Survex

Release 1.4.17

Mar 20, 2025

CONTENTS

This is the manual for Survex - a Free and Open-Source Software package for cave surveyors.

CHAPTER

ONE

INTRODUCTION

This section describes what Survex is, and outlines the scope of this manual.

1.1 About Survex

Survex is a multi-platform Free and Open-Source Software cave surveying package. Versions 1.2 and later run on Linux, other UNIX-like platforms, Microsoft Windows, and macOS. We're investigating support for phones and tablets.

We are well aware that not everyone has access to super hardware - often surveying projects are run on little or no budget and any computers used are donated. We aim to ensure that Survex is feasible to use on low-spec machines. Obviously it won't be as responsive, but we intend it to be usable. Please help us to achieve this by giving us some feedback if you use Survex on a slow machine.

Survex is capable of processing extremely complex caves very quickly and has a very effective, real-time cave viewer which allows you to rotate, zoom, and pan the cave using mouse or keyboard. We have tested it extensively using CUCC and ARGE's surveys of the caves under the Loser Plateau in Austria (over 90,000 survey legs and over 350km of underground survey data). This can all be processed in around 13 seconds on a computer bought in 2012.

Survex is also used by many other survey projects around the world, including the Ogof Draenen survey, the Easegill resurvey project, the OFD survey, the OUCC Picos expeditions, and the Hong Meigui China expeditions.

Survex is still actively being worked on. Version 1.0 was complete in some sense, but development has continued since.

We encourage feedback from users on important features or problems, which will help to direct future development. See the "Mailing List" section of this manual for the best way to contact us.

1.2 About this Manual

If there's a part of this manual you find hard to understand, please do let us know. We already know Survex well, so it can be hard for us to spot areas where the manual doesn't given enough information, or doesn't explain things clearly enough to follow when you don't know what's going on. It's helpful is you can suggest a better wording, but don't worry if you can't, just explain the problem as precisely as you can.

The master version of this manual is maintained as reStructured Text, and automatically converted to a number of other formats. If you are going to send us *major* changes, it's much easier to include them if you work from this master version rather than changing one of the generated files.

1.2.1 Terminology

Throughout this document we use British terminology for surveying:

station

a point in the cave that you survey from and/or to

leg
a line joining two stations

survey
a group of legs surveyed on the same trip

CHAPTER

TWO

GETTING STARTED

This section covers how to obtain the software, and how to unpack and install it, and how to configure it.

2.1 Obtaining Survex

The latest version is available from the Survex website. It is freely redistributable, so you are welcome to get a copy from someone else who has already downloaded it, and you can give copies to others.

If you want some sample data to experiment with, you can download some from the Survex website too: https://survex.com/software/sample.tar.gz

2.2 Installing Survex

The details of installation depend greatly on what platform you are using, so there is a separate section below for each platform.

2.2.1 Linux

Pre-built versions of Survex are available for some Linux distributions. See the Survex for Linux download page on our website for up-to-date information.

You'll need root access to install these prebuilt packages. If you don't have root access you will need to build from source (see the next section).

2.2.2 macOS

The easiest way to install a recent release of Survex on macOS is by using the Homebrew package manager. If you don't already use Homebrew, you'll need to install it first. See the macOS download page on the website for installation instructions.

2.2.3 Other versions of UNIX

For other UNIX versions you'll need to get the source code and compile it on your system. Unpack the sources and read the file called INSTALL in the top level for details about building from source.

2.2.4 Microsoft Windows

This version comes packaged with an installation wizard. Just run the downloaded installer package and it will lead you through the installation process. Since Survex 1.4.9, this pre-built version requires a 64-bit version of Microsoft Windows 7 or later.

Survex 1.4.8 and later support installing for all users (which requires administrator rights) or just for the current user (which doesn't). If installed for just the current user, other user accounts won't see the file associations, menu entries, desktop icons, etc for Survex.

Note that if you have an existing installation the installer will see it and try to upgrade it, and if that installation was done with administrator rights (which any installation of 1.4.7 or earlier will be) you'll also need administrator rights to upgrade. To change to a non-admin installation you need to first uninstall the existing admin install (which will need admin rights) then install the new version.

The survey viewer that's part of Survex is called aven, and uses OpenGL for 3d rendering.

If you find that 3D rendering is sometimes very slow (e.g. one user reported very slow performance when running full screen, while running in a window was fine) then try installing the OpenGL driver supplied by the manufacturer of your graphics card rather than the driver Microsoft supply.

The installer creates a Survex group in the Programs sub-menu of the Start menu containing the following items:

- Aven
- Documentation
- Uninstall Survex

  Icons are installed for `.svx`, `.3d`, `.err`, and `.pos` files, and also for Compass Plot files (`.plt` and `.plf`) (which Survex can read). Double-clicking on a `.svx` file loads it for editing. To process it to produce a `.3d` file, right click and choose "Process" from the menu - this runs aven to process the `.svx` file and automatically load the resultant `.3d` file. All the Survex file types can be right clicked on to give a menu of possible actions.

  `.svx`

  Process
  Process file with aven to produce `.3d` file (and `.err` file)

  `.3d`

  Open
  Load file into Aven

  Print
  Print the file via Aven

  Extend
  Produce extended elevation

  Convert to DXF
  This entry used to be provided to allow converting to a DXF file (suitable for importing into many CAD packages) but this functionality is now available from inside Aven with the ability to control what is exported, and this entry was dropped in 1.2.35.

  Convert for hand plotting
  This entry used to be provided to allow converting to a `.pos` file listing all the stations and their coordinates, but this functionality is now available from inside Aven with the ability to control what is exported, and this entry was dropped in 1.2.35.

  `.err`

  Open
  Load file into Notepad

  Sort by Error
  Sort `.err` file by the error in each traverse

  Sort by Horizontal Error
  Sort `.err` file by the horizontal error in each traverse

Sort by Vertical Error
Sort `.err` file by the vertical error in each traverse

Sort by Percentage Error
Sort `.err` file by the percentage error in each traverse

Sort by Error per Leg
Sort `.err` file by the error per leg in each traverse

2.3 Configuration

2.3.1 Selecting Your Preferred Language

Survex has extensive internationalisation capabilities. The language used for messages from Survex and most of the libraries it uses can be changed. By default this is automatically picked up from the language the operating system is set to use (from "Regional Settings" in Control Panel on Microsoft Windows, from the LANG environment variable on UNIX). If no setting is found, or Survex hasn't been translated into the requested language, UK English is used.

However you may want to override the language manually - for example if Survex isn't available in your native language you'll want to choose the supported language you understand best.

To do this, you set the `SURVEXLANG` environment variable. Here's a list of the codes currently supported:

Code	Language
en	International English
en_US	US English
bg	Bulgarian
ca	Catalan
de	German
de_CH	Swiss German
el	Greek
es	Spanish
fr	French
hu	Hungarian
id	Indonesian
it	Italian
pl	Polish
pt	Portuguese
pt_BR	Brazillian Portuguese
ro	Romanian
ru	Russian
sk	Slovak
zh_CN	Chinese (Simplified)

Here are examples of how to set this environment variable to give messages in French (language code `fr`):

Microsoft Windows
For MS Windows proceed as follows (this description was written from MS Windows 2000, but it should be fairly similar in other versions): Open the Start Menu, navigate to the Settings sub-menu, and open Control Panel. Open System (picture of a computer) and click on the Advanced tab. Choose `Environmental Variables`, and create a new one: name `SURVEXLANG`, value `fr`. Click `OK` and the new value should be effective immediately.

UNIX - sh/bash
```
SURVEXLANG=fr ; export SURVEXLANG
```

UNIX - csh/tcsh

```
setenv SURVEXLANG fr
```

If Survex isn't available in your language, you could help out by providing a translation. The initial translation is likely to be about a day's work; after that translations for new or changed messages are occasionally required. Contact us for details if you're interested.

