Tseries Technical Documentation

Table of Contents

Description:

Tseries is an open-source time-series viewer for meteorological purposes. Tseries is developed and implemented at the Norwegian Meteorological Institute met.no, as a part of the operational meteorological workstation. Tseries can be used standalone, or in interaction with the DIANA meteorological workstation.

The tseries package contains two programs:

1. *tseries.bin* is the timeseries viewer with a Graphical user interface
2. btseries is a batch version of tseries, using the same graphical engine

The Goal of this Document is to explain the basic structure of tseries setup- and stylefiles in order to adapt tseries to a specific environment. The intended readership for this document is it-personnel and meteorologists, in charge of the operational setup of a local tseries implementation. The description of the tseries GUI is integrated in the applications help menu. The btseries user interface is described in the “btseries –help” option.

Command line parameters:

Most of the input in tseries is given by the control file, but can be overridden by command line input:

-s [setupfile] if the setup file is not directly defined or invalid, tseries will try to find a valid setup file with the name *tseries.ctl* in the system. First looking in the current directory, then in $HOME/.tseries/ and at last in /etc/tseries. If no valid setupfile exists, tseries will exit.

-S [site] site is just a short string. The user can define elements in the setup-file that are only used by certain groups. By this, several groups of users can share one setup-file to simplify maintenance.

-l [lang] The language for the GUI. Can be configured in the GUI itself. Available languages are English(default), Norwegian Nynorsk, Norwegian Bokmål, Swedish, German, French. New languages can be added by using the qt-linguist tool.

-T [title] Changes the window title, useful if you are running several instances of tseries at the same time.

-H [wdbhost]

-u [wdbuser]

-d [section1:key1=token1 section2:key2=token2] You can override any entity in the setup file by this option. Example: You can use a different Diagrams definition file, without changing the setup file, by the option -d files:Defs=./tsDiagrams.def

Datastreams:

There are several possibilities to add a Datastream to a tseries Diagram:

1. **HDF** is the most common used datasource. In HDF all points are predefined and named, Hdf input is defined in the base control file *tseries.ctl.* All **HDF** streams mentioned in *tseries.ctl* are indexed at startup. Streams from different **HDF** files can be combined in a single Diagram.
2. **ASCII** is possible and quite easy to implement. But in an operational environment ASCII streams are space-consuming and ineffective. An usage example of ASCII streams is presented in the appendix of this Document.
3. **WDB** the Weather and Water Database WDB is an open-source database system based on postgres. The system is developed at met.no. **WDB** is holding data as fields. But you can retrieve point interpolated data directly into tseries. As **WDB** interpolates data on-demand, **WDB** is the most flexible datastream available, but it is slower than local **HDF** files. **WDB** is the dataengine for for the web-based, free weather service yr.no
4. **KDVH** is a met.no properitary datainterface, based on URL. **KDVH** reads observations from the Norwegian climatic database. The Data are defined as Diagram entries in the control file (*tseries.ctl*) and can be integrated in any given diagram.

Diagrams:

A Diagram in tseries consist of one or more datastreams connected to a given style. There are several files needed to generate a diagram:

If you want to generate a new diagram, you have to proceed the following way:

1. Create a new folder.
2. Copy a tseries.ctl and a tsDiagrams file into the folder.
3. Edit tseries.ctl, if the new diagram implements a new style and not a new stream, then just change the definition for **Defs** to the tsDiagrams file in this folder.
4. Copy a stylefile into the folder (for instance style.meteogram) rename the stylefile.
5. Edit tsDiagrams. Compose the parameters and models for the new diagram. Connect the diagram to the stylefile in this folder.
6. Edit the stylefile in this folder.
7. Start tseries.bin in this folder, the local tseries.ctl will be the default.
8. Repeat 6. and 7. until you are happy with the result.

The Main Control file – tseries.ctl

The control file is split into several sections. All parameters given in the start of the file can be used as substitutes by $(parameter) anywhere later in the file. Unix Environment variables can be substituted by ${parameter}.

Single lines can be commented out by “#”

Keys, Sites and sections are “case-independent”

Sites:

Tseries.ctl is a multi-user setup-file. To avoid the administration of several setup-files within your company, you can write several different configurations in the same file. The file is processed top-down. When a variable is defined several times, the last definition is valid.

