wprintf** functions**,** use the prefixes **h** and **l** similarly to format type specifiers **s** and **S**.

2.6 Date & Time format

The format argument consists of one or more codes; based on the C/C++ printf, the formatting codes are preceded by a percent sign (%). Characters that do not begin with % or # are copied unchanged to logfile or output. The formatting codes for date and time in custom logfiles are listed below:

%a Abbreviated weekday name
%A Full weekday name
%b Abbreviated month name
%B Full month name
%c Date and time representation appropriate for locale
%d Day of month as decimal number (01 - 31)
%H Hour in 24-hour format (00 - 23)
%I Hour in 12-hour format (01 - 12)
%j Day of year as decimal number (001 - 366): (=Julian day)
%m Month as decimal number (01 - 12)
%M Minute as decimal number (00 - 59)
%p Current locale's A.M./P.M. indicator for 12-hour clock
%S Second as decimal number (00 - 59)
#s Seconds past midnight
%U Week of year as decimal number, with Sunday as first day of week (00 - 51)
%w Weekday as decimal number (0 - 6; Sunday is 0)
%W Week of year as decimal number, with Monday as first day of week (00 - 51)
%x Date representation for current locale
%X Time representation for current locale
%y Year without century, as decimal number (00 - 99)
%Y Year with century, as decimal number
%z, %Z Time zone name or abbreviation; no characters if time zone is unknown

%% Percent sign