2.4 Using Survex

Most common tasks can now be accomplished through `aven` - processing survey data, viewing the processed data, printing, exporting to other formats, and producing simple extended elevations.

A few tasks still require you to use the command line; some functionality which is available via `aven` is also available from the command line, which allows it to be scripted.

The command line programs that come with Survex are:

`cavern`
: Processes survey data. Since Survex 1.2.3 you can process `.svx` files by opening them with `aven`, so you don't need to use `cavern` from the command line if you don't want to, but it's still available for users who prefer to work from the command line and for use in scripts.

`diffpos`
: Compares the positions of stations in two processed survey data files (`.3d`, `.pos`, `.plt`, etc).

`dump3d`
: Dumps out a list of the items in a processed survey data file (`.3d`, `.plt`, etc). `dump3d` was originally written for debugging, but can also be useful if you want to access processed survey data from a script.

`extend`
: Produces extended elevations - this is probably the most useful of these command line tools. Since Survex 1.2.27 you can produce simple extended elevations from `aven` using the "Extended Elevation" function. However the command line tool allows you to specify a spec file to control how the survey is extended, which you can't currently do via `aven`.

`sorterr`
: Reorders a .err file by a specified field.

`survexport`
: Provides access to `aven`'s "Export" functionality from the command line, which can be useful in scripts. Added in Survex 1.2.35.

CHAPTER

THREE

SURVEX PROGRAMS

This section describes command-line use of Survex. Our aim is to make all functionality available without needing to use the command line (though we aren't quite there currently - the main thing lacking is more complex use of `extend`). We also aim to give access from the command line to any functionality which it is useful to be able to use from scripts, so that users who do like to use the command line can take full advantage of it.

3.1 Standard Options

All Survex programs support to the following command line options:

`--help`
display option summary and exit

`--version`
output version information and exit

Tools which take a processed survey data file to read also provide a way to specify the prefix of a sub-survey to restrict reading to:

`-s`, `--survey`=*SURVEY*
Only load the sub-survey *SURVEY*.

3.2 Short and Long Options

Options have two forms: short (a dash followed by a single letter e.g. `cavern -q`) and long (two dashes followed by one or more words e.g. `cavern --quiet`). The long form is generally easier to remember, while the short form is quicker to type. Options are often available in both forms, but more obscure or potentially dangerous options may only have a long form.

Note

Command line options are case sensitive, so `-B` and `-b` are different (this didn't used to be the case before Survex 0.90). Case sensitivity doubles the number of available short options (and is also the norm on UNIX-like platforms).

3.3 Filenames on the Command Line

Filenames with spaces can be processed (provided your operating system supports them - UNIX does, and so do modern versions of Microsoft Windows). You need to enclose the filename in quotes like so:

```
cavern "Spider Cave"
```

A file specified on the command line of any of the Survex suite of programs will be looked for as specified. If it is not found, then the file is looked for with the appropriate extension appended, so the command above will look first for `Spider Cave`, then for `Spider Cave.svx`.

3.4 Command Reference

3.4.1 aven

SYNOPSIS

> `aven` [–survey=*SURVEY*] [–print] *SURVEY_FILE*

DESCRIPTION

Aven displays processed cave surveys in a window and allows you to manipulate the view.

If `SURVEY_FILE` is an unprocessed survey data format which `cavern` can process, then `aven` will run `cavern` on it, and if successful, display the processed data. If there are any warnings and errors, it will show a log window with the output with clickable links to open the affected file at the problematic line.

`SURVEY_FILE` can also be processed survey data - a Survex `.3d` file, a Compass `.plt` file or a CMAP `.sht` file. It can also be a Survex `.pos` file or a CMP `.una` or `.adj` file, but for these only stations are shown, not any legs (for `.pos` this is because the format only records station positions). (All Survex programs which read `.3d` files can also transparently handle these formats.)

On-Screen Indicators

There is an auto-resizing scale bar along the bottom of the screen which varies in length as you zoom in or out. You can left-button drag on this to zoom, and right click gives a menu to select units or hide the scale bar (to get it back go to the Control->Indicators menu from the menu bar).

In the lower right corner is a compass indicator showing which way is North. You can drag this to rotate the view; if while dragging you move the mouse outside the compass the view snaps to 45° positions: N/NE/E/SE/S/SW/W/NW. Right click gives menu to change the view to N/S/E/W, select units, or hide the compass (to get it back go to the Control->Indicators menu from the menu bar).

Just to the left of the compass is clino indicator showing the angle of tilt. You can drag this to tilt the view; if while dragging you move the mouse outside the clino the view snaps to 90° positions: plan/elevation/inverted plan. Right click gives menu to change the view to plan/elevation, select units, or hide the clino (to get it back go to the Control->Indicators menu from the menu bar).

In the upper right is a colour key showing the correspondence between colour and depth (by default - you can also colour by other criteria). Right click gives a menu to choose what to colour by, select units, or hide the colour key (to get it back go to the Control->Indicators menu from the menu bar).

Mouse Control

Using the mouse to move the cave will probably feel most natural. We suggest you try each of these out after reading this section to get a feel for how they work.

If you hold down the right button then the view is panned when you move the mouse - it effectively feels like you are dragging the cave around.

If you hold down the left button, then the view is rotated as you move left or right, and zoomed as you move up and down. If you hold down `Ctrl` while dragging with the left mouse button, then moving up and down instead tilts the view. Tilt goes 180 degrees from plan view through elevation view to a view from directly below (upside down plan) - aven deliberately doesn't allow going beyond horizontal into an inverted view.

If your mouse has a middle button then holding it down and moving the mouse up and down tilts the cave. Moving the mouse left and right has no effect.

And if you have a scrollwheel, this can be used to zoom in/out.

By default the mouse moves the cave, but if you press `Ctrl-R`, then the mouse will move the viewpoint instead (i.e. everything will go in the opposite direction). Apparently this feels more natural to some people.

Keyboard Control

As with mouse controls, a little experimentation should give a better understanding of how these work.

All keyboard shortcuts have a corresponding menu items which should show the keyboard shortcut - this provides a way within the application to see the keyboard shortcut for a particular action.

`Delete` is useful if you get lost! It resets the scale, position, and rotation speed, so that the cave returns to the centre of the screen. There are also keyboard controls to use

`P` and `L` select Plan and eLevation respectively. Changing between plan to elevation is animated to help you see where you are and how things relate. This animation is automatically disabled on slow machines to avoid user frustration. You can force skipping the animation by pressing the key again during it, so a double press will always take you there quickly.

`Space` toggles on and off automatic rotation about a vertical axis through the current centre point (which is moved by panning the view or by selecting a station or survey). `R` toggles the direction of auto-rotation. The speed of auto-rotation can be controlled by `Z` and `X`.

Crosses and/or labels can be displayed at survey stations. `Ctrl-X` toggles crosses and `Ctrl-N` station names. `Ctrl-L` toggles the display of survey legs and `Ctrl-F` of surface survey legs.

`Ctrl-G` toggles display of an auto-sizing grid.

`Ctrl-B` toggles display of a bounding box.

`O` toggles display of non-overlapping/all names. For a large survey turning on overlapping names will make update rather slow.

Holding down `Shift` accelerates all the following movement keys:

The cursor keys pan the survey view (like dragging with the right mouse button).

`Ctrl` plus cursor keys rotate and tilt (like the mouse left button with `Ctrl` held down). `C`/`V` do the same as `Ctrl` plus cursor left/right, while Apostrophe `'`, and Slash `/` do the same as `Ctrl` plus cursor up/down.

`[` and `]` zoom out and in respectively.

OPTIONS

`-p, --print`
: Load the specified file, open the print dialog to allow printing, then exit.

`-s, --survey`=*SURVEY*
: Only load the sub-survey *SURVEY*.

`--help`
: display short help and exit

`--version`
: output version information and exit

3.4.2 cavern

SYNOPSIS

`cavern` [*OPTIONS*] *SURVEY_DATA_FILE*…

DESCRIPTION

`cavern` is the Survex data processing engine.