Elements within sites in tseries are usually inactive, the parser will ignore them, unless the user starts tseries with the flag that defines the specific site (the -S flag). A site is defined by a colon separated list of strings in square brackets. A site section ends by another site or a set of empty brackets.

Example:

```
server=meat.com

[FRUIT]
server=banana.com

[VEGETABLES:OTHERS]
server=potato.com
[]
```

Basically the server is defined as “meat.com”, if you want to use the “banana.com” server, you have to start tseries by:

```
 tseries -S FRUIT
```

If you want to use potato.com, you can start tseries either with “ -S VEGETABLES” or “-S OTHERS”. This site is valid for for both flags.

Sections

Sections are defined by chevrons like “<Path>”. A section is valid until the next section starts. Anything in tseries.ctl is organised in sections from the beginning of the first section. Entities written before the first section starts are pushed into a hash table and can be used freely as substitutes through the entire setup file.

<Path>

Etc:	Path to find basic configurations
Images:	Path to find images used in diagrams, like weather symbols
Doc:	Path to the Documentation
Lang:	Path to find language files, tseries looks for tseries_XX.qm files in this directory, where XX is the country code. .qm files are generated by the the Qt linguist tool.

<Files>

Defs:	The diagram definition file
Configure:	Autogenerated File to store configuration (like window size, language etc) from the GUI.
WeatherSymbols:	The weather symbol definition file
stdImage:	Image to send to diana to show stations on the map
finImage:	Image to display in diana when the “find” button is triggered
iconImage:	The tseries icon
baseFilter:	A filter file. If you use the filter, tseries will only display stations defined in the filter. The filter can be altered and saved locally by the user.
Bookmarks:	In WDB mode you can add your own bookmarks (stations). They will be saved in this file
commonBookmarks:	A file with bookmarks given by your company, valid for all users

<Server>

The server section is used to define the connection to diana via coserver

Client:	What application is the target (Diana)
Name:	What name is sendt to the Client
Command:	What command is coserver using

<Streams>

The streams section is parsed by an older system, the pets graphical engine, on which tseries is build.
All data input from HDF files is defined here, always the same way.

A new model starts with a collection name, this is more for backwards-compatibility, these values are not used by tseries

CollectionName=ECMeteogrammer
PreferredDiagram=Meteogram
InitialOpen=0

In a collection, any datafile has to be registered the following way

DataFile:	The hdf file
DataDescription:	The data description. A hdf file can contain several streams. You have to find the exact stream name in the file (when in doubt use vshow). The data description is used to define values in the diagram definition.
DataType:	HDF or ASCII
Contents:	What parameter are defined in the file, x means any. The string is separated into four sections: Parameter,level,stream,runtime • **x,x,ECMWF,12** – means: All parameters from ECMWF, 12 • x,x,ECMWF,x -All parameters from ECMWF, all times

The Diagram definition file – tsDiagrams.def

The Diagram definition file ties the streams to the actual diagram styles. One model input can appear in several diagrams, as well as several different models can use the same diagram type (like a meteogram). It is also possible to use one parameter from a defined stream into a certain diagram that is used by another stream. An example is the use of wind speed from an atmospheric model into a wave-diagram.

Tseries builds its User interface based on this file. A stream appears first in the menus when it is defined here. If you want to remove a streams from tseries, you have to remove it from this file as well. Otherwise it will appear as an empty entry in the menus.

The [models] section

Any model have to be defined in the models section, it also allows to change the display name of the model in the diagrams. The key has to be equal the stream name from tseries.ctl like

```
ECMWF=Ecmwf
```

ECMWF is the name from tseries.ctl, and basically the stream definition In the HDF file, Ecmwf will be displayed in the menus

The Diagrams

Each diagram definition connects some parameters from some models to a style. A style can be used several times in different contexts. For instance is the S-meteogram (symbol meteogram) a diagram that uses the meteogram style, but it has also the weather parameter (WW) as input.

Each diagram defintion starts with the [DIAGRAM] keyword and ends with the [ADD] keyword.