`cavern` is a command line tool, but if you're not a fan of working from the command line you can open unprocessed survey data files with `aven` and it will run `cavern` for you, and if successful, display the processed data. If there are any warnings and errors, `aven` will show a log window with the output with clickable links to open the affected file at the problematic line.

If multiple survey data files are listed on the command line, they are processed in order from left to right. Settings are reset to their defaults before processing each file.

Each *SURVEY_DATA_FILE* must be unprocessed survey data in a format which Survex supports, either native format (`.svx`) or Compass format (`.mak`, `.dat` or `.clp`), or Walls format (`.wpj` or `.srv`).

Support for Compass `.clp` was added in Survex 1.4.6; support for Walls was added in Survex 1.4.9.

OPTIONS

-o, --output=*OUTPUT*
Sets location for output files.

-q, --quiet
Only show a brief summary (`--quiet --quiet` or `-qq` will display warnings and errors only).

-s, --no-auxiliary-files
do not create .err file.

-w, --warnings-are-errors
turn warnings into errors.

--log
Send screen output to a .log file.

-v, --3d-version=*3D_VERSION*
Specify the 3d file format version to output. By default the latest version is written, but you can override this to produce a 3d file which can be read by software which doesn't understand the latest 3d file format version. Note that any information which the specified format version didn't support will be omitted.

--help
display short help and exit

--version
output version information and exit

OUTPUT

If there were no errors during processing, cavern produces two output files, with the extensions `.3d` and `.err` (unless `--no-auxiliary-files` is specified in which case only the `.3d` file is produced).

These two files are always created with their respective extensions. By default they are created in the current directory, with the same base filename as the first *SURVEY_DATA_FILE* listed on the command line.

E.g. if you process the data file `entrance.svx` with the command `cavern entrance` or `cavern entrance.svx` then the files `entrance.3d` and `entrance.err` will be created.

You can change the directory and/or base filename using the `--output` command line option. If you specify a directory then output files will go there instead of the current directory, but still use the basename of the first *SURVEY_DATA_FILE*. If you specify a filename which is not a directory (note that it doesn't need to actually exist as a file) then the directory this file is in is used, and also the basename of the filename is used instead of the basename of the first *SURVEY_DATA_FILE*.

Details of the output files:

`.3d`

This is a binary file format containing the adjusted survey data and associated meta data.

`.err`

This is a text file which contains statistics about each traverse in the survey which is part of a loop. It includes various statistics for each traverse:

Original length

This is the measured length of the traverse (for a "normal" or "diving" survey this is the sum of the tape readings after applying calibration corrections).

Number of legs

The number of survey legs in the traverse

Moved

How much one end of the traverse moved by relative to the other after loop closure

Moved per leg

Moved / Number of legs

Percentage error

(*Moved / Original length*) as a percentage. This seems to be a popular measure of how good or bad a misclosure is, but it's a problematic one because a longer traverse will naturally tend to have a lower percentage error so you can't just compare values between traverses. We recommend using the *E*, *H* and *V* values instead.

Error (*E*)

This isn't labelled in the *.err* file but is the value on a line by itself. In `aven` it's the value used by *Colour by Error*. It is *Moved* divided by the standard deviation for the traverse based on the standard errors specified for the instruments. This tells us how plausible it is that the misclosure is just due to random errors. It is a number of standard deviations, so the 68-95-99.7 rule applies - e.g. approximately 99.7% of traverses should have a value of 3.0 or less (assuming the specified instrument standard deviations are realistic).

Horizontal Error (*H*)

This is like *E* but only considers the horizontal component. In `aven` it's the value used by *Colour by Horizontal Error*. You can identify suspect traverses by looking at *E* and then compare *H* and *V* to see what sort of blunder might explain the misclosure. For example, if *H* is small but *V* is large it could be a clino reading or plumb with an incorrect sign, or a tape blunder on a plumbed leg; if *H* is large but *V* is small it could be a compass blunder, or a tape blunder of a nearly-flat leg.

Vertical Error (*V*)

This is like *E* but only considers the vertical component. In `aven` it's the value used by *Colour by Vertical Error*.

This information is now also present in the `.3d` file so you can view the survey coloured by these errors, but the `.err` file can still be useful as you can sort it using `sorterr` to get a ranked list of the sections of survey with the worst misclosure errors.

Cavern also reports a range of statistics at the end of a successful run:

- The highest and lowest stations and the height difference between them

- The East-West and North-South ranges, and the Northernmost, Southernmost, Easternmost, and Westernmost stations.
- The total length of the survey (before and after adjustment). This total excludes survey legs flagged as `SURFACE`, `DUPLICATE`, or `SPLAY`.
- The number of stations and legs. Note that a `*equate` is counted as a leg in this statistic.
- The number of each size of node in the network (where size is number of connections to a station) i.e. a one node is the end of a dead-end traverse, a two-node is a typical station in the middle of a traverse, a three-node is a T-junction etc.
- How long the processing took and how much CPU time was used.

If you successfully processed your data by loading it into `aven` then you can see this log output by using `File->Show Log` (also available as an icon in the toolbar).

Error Messages

There are many different error messages that you can get when processing data. Along with the error message, a location is reported. For an error like "file not found" this only reports the filename, but usually it will give the filename and line number of the offending line, and in many cases also an offset or span within the line.

The format of the location data follows that used by the GCC compiler so if your text editor can parse errors from GCC then you should be able to set it to allow you to jump to the file and line of each error.

One common cause of errors and warnings are typing mistakes. Another is your survey data not being all attached to fixed points (which is a warning since Survex 1.4.10, but was an error prior to this; in this situation, Survex will list at least one station in each piece of survey data which is not connected).

We try to make error and warning messages self-explanatory, but welcome feedback on cases where you get a message which seems unclear.

Generally you want to look at the first reported error first as there can be a cascade effect where one error triggers another. Cavern will stop after more than 50 errors. This usually indicates something like the incorrect data order being specified and deluging the user with error messages in such cases usually makes the actual problem less clear.

3.4.3 diffpos

SYNOPSIS

`diffpos` *FILE1 FILE2* [*THRESHOLD*]

DESCRIPTION

Diffpos reports stations which are in one file but not the other, and also stations which have moved by more than a specified threshold distance in X, Y, or Z. *THRESHOLD* is a distance in metres and defaults to 0.01m if not specified.

Note that the input files can be any format the "img" library can read (and can be different formats), so it works with Survex `.3d` and `.pos` files, Compass `.plt` and `.plf` files, CMAP `.sht`, `.adj` and `.una` files.

OPTIONS

`--help`
display short help and exit

`--version`
output version information and exit

3.4.4 dump3d

SYNOPSIS

`dump3d` [–survey=*SURVEY*] [–rewind] [–show-dates] [–legs] *INPUT_FILE*

DESCRIPTION

Dump out the entries in a processed survey data file - useful for debugging, and also provides a textual format which is fairly easy to parse if you want to write a simple script to pull out information from such files.

Don't be mislead by the "3d" in this tool's name - it can be used to dump a file in any format the "img" library can read, so it works with Survex `.3d` and `.pos` files, Compass `.plt` and `.plf` files, CMAP `.sht`, `.adj` and `.una` files.

If you're parsing the output in a script, you may want to use option `-legs` so you get a `LEG` item for each leg with from and to coordinates instead of each traverse being a `MOVE` item followed by a series of `LINE` items.

The `--show-dates` option uses `.` as the separator between date components by default (e.g. `2024.12.01`), but (since Survex 1.4.13) you can specify a different separator. If this separator is `.` then `-` is used between two dates which form a range, otherwise a space is used. For convenience, `-D` is provided as a short-cut for `--show-dates=-` which outputs dates in the ISO date format.

(The `--rewind` option is only provided to allow debugging and testing the `img_rewind()` function.)

OPTIONS

`-s`, `--survey`=*SURVEY*
only load the sub-survey with this prefix

`-r`, `--rewind`
rewind file and read it a second time

`-d`, `--show-dates[=SEPARATOR]`
show survey date information (if present)

`-D`
equivalent to –show-dates=-

`-l`, `--legs`
convert MOVE and LINE into LEG

`--help`
display short help and exit

`--version`
output version information and exit

3.4.5 extend

SYNOPSIS

`extend` [–survey=*SURVEY*] [–specfile=*ESPEC_FILE*] [–show-breaks] *INPUT_FILE* [*OUTPUT_3D_FILE*]

DESCRIPTION

`INPUT_FILE` can be a Survex `.3d` file, a Compass `.plt` file or a CMAP `.sht` file (all Survex programs which read `.3d` files can also transparently handle these formats).

If no `--specfile` option (or short option `-p`) is given, extend starts with the highest station marked as an entrance which has at least one underground survey leg attached to it. If there are no such stations, the highest deadend station in the survey (or the highest station if there are no deadends) is used. Extend puts the first station on the left, then folds each leg out individually to the right, breaking loops arbitrarily (usually at junctions).

If the output filename is not specified, extend bases the output filename on the input filename, but replacing the extension with `_extend.3d`. For example, `extend deep_pit.3d` produces an extended elevation called `deep_pit_extend.3d`.

The `--survey=`*SURVEY* option (short option `-s`) restricts processing to the survey *SURVEY* including any sub-surveys.

If you pass `--show-breaks` (short option `-b`) then a leg flagged as "surface survey" will be added between each point at which a loop has been broken - this can be very useful for visualising the result in aven.

This approach suffices for simple caves or sections of cave, but for more complicated situations human intervention is required. More complex sections of cave can be handled with a specfile giving directions to switch the direction of extension between left and right, to explicitly specify the start station, or to break the extension at particular stations or legs.

The specfile is in a format similar to cavern's data format:

```
; This is a comment

; start the elevation at station entrance.a
*start entrance.a  ;this is a comment after a command

; start extending leftwards from station half-way-down.5
*eleft half-way-down.5

; change direction of extension at further-down.8
*eswap further-down.8

; extend right from further-down.junction, but only for
; the leg joining it to very-deep.1, other legs continuing
; as before
*eright further-down.junction  very-deep.1

; break the survey at station side-loop.4
*break side-loop.4

; break survey at station side-loop.junction but only
; for leg going to complex-loop.2
*break side-loop.junction complex-loop.2
```

This approach requires some trial and error, but gives useful results for many caves. The most complex systems would benefit from an interactive interface to select and view the breaks and switches of direction.

3.4.6 sorterr

SYNOPSIS

sorterr [*OPTIONS*] *ERR_FILE* [*HOW_MANY*]

DESCRIPTION

`sorterr` re-sorts a .err file by the specified criterion (or by the error ratio by default). Output is sent to stdout, or if `--replace` (short option `-r`) is specified the input file is replaced with the sorted version. By default all entries in the file are included - if a second parameter is given then only the top `HOW_MANY` entries after sorting are returned.

OPTIONS

-h, --horizontal
sort by horizontal error factor

-v, --vertical
sort by vertical error factor

-p, --percentage
sort by percentage error

-l, --per-leg
sort by error per leg

-r, --replace
replace .err file with re-sorted version

--help
display short help and exit

--version
output version information and exit

3.4.7 survexport

SYNOPSIS

`survexport` [*OPTIONS*] *INPUT_FILE* [*OUTPUT_FILE*]

DESCRIPTION

The input formats supports are all those supported by Survex's "img" library - Survex .3d, Survex .pos, Compass PLT and CMAP XYZ files.

Currently the output formats supported are CSV, DXF, EPS (Encapsulated PostScript), GPX, HPGL for plotters, JSON, KML, Survex POS files, and SVG.

Also survexport can produce Compass .plt files, which were primarily intended for importing into Carto; the principal author of Carto has sadly died and it seems Carto is no longer actively developed, but we've left this support in place in case it is useful - the generated files can be used with Compass itself for example, though they are rather crudely structured.

POS Format

The POS format is a Survex-specific format containing a list of stations with coordinates (ordered x,y,z [East, North, Up]) and complete names. In old versions of Survex it was produced by the (now removed) `3dtopos` tool. Since Survex 1.2.19 it can be generated by `survexport` or by `aven`'s export feature.

The header line is translated to the user's language, but always starts with `(` and ends with `)`.

While not a requirement of the format, in `.pos` files created by Survex the stations are sorted by name such that numbers occur in the correct order (so `2` before `10`). Numbers with a prefix and/or suffix are sorted by the prefix as a string, then the number part as above, then by the suffix as a string, so you'd get:

```
040.sv8
040.sv8a
040.sv8b
040.sv8c
040.sv9
040.sv10
040.sv11
40_entrance_tag
40b_entrance_tag
```

DXF Format

DXF export separates Splays, Surface legs, Surface points, survey legs, and survey stations onto separate layers. Splays will export dotted, and surface legs dashed. This is not currently configurable.

OPTIONS

-s, --survey=*SURVEY*
: only load the sub-survey with this prefix

--scale=*SCALE*
: scale (`50`, `0.02`, `1:50` and `2:100` all mean 1:50)

--bearing=*BEARING*
: bearing (`90`, `90d`, `100g` all mean 90°)

--tilt=*TILT*
: tilt (`45`, `45d`, `50g`, `100%` all mean 45°)

--plan
: plan view (equivalent to `--tilt=-90`)

--elevation
: elevation view (equivalent to `--tilt=0`)

--legs
: underground survey legs

--surface-legs
: surface survey legs

--splays
: splay legs

--crosses
: station markers

--station-names
: station labels

--entrances
: entrances

--fixes
: fixed points

--exports
: exported stations

`--cross-sections`
cross-sections

`--walls`
walls

`--passages`
passages

`--origin-in-centre`
origin in centre

`--full-coordinates`
full coordinates

`--clamp-to-ground`
clamp to ground

`--defaults`
include items exported by default

`-g`, `--grid`, `--grid=`*GRID*
generate grid with spacing `GRID` metres (default `100`)

`-t`, `--text-height=`*TEXT_HEIGHT*
station labels text height (default `0.6`)

`-m`, `--marker-size=`*MARKER_SIZE*
station marker size (default `0.8`)

`--csv`
produce CSV output

`--dxf`
produce DXF output

`--eps`
produce EPS output

`--gpx`
produce GPX output

`--hpgl`
produce HPGL output

`--json`
produce JSON output

`--kml`
produce KML output

`--plt`
produce Compass PLT output for Carto

`--pos`
produce Survex POS output

`--svg`
produce SVG output

`--help`
display short help and exit

`--version`
: output version information and exit

CHAPTER

FOUR

SURVEX DATA FILES

Survey data is entered in the form of text files. You can use any text editor you like for this, so long as it has the capability of writing a plain text file in UTF-8 or ASCII encoding. The data format is very flexible; unlike some other cave surveying software, Survex does not require survey legs to be rearranged to suit the computer, and the ordering of instrument readings on each line is fully specifiable. So you can enter your data much as it appears on the survey notes, which is important in reducing the opportunities for transcription errors.

Also all the special characters are user-definable - for example, the separators can be spaces and tabs, or commas (e.g. when exporting from a spreadsheet), etc; the decimal point can changed to be a comma (as used in continental Europe), or a slash (sometimes used for clarity in written survey notes), or anything else you care to choose. This flexibility means that it should be possible to read in data from almost any sort of survey data file without much work.

Survex places no restrictions on you in terms of the ordering of survey legs. You can enter or process data in any order and Survex will read it all in before determining how it is connected. You can also use the hierarchical naming so that you do not need to worry about using the same station name twice.

The usual arrangement is to have one file which lists all the others that are included (e.g., `161.svx`). Then `cavern 161` will process all your data. To just process a section use the filename for that section, e.g. `cavern dtime` will process the dreamtime file/section of Kaninchenhöhle. To help you out, if the survey has no fixed points and you are using Survex's default unspecified coordinate system, `cavern` will pick a station and fix it at (0,0,0) (and print a info message to this effect).