Examples:

```
# Standard meteogram
[DIAGRAM]
# The name of the diagram as it appears in the menu
NAME=Meteogram
# The connected style file
STYLEFILE=/usr/share/tseries/5.3/style/style.meteogram
# The streams connected to this diagram
[LEGALMODELS]
MODEL=ECMWF
MODEL=HARMONIE_2.5km

# modelindependent means, that these parameters will be taken from
# all models, if available
[MODELINDEPENDENT]
PARAM=UU,x,x,x
PARAM=VV,x,x,x
PARAM=FF,x,x,x
PARAM=DD,x,x,x
[END]

# modelspecific means that these parameters are only used if they
# come from the specified model.
# modflspecifi values are not necessary part of the legalmodels.
# If they are not defined there, they will not occur in the menu

[MODELSPECIFIC]
MODEL=HARMONIE_2.5km
PARAM=TT_U,x,x,x
[END]

[MODELSPECIFIC]
MODEL=OBS
PARAM=TT,x,x,x
[END]

[ADD]
```

Styles

Styles are defined in stylefiles. These are connected to data by tsDiagrams.def.

By altering the stylesfiles you can control colors, line and object types in the diagram.

There are hundreds of different types in a stylefile. You have to start with an existing stylefile and reduce that given stylefile until it fits your purpose. A style starts with the definition of the standard values: like default patterns, background color for the diagram etc.

```
#================================================================
=
#
# Global settings
#
#================================================================
=


bgColor=      WHITE
topMargin=    50
bottomMargin= 50
leftMargin=   150
rightMargin=  150

# -------------------------------
# default values

type=         DUM_PRIMITIVE
parameter=    x,x,x,x
mother=       DIAGRAM
plotAll=      FALSE
enabled=      TRUE

color=        RED
color2=       YELLOW
font=         NORMAL
linePattern=  FULL
fillPattern=  SOLID
align=        LEFT
spacing=      MEDIUM
intSpacing=   SMALL
label=        TRUE
patternInColour= FALSE
lineWidth=    1.0
axisWidth=    2.0
tickWidth=    1.0
minRange=     0
interval=     1
delta=        2
```

```
minMargin=    0
maxIsSet=     FALSE
minIsSet=     FALSE
minValue=     100000
maxValue=    -100000
yaid=         0
centerVector= FALSE
numTickMajor= 10
numTickMinor= 2
labelSpace=   120
quantized=    FALSE
quantum=      1.0
gridxonly=    FALSE
axisgrid=     FALSE
gridwidth=    1
gridcolor=    GREY25
gridstyle=    DOTTED

[DEFAULT]
```

Then you have to define axes

first at least one x-axis

```
 # --------------------------------
# utc-time
[NEW]
type=     UTC
order=    8
height=   25
spacing=  SMALL
font=     NORMAL
color=    BLUE
minSkipX= 15
[ADD]
```

And then several y-axes. A curve in the diagram is connected to an axis. If you want to display, for instance several temperatures, you have to create a temperature axis. Then you have to connect the curves to that axis. By that it is guaranteed that the curves are scaled correctly to each other. Each axis has an unique identifier (y-axis-id/yaid). Yaid is used to connect parameters to the axis.

```
 # ----------------------------
# Yaxis (T2m)

[NEW]
type=        YAXIS_STATIC
yaid=        0
delta=       1.0
```

```
interval=    0.5
minMargin=   0.25
minRange=    5.0
minIsSet=    FALSE
maxIsSet=    FALSE
text=
align=       LEFT
axis=        LEFTLEFT
color=       RED
lineWidth=   2
linePattern= FULL
patternInColour=TRUE
[ADD]
```

With axis in place, you have to define a parameter and connect it to it. In this case, the curve is presented in 2 different colors, one for -100 to 0 degrees and one for 0-100 degrees.

```
 [NEW]
type=            LINE
yaid=            0
parameter=       TT,0,x,x
text=            T2m [°C]
lineWidth=       2
linePattern=     FULL
patternInColour= FALSE
colorbyvalue=    TRUE
datalimits=      -100,0,100
colorlist=       BLUE,RED
patternInColour= FALSE
[ADD]
```

Valid types:

There are several types in tseries that can be used by the different elements.