It is up to you what data you put in which files. You can have one file per trip, or per area of the cave, or just one file for the whole cave if you like. On a large survey project it makes sense to group related surveys in the same file or directory.

4.1 Readings

Blank lines (i.e. lines consisting solely of `BLANK` characters) are ignored, except that in interleaved data a blank line ends the current traverse.

The last line in the file need not be terminated by an end of line character.

All fields on a line must be separated by at least one `BLANK` character. An `OMIT` character (default `-`) indicates that a field is omitted. If the field is not optional, then an error is given.

4.2 Survey Station Names

Survex has a powerful system for naming stations. It uses a hierarchy of survey names, similar to the nested folders your computer stores files in. So point 6 in the entrance survey of Kaninchenhöhle (cave number 161) can just be referred to as station 6 in the context of that particular survey, but it has the fully qualified name: 161.entrance.6

This seems a natural way to refer to station names. It also means that it is very easy to include more levels, for example if you want to plot all the caves in the area you just list them all in another file, specifying a new prefix. So to group 3 nearby caves on the Loser Plateau you would use a file like this:

```
*begin Loser
*include 161
*include 2YrGest
*include 145
*end Loser
```

The entrance series point mentioned above would now be referred to as: Loser.161.entrance.6

You do not have to use this system at all, and can just give all stations unique identifiers if you like: 1, 2, 3, 4, 5, … 1381, 1382 or AA06, AA07, P34, ZZ6, etc.

Station and survey names may contain any alphanumeric characters and additionally any characters in `NAMES` (default `_` and `-`). Alphabetic characters may be forced to upper or lower case by using the `*case` command. Station names may be any length - if you want to only treat the first few characters as significant you can get cavern to truncate the names using the `*truncate` command.

If you have survey data which uses `.` as part of the station name (for example, if you use the Toporobot convention of naming stations along a side passage from station `6` as `6.1`, `6.2`, `6.3`, etc) then there are two sensible options:

- You can change the separator to a different character which you don't want to use in station names (e.g. `:`) and set `.` as an allowed name character like so:

  ```
  *set separator :
  *set names ._-
  ```

  Note that the character(s) listed replace those previously allowed, so here we have explicitly list `_` and `-` as still allowed in station names to effectively add `.`.

  If you want to do this, you should use Survex 1.4.6 or later. 1.4.12 also fixed a bug with survey filtering when loading `.3d` files which use a separator other than `.`.

- You can use a different character instead of `.` for naming such side-passage stations, e.g. `6_1`, `6_2`, etc. This has the advantage of working with older Survex versions.

4.2.1 Anonymous Stations

Survex supports the concept of anonymous survey stations, that is survey stations without a name. Each time an anonymous station name is used it represents a different point. Currently three types of anonymous station are supported, referred to by one, two or three separator characters - with the default separator of `.`, that means `.`, `..`, and `...` are anonymous stations. Their meanings are:

Single separator (`.` by default)
: An anonymous non-wall point at the end of an implicit splay.

Double separator (`..` by default)
: An anonymous wall point at the end of an implicit splay.

Triple separator (`...` by default)
: An anonymous point with no implicit flags on the leg (intended for cases like a disto leg along a continuing passage).

You can map `-` to `..` (for compatibility with data from pocket topo) using the command:

```
*alias station - ..
```

Support for anonymous stations and for `*alias station - ..` was added in Survex 1.2.7.

Support for anonymous stations in the `cartesian` data style was added in Survex 1.4.10.

4.3 Numeric fields

Measurements start with an optional `PLUS` or `MINUS` character (+ and - by default), and have an optional decimal point (represented by a `DECIMAL` character, default: `.`) which may be embedded, leading or trailing. No spaces are allowed between the various elements.

All of these are valid examples: +47, 23, -22, +4.5, 1.3, -0.7, +12., 23., -34, +.15, .4, -.05

In formal syntax that's either:

`[<MINUS>|<PLUS>] <integer part> [ <DECIMAL> [ <decimal fraction> ] ]`

or

`[<MINUS>|<PLUS>] <DECIMAL> <decimal fraction>`

4.4 Accuracy

Accuracy assessments may be provided or defaulted for any survey leg. These determine the distribution of loop closure errors over the legs in the loop. See `*SD` for more information.

4.5 Cavern Commands

Commands in `.svx` files are introduced by an asterisk (by default - this can be changed using the `set` command).

The commands are documented below in a common format:

- Command Name
- Syntax
- Example
- Validity
- Description
- Caveats
- See Also

4.5.1 ALIAS

Syntax

`*alias station <alias> <target>`

`*alias station <alias>`

Example

```
*begin parsons_nose
*alias station - ..
1 2 12.21 073 -12
2 -  4.33 011 +02
2 -  1.64 180 +03
2 3  6.77 098 -04
*end parsons_nose
```

Description

`*alias` allows you to map a station name which appears in the survey data to a different name internally. At present, you can only create an alias of `-` to `..`, which is intended to support the pocket topo style notation of `-` being a splay to an anonymous point on the cave wall. You can also unalias `-` with `*alias station -`.

Aliases are scoped by `*begin`/`*end` blocks - when a `*end` is reached, the aliases in force at the corresponding `*begin` are restored.

`*alias` was added in Survex 1.2.7.

See Also

`*begin`, `*end`

4.5.2 BEGIN

Syntax

`*begin <survey>`

`*begin`

Example

```
*begin littlebit
1 2 10.23 106 -02
2 3  1.56 092 +10
*end littlebit
```

```
; length of leg across shaft estimated
*begin
*sd tape 2 metres
9 10 6.   031 -07
*end
```

Description

`*begin` stores the current values of the current settings such as instrument calibration, data format, and so on. These stored values are restored after the corresponding `*end`. If a survey name is given, this is used inside the `*begin`/`*end` block, and the corresponding `*end` should have the same survey name. `*begin`/`*end` blocks may be nested to indefinite depth.

See Also

`*end`, `*prefix`

4.5.3 CARTESIAN

Syntax

`*cartesian grid`

`*cartesian magnetic`

`*cartesian true`

`*cartesian grid <rotation> <units>`

`*cartesian magnetic <rotation> <units>`

`*cartesian true <rotation> <units>`

Example

```
*cartesian magnetic
```

```
*cartesian true 90 degrees
```

Description

`*cartesian` specifies which North cartesian data is aligned to, and can optionally specify an extra rotation to apply. The default is that it's aligned with True North.

Notes on the different North options:

`GRID`
: North in the current input coordinate system (as set by e.g. `*cs UTM30`). If no input or output coordinate system is set then this is the same as `TRUE` since in Survex's default unspecified coordinate system True North is the same as Grid North.

`MAGNETIC`
: Magnetic North. If using automatically calculated declinations then this will be calculated at the `*date` in effect for each cartesian data reading.

`TRUE`
: True North. If no input or output coordinate system is set then this is the same as `GRID` since in Survex's default unspecified coordinate system True North is the same as Grid North.

`*cartesian` was added in Survex 1.4.10. Prior to this cartesian data was documented as aligned with True North, but if an output coordinate system was specified it was actually aligned with this (which was not intended and doesn't really make sense since changing the output coordinate system would rotate cartesian data by the difference in grid convergence).

See Also

`*cs`, `*data cartesian`, `*date`, `*declination`

4.5.4 CALIBRATE

Syntax

`*calibrate <quantity list> <zero error>`

`*calibrate <quantity list> <zero error> <scale>`

`*calibrate <quantity list> <zero error> <units>`

`*calibrate <quantity list> <zero error> <units> <scale>`

`*calibrate default`

Example

```
*calibrate tape +0.3
```

Description

`*calibrate` is used to specify instrument calibrations, via a zero error and an optional scale factor (which defaults to 1.0 if not specified). Without an explicit calibration the zero error is 0.0 and the scale factor is 1.0.

`<quantity list>` is one or more of:

Quantity	Aliases
LENGTH	TAPE
BEARING	COMPASS
GRADIENT	CLINO
BACKLENGTH	BACKTAPE
BACKBEARING	BACKCOMPASS
BACKGRADIENT	BACKCLINO
COUNT	COUNTER
LEFT	
RIGHT	
UP	CEILING
DOWN	FLOOR
DEPTH	
EASTING	DX
NORTHING	DY
ALTITUDE	DZ
DECLINATION	

The specified calibration is applied to each quantity in the list, which is handy if you use the same instrument to measure several things, for example:

```
*calibrate left right up down +0.1
```

You need to be careful about the sign of the ZeroError. Survex follows the convention used with scientific instruments - the ZeroError is what the instrument reads when measuring a reading which should be zero. So for example, if your tape measure has the end missing, and you are using the 30cm mark to take all measurements from, then a zero distance would be measured as 30cm and you would correct this with:

```
*CALIBRATE tape +0.3
```

If you tape was too long, starting at -20cm (it does happen!) then you can correct it with:

```
*CALIBRATE tape -0.2
```

Note: ZeroError is irrelevant for Topofil counters and depth gauges since pairs of readings are subtracted.

In the first form in the synopsis above, the zero error is measured by the instrument itself (e.g. reading off the number where a truncated tape now ends) and any scale factor specified applies to it, like so (Scale defaults to 1.0):

```
Value = ( Reading - ZeroError ) * Scale
```

In the second form above (supported since Survex 1.2.21), the zero error has been measured externally (e.g. measuring how much too long your tape is with a ruler) - the units of the zero error are explicitly specified and any scale factor is not applied to it:

```
Value = ( Reading * Scale ) - ZeroError
```

With the default scale factor of 1.0 the two forms are equivalent, though they still allow you to document how the zero error has been determined.

With older Survex versions, you would specify the magnetic declination (difference between True North and Magnetic North) by using `*calibrate declination` to set an explicit value (with no scale factor allowed).

Since Survex 1.2.22, it's recommended to instead use the new `*declination` command instead - see the documentation of that command for more details.

See Also

`*declination`, `*units`

4.5.5 CASE

Syntax

`*case preserve`

`*case toupper`

`*case tolower`

Example

```
*begin bobsbit
; Bob insists on using case sensitive station names
*case preserve
1 2   10.23 106 -02
2 2a   1.56 092 +10
2 2A   3.12 034 +02
2 3    8.64 239 -01
*end bobsbit
```

Description

`*case` determines how the case of letters in survey names is handled. By default all names are forced to lower case (which gives a case insensitive match), but you can tell cavern to force to upper case, or leave the case as is (in which case `2a` and `2A` will be regarded as different).

See Also

`*truncate`

4.5.6 COPYRIGHT

Syntax

`*copyright <year> <text>` `*copyright <year1>-<year2> <text>`

Example

```
*begin littlebit
*copyright 1983 CUCC
1 2 10.23 106 -02
2 3  1.56 092 +10
*end littlebit
```

```
*copyright 1976-2024 "CUCC Expo"
```

Validity

valid at the start of a `*begin`/`*end` block.

Description

`*copyright` allows the copyright information to be recorded in a way that can be automatically collated.

The date can be specified as a single year or a range of years. Two digit years are not allowed and the end of the range can not be before the start.

The text is expected to identify the copyright holder - typically it will be the name of a person or group. Unless it is a single word you should put double quotes around it.

Prior to Survex 1.4.17 there weren't any checks of the syntax. Essentially `*copyright` used to be treated like a named comment line.

With Survex 1.4.17 and later you'll get a warning for an empty `*copyright`, for an invalid date, or if you open but fail to close double quotes around the text.

These diagnostic messages were made warnings to avoid breaking processing of existing datasets which might contain `*copyright` lines which don't conform with the defined syntax (especially as the format of the date and text fields were not documented prior to 1.4.17).

See Also
`*begin`

4.5.7 CS

Syntax
`*cs <coordinate system>`

`*cs out <coordinate system>`

Example

```
*cs UTM60S
*fix beehive 313800 5427953 20
```

```
; Output in the coordinate system used in the Totes Gebirge in Austria
*cs out custom "+proj=tmerc +lat_0=0 +lon_0=13d20 +k=1 +x_0=0 +y_0=-5200000␣
↪+ellps=bessel +towgs84=577.326,90.129,463.919,5.137,1.474,5.297,2.4232"
```

Description
`*cs` allows the coordinate systems used for fixed points and for processed survey data to be specified.

The "input" coordinate system is set with `*cs` and you can change it between fixed points if you have some fixed points in different coordinate systems to others.

The "output" coordinate system is set with `*cs out` and is what the survey data is processed in and the coordinate system used for resultant `.3d` file (which means Aven knows how to translate coordinates to allow export to formats such as GPX and KML, and to overlay terrain data and other geodata). The output coordinate system must be in metres with axis order (East, North, Up), so for example `*cs out long-lat` isn't valid because it isn't in metres, while `*cs out jtsk` isn't valid because the axes point West and South.

`*cs` was added in Survex 1.2.14, but handling of fixed points specified with latitude and longitude didn't work until 1.2.21. Also `*fix` with standard deviations specified also didn't work until 1.2.21.

The currently supported coordinate systems are:

- `EPSG:` followed by a positive integer code. EPSG codes cover most coordinate systems in use. The website https://epsg.io/ is a useful resource for finding the EPSG code you want. For example, `EPSG:4167` is NZGD2000. Supported since Survex 1.2.15.
- `CUSTOM` followed by a PROJ string (like in the example above).
- `ESRI:` followed by a positive integer code. ESRI codes are used by ArcGIS to specify coordinate systems (in a similar way to EPSG codes) and PROJ supports many of them. Supported since Survex 1.2.15.
- `EUR79Z30` for UTM zone 30, EUR79 datum. Supported since Survex 1.2.15.

- `IJTSK` for the modified version of the Czechoslovak S-JTSK system where the axes point East and North. Supported since Survex 1.2.15.
- `IJTSK03` for a variant of IJTSK. Supported since Survex 1.2.15.
- `JTSK` for the Czechoslovak S-JTSK system. Its axes point West and South, so it's not supported as an output coordinate system. Supported since Survex 1.2.16.
- `JTSK03` for a variant of JTSK. Supported since Survex 1.2.16.
- `LONG-LAT` for longitude/latitude. The WGS84 datum is assumed. NB `*fix` expects the coordinates in the order x,y,z which means longitude (i.e. E/W), then latitude (i.e. N/S), then altitude. Supported since Survex 1.2.15.
- `OSGB:` followed by a two letter code for the UK Ordnance Survey National Grid. The first letter should be 'H', 'N', 'O', 'S' or 'T'; the second any letter except 'I'. For example, `OSGB:SD`. Supported since Survex 1.2.15.
- `S-MERC` for the "Web Mercator" spherical mercator projection, used by online map sites like OpenStreetMap, Google maps, Bing maps, etc. Supported since Survex 1.2.15.
- `UTM` followed by a zone number (1-60), optionally followed by "N" or "S" specifying the hemisphere (default is North). The WGS84 datum is assumed. A potential source of confusion here is the Military Grid Reference System which divides each UTM zone into latitude bands represented by a letter suffix, so here 33S and 33N have different meanings to those in Survex - they are both parts of UTM zone 33, but both are in the Northern hemisphere (33S is around Sicily, 33N around Cameroon). To use such coordinates in Survex, replace suffixes "C" to "M" with "S", and "N" to "X" with "N".

By default, Survex works in an unspecified coordinate system (and this was the only option before `*cs` was added). However, it's useful for the coordinate system which the processed survey data is in to be specified if you want to use the processed data in ways which required knowing the coordinate system (such as exporting a list of entrances for use in a GPS). You can now do this by using `*cs out`.