Colors:

- BLACK
- BLUE
- GREEN
- CYAN
- RED
- MAGENTA
- YELLOW
- WHITE
- GREY25
- GREY40
- GREY45
- GREY50
- GREY55
- GREY60
- GREY65
- GREY70
- GREY75
- GREY80
- GREY85
- GREY90
- GREY95

- MIST_RED
- MIST_GREEN
- MIST_BLUE
- DARK_GREEN
- BROWN
- ORANGE
- PURPLE
- LIGHT_BLUE
- DARK_YELLOW
- DARK_RED
- DARK_BLUE
- DARK_CYAN
- DARK_MAGENTA
- MIDNIGHT_BLUE
- DNMI_GREEN
- DNMI_BLUE
- RUST_RED
- DAWN_RED
- SUN_YELLOW
- SPRING_GREEN
- IRR_GREEN
- MOSS_GREEN
- GRASS_GREEN
- THUNDER_GREY
- SEA_BLUE
- SKY_BLUE
- ICE_BLUE
- GLACIER_WHITE
- RAIN_GREY
- IVORY
- DARK_IVORY
- BLUEGREY
- GREYBROWN
- OLIVE
- MUDDYGREEN
- GREYBLUE
- DARK_BLUEGREY
- DARKOLIVE
- RED_YELLOW
- YELLOW_RED

Fillpatterns

- DIAGRIGHT
- DIAGLEFT
- DIAGCROSS
- HORIZONTAL
- VERTICAL
- SHORIZONTAL
- SVERTICAL
- SQUARE
- SOLID

Linestyle

- DASHED
- DASHDOTTED
- DASHDASHDOTTED
- DOTTED

Appendix

Appendix 1 – defining an ASCII stream

An ASCII stream can be integrated in any diagram. In this example – an observation is added to a standard diagram by ascii.

First, you need the ASCII stream itself: (here mypos.txt)

```
Model=OBS

Position=MYPOS,60.3,10.1
TIME                  LEVEL   TT_O   MSLP_O  FF   DD    CC
2013-03-12 00:00:00   0       18.0   999.1   3.2  179   4
2013-03-12 02:00:00   0       15.9   999.1   5.1  280   5
```

You can use one file for all timeseries or one file per timeseries. If you have several files you have to register all of them in tseries.ctl, but from that point they will be processed equally. Tseries builds the streams from the file and sorts them under the station name. If You use the same station name from another model, tseries will mix the ascii stream into the diagram.
In tseries.ctl, this is registered by:

```
CollectionName=Observation
PreferredDiagram=Meteogram
InitialOpen=0

DataFile=mypos.txt
DataDescription=OBS
DataType=ASCII
Contents=x,x,OBS,x
```

Now you have a stream (OBS) with the parameters TT_O, MSLP_O,FF,DD and CC. I used different names for TT and MSLP to add different styles to them (thinner lines).
This stream can be added to a given diagram in tsDiagrams.def

```
[MODELS]

ECMWF=Ecmwf
OBS=Obs


# Standard meteogram
[DIAGRAM]
NAME=Meteogram
STYLEFILE=style.meteogram
```

```
[LEGALMODELS]
MODEL=ECMWF

[MODELINDEPENDENT]
PARAM=UU,x,x,x
PARAM=VV,x,x,x
PARAM=FF,x,x,x
PARAM=DD,x,x,x
PARAM=WVFD,x,x,x
PARAM=TT,x,x,x
PARAM=MSLP,x,x,x
PARAM=RR1,x,x,x
PARAM=RR2,x,x,x
PARAM=RR,x,x,x
PARAM=FG,x,x,x
PARAM=CL,x,x,x
PARAM=CM,x,x,x
PARAM=CH,x,x,x
PARAM=CC,x,x,x
[END]

[MODELSPECIFIC]
MODEL=OBS
PARAM=TT_O,x,x,x
PARAM=MSLP_O,x,x,x
PARAM=DD,x,x,x
PARAM=FF,x,x,x
PARAM=CC,x,x,x
PARAM=WVFD,x,x,x
[END]

[ADD]
```

All parameters that are defined in style.meteogram anyway will be displayed immediately. The others have to be defined as seperatly.