It is also useful to be able to take coordinates for fixed points in whatever coordinate system you receive them in and put them directly into Survex, rather than having to convert with an external tool. For example, you may have your GPS set to show coordinates in UTM with the WGS84 datum, even though you want the processed data to be in some local coordinate system. Someone else may provide GPS coordinates in yet another coordinate system. You just need to set the appropriate coordinate system with `*cs` before each group of `*fix` commands in a particular coordinate system.

If you're going to make use of `*cs`, then a coordinate system must be specified for everything, so a coordinate system must be in effect for all `*fix` commands, and you must set the output coordinate system before any points are fixed.

Also, if `*cs` is in use, then you can't omit the coordinates in a `*fix` command, and a fixed point won't be invented if none exists.

If you use `*cs out` more than once, the second and subsequent commands are silently ignored - this makes it possible to combine two datasets with different `*cs out` settings without having to modify either of them.

Something to be aware of with `*cs` is that altitudes are currently assumed to be "height above the ellipsoid", whereas GPS units typically give you "height above sea level", or more accurately "height above a particular geoid". This is something we're looking at how best to address, but you shouldn't need to worry about it if your fixed points are in the same coordinate system as your output, or if they all use the same ellipsoid. For a more detailed discussion of this, please see: https://expo.survex.com/handbook/survey/coord.htm

See Also
`*declination auto`, `*fix`

4.5.8 DATA

Syntax

```
*data <style> <ordering>

*data

*data default

*data ignore
```

Example

```
*data normal from to compass tape clino
```

```
*data normal station ignoreall newline compass tape clino
```

Description

<style>
NORMAL|DIVING|CARTESIAN|TOPOFIL|CYLPOLAR|NOSURVEY|PASSAGE

<ordering>
ordered list of instruments - which are valid depends on the style.

In Survex 1.0.2 and later, TOPOFIL is simply a synonym for NORMAL, left in to allow older data to be processed without modification. Use the name NORMAL by preference.

Most of the styles support two variants - interleaved and non-interleaved. Non-interleaved is "one line per leg", interleaved has a line for the data shared between two legs (e.g. STATION:FROM/TO, DEPTH:FROMDEPTH/TODEPTH, COUNT:FROMCOUNT/TOCOUNT). Note that not all readings that can be shared have to be, for example here the to/from station name is shared but the depth gauge readings aren't:

```
*data diving station newline fromdepth compass tape todepth
```

In addition, interleaved data can have a DIRECTION reading, which can be F for a foresight or B for a backsight (meaning the direction of the leg is reversed).

In interleaved data, a blank line (one which contains only characters which are set as BLANK) ends the current traverse so can be used to handle branches in the survey, e.g.:

```
*data normal station newline tape compass clino

1
    9.34   087   -05
2
    ; single leg up unexplored side passage
    4.30   002    +06
3

2
    ; and back to the main package
    6.29   093    -02
4
```

In data styles which include a TAPE reading (i.e. NORMAL, DIVING, and CYLPOLAR data styles), TAPE may be replaced by FROMCOUNT/TOCOUNT (or COUNT in interleaved data) to allow processing of surveys performed with a Topofil instead of a tape.

In Survex 1.2.44 and later, you can use `*data` without any arguments to keep the currently set data style, but resetting any state. This is useful when you're entering passage tubes with branches - see the description of the `PASSAGE` style below. (This feature was originally added in 1.2.31, but was buggy until 1.2.44 - any data up to the next `*data` gets quietly ignored.)

DEFAULT
Select the default data style and ordering (`NORMAL` style, ordering: `from to tape compass clino`).

IGNORE
Ignores survey data until another `*data` command sets a different style, or until the end of the enclosing `*begin`…`*end` block. Note that commands are still processed, only survey data is ignored. This is useful if you have some survey data which has been superseded by a better survey of the same passage, but you want to keep the superseded data around, just not process it. Added in Survex 1.4.11.

NORMAL
The usual tape/compass/clino centreline survey. For non-interleaved data the allowed readings are: `FROM TO TAPE COMPASS CLINO BACKTAPE BACKCOMPASS BACKCLINO`; for interleaved data the allowed readings are: `STATION DIRECTION TAPE COMPASS CLINO BACKTAPE BACKCOMPASS BACKCLINO`.

`BACKTAPE` was added in Survex 1.2.25.

The `CLINO`/`BACKCLINO` reading is not required - if it is omitted in the `*data` command then the vertical standard deviation is taken to be proportional to the tape measurement for all reading. Alternatively, if the reading is included in the `*data` command then individual clino readings can be given as `OMIT` (default `-`) and will be treated in this way, which allows for data where only some clino readings are missing.

Examples of style `NORMAL`:

```
*data normal from to compass clino tape
1 2 172 -03 12.61
2 3 202  -   8.59 ; clino not recorded
```

```
*data normal station newline direction tape compass clino
1
  F 12.61 172 -03
2
```

```
*data normal from to compass clino fromcount tocount
1 2 172 -03 11532 11873
```

```
*data normal station count newline direction compass clino
1 11532
  F 172 -03
2 11873
```

DIVING
An underwater survey where the vertical information is from a diver's depth gauge. This style can also be also used for an above-water survey where the altitude is measured with an altimeter. `DEPTH` is defined as the altitude (Z) so increases upwards by default. So for a diver's depth gauge, you'll need to use `*CALIBRATE` with a negative scale factor (e.g. `*calibrate depth 0 -1`).

For non-interleaved data the allowed readings are: `FROM TO TAPE COMPASS CLINO BACKTAPE BACKCOMPASS BACKCLINO FROMDEPTH TODEPTH DEPTHCHANGE` (the vertical can be given as readings at each station, (`FROMDEPTH`/`TODEPTH`) or as a change along the leg (`DEPTHCHANGE`)).

`BACKTAPE` was added in Survex 1.2.25.

For interleaved data the allowed readings are: `STATION DIRECTION TAPE COMPASS BACKTAPE BACKCOMPASS DEPTH DEPTHCHANGE`. The vertical change can be given as a reading at the station (`DEPTH`) or as a change along the leg (`DEPTHCHANGE`):

```
*data diving from to tape compass fromdepth todepth
1 2 14.7 250 -20.7 -22.4
```

```
*data diving station depth newline tape compass
1 -20.7
 14.7 250
2 -22.4
```

```
*data diving from to tape compass depthchange
1 2 14.7 250 -1.7
```

Survex 1.2.20 and later allow an optional `CLINO` and/or `BACKCLINO` reading in `DIVING` style. At present these extra readings are checked for syntactic validity, but are otherwise ignored. The intention is that a future version will check them against the other readings to flag up likely blunders, and average with the slope data from the depth gauge and tape reading.

CARTESIAN

Cartesian data style allows you to specify the (x,y,z) changes between stations. It's useful for digitising surveys where the original survey data has been lost and all that's available is a drawn up version.

```
*data cartesian from to northing easting altitude
1 2 16.1 20.4 8.7
```

```
*data cartesian station newline northing easting altitude
1
  16.1 20.4 8.7
2
```

By default, the North used is True North, but you can specify to use Magnetic or Grid North (in the input coordinate system) with the `*cartesian` command, and also specify an additional rotation to apply (since Survex 1.4.10).

In Survex < 1.4.10, if `*cs` was used then cartesian data were incorrectly interpreted as relative to Grid North in the output coordinate system (if `*cs` is not used then Grid North in the default unspecified coordinate system is the same as True North).

CYLPOLAR

A `CYLPOLAR` style survey is very similar to a diving survey, except that the tape is always measured horizontally rather than along the slope of the leg.

```
*data cylpolar from to tape compass fromdepth todepth
1 2 9.45 311 -13.3 -19.0
```

```
*data cylpolar station depth newline tape compass
1 -13.3
 9.45 311
2 -19.0
```

```
*data cylpolar from to tape compass depthchange
1 2 9.45 311 -5.7
```

NOSURVEY

A NOSURVEY survey doesn't have any measurements - it merely indicates that there is line of sight between the pairs of stations.

```
*data nosurvey from to
1 7
5 7
9 11
```

```
*data nosurvey station
1
7
5

*data
9
11
```

PASSAGE

This survey style defines a 3D "tube" modelling a passage in the cave. The tube joins the survey stations listed in the order listed. It's permitted to go between survey stations which aren't directly linked by the centre-line survey. This can be useful - sometimes the centreline will step sideways or up/down to allow a better sight for the next leg and you can ignore the extra station. You can also define tubes along unsurveyed passages, akin to "nosurvey" legs in the centreline data.

This means that you need to split off side passages into separate tubes, and hence separate sections of passage data, starting with a new `*data` command with no arguments.

Simple example of how to use this data style (note the use of ignoreall to allow a free-form text description to be given):

```
*data passage station left right up down ignoreall
1  0.1 2.3 8.0 1.4  Sticking out point on left wall
2  0.0 1.9 9.0 0.5  Point on left wall
3  1.0 0.7 9.0 0.8  Highest point of boulder
```

Each `*data passage` data block describes a single continuous tube - to break a tube or to enter a side passage you need to have a second block. With Survex 1.2.30 and older, you had to repeat the entire `*data passage` line to start a new tube, but in Survex 1.2.31 and later, you can just use `*data` without any arguments.

For example here the main passage is 1-2-3 and a side passage is 2-4:

```
*data passage station left right up down ignoreall
1  0.1 2.3 8.0 1.4  Sticking out point on left wall
2  0.0 1.9 9.0 0.5  Point on left wall opposite side passage
3  1.0 0.7 9.0 0.8  Highest point of boulder

; If you need to be compatible with Survex 1.2.30 or earlier
; you need to repeat the full "*data" command here instead.
*data
2  0.3 0.2 9.0 0.5
4  0.0 0.5 6.5 1.5  Fossil on left wall
```

IGNORE skips a field (it may be used any number of times), and IGNOREALL may be used last to ignore the rest of the data line.

`LENGTH` is a synonym for `TAPE`; `BEARING` for `COMPASS`; `GRADIENT` for `CLINO`; `COUNT` for `COUNTER`.

The units of each quantity may be set with the `*units` command.

See Also
`*units`

4.5.9 DATE

Syntax

`*date <date type(s)> <ISO date>`

`*date <date type(s)> <ISO date1> <ISO date2>`

`*date <ISO date>`

`*date <ISO date1> <ISO date2>`

`*date <legacy date>`

`*date <legacy date1>-<legacy date2>`

Example

```
*date explored 1987-06-20 1987-06-28
*date surveyed 1987-07-11
```

```
*date explored surveyed 2024-11-29
```

```
*date 1976-08
```

```
*date 1968
```

```
*date 1987.07.27
```

```
*date 1985.08.12-1985.08.13
```

```
*date 2000.10
```

Validity
valid at the start of a `*begin`/`*end` block.

Description
`*date` specifies the date that the survey was done. A range of dates can be specified (e.g. for "surveyed" date, this is useful for overnight or multi-day surveying trips).

Date components must be in the order year then month then day. Later components can be omitted to specify the date to the granularity of a month or year (which is sometimes useful for older survey data where the exact date of a survey may no longer be known). Such partial dates are treated as a date range for that whole month or year; if used in a date range, the appropriate extreme of the year or month is used as that end of the range - e.g. `2001 2004-06` is from the start of 2001 to the end of June 2004.

Survex 1.4.13 added support for date types `explored` and `surveyed` (`*date` without a type is assumed to be specifying the `surveyed` date only).

Survex 1.4.13 also added support for the ISO date format, where the separator between components is `-`. In older versions the separator between components had to be `.` (with `-` used between dates in a range). The older date format is still accepted, but we strongly recommend using ISO format dates in new data because it's a standardised date format.

We recommend avoiding two digit years because of the inherent ambiguity, but they are accepted (with a warning) and assumed to be 19xx.

Currently dates before 1900 and after 2078 result in a warning and are ignored.

A date which is in the future in the local timezone will also be warned about to help catch date errors. This warning could be incorrectly triggered if you surveyed some data and promptly sent it to somebody in a timezone behind yours - e.g. American Samoa (UTC-11) is always a day behind Samoa (UTC+13). This could be avoided by making the threshold for the warning less tight (e.g. add a day or interpret today's date as "anywhere on Earth"), but then we could fail to warn when locally processed data you've just entered with tomorrow's date which seems worse than an incorrect warning in an unusual case which will go away after less than a day.

The `explored` date is parsed but not currently stored anywhere. A future version will write it to the `.3d` file and make it available to aven, etc.

See Also

`*begin`, `*instrument`, `*team`

4.5.10 DECLINATION

Syntax

`*declination auto <x> <y> <z>`

`*declination <declination> <units>`

Description

The `*declination` command is the modern way to specify magnetic declinations in Survex. It was added in Survex 1.2.21 but buggy so not really usable until 1.2.22.

Magnetic declination is the difference between Magnetic North and True North. A compass measures an angle relative to Magnetic North - adding the magnetic declination gives bearings relative to True North. The magnetic declination varies with location and also with time as the Earth's magnetic field moves.

Generally it's best to specify a suitable output coordinate system, and use `*declination auto` so Survex corrects for magnetic declination for you. Survex 1.2.27 and later will also automatically correct for grid convergence (the difference between Grid North and True North).

If you don't specify an output coordinate system, but fix one or more points then Survex works implicitly in the coordinate system your fixed points were specified in. This mode of operation is provided for compatibility with datasets from before support for explicit coordinate systems was added to Survex - it's much better to specify the output coordinate system as above. But if you have a survey of a cave which isn't connected to any known fixed points then you'll need to handle it this way, either fixing an entrance to some arbitrary coordinates (probably (0,0,0)) or letting Survex pick a station as the origin. If the survey was all done in a short enough period of time that the magnetic declination won't have changed significantly, you can just ignore it and Grid North in the implicit coordinate system will be Magnetic North at the time of the survey. If you want to correct for magnetic declination, you can't use `*declination auto` because the IGRF model needs the real world coordinates, but you can specify literal declination values for each survey using `*declination <declination> <units>`. Then Grid North in the implicit coordinate system is True North.

Prior to 1.2.21, `*calibrate declination` was used instead. If you use a mixture of `*calibrate declination` and `*declination`, they interact in the natural way - whichever was set most recently is used for each compass reading (taking into account survey scope). We don't generally recommend mixing the two, but it's useful to understand how they interact if you want to combine datasets using the old and new commands, or if you have a large existing dataset and want to migrate it without having to change everything at once.

Note that the value specified uses the conventional sign for magnetic declination, unlike the old `*calibrate declination` which needed a value with the opposite sign (because `*calibrate` specifies a zero error), so take care when updating old data, or if you're used to the semantics of `*calibrate declination`.

When `*declination auto` is used cavern uses the IGRF (International Geomagnetic Reference Field) model to compute magnetic declinations. A revised version of the IGRF model is usually issued every 5 years, and calculates values using a model based on observations for years before it is issued, and on predictions for 5 years after it is issued.

Here's a table of the first Survex version to support each version of the IGRF model:

IGRF Version	Survex version	Survex version release date
14	1.4.13	2024-12-01
13	1.2.43	2020-02-28
12	1.2.21	2015-07-28

The IGRF model takes a date and a location as inputs. Survex uses the specified date of the survey (if the survey's date is a range, the centre of that range is used), and uses the "x y z" coordinates specified in the `*declination auto` command as the location in the current input coordinate system (as set by `*cs`). Most users can just specify a single representative location somewhere in the area of the cave. If you're not sure what to use, pick some coordinates roughly in the middle of the bounding box of the cave - it doesn't need to be a fixed point or a known refindable location, though it can be if you prefer.

You might wonder why Survex needs a representative location instead of calculating the magnetic declination and grid convergence for the actual position of each survey station. The reason is that we need to adjust the compass bearings before we can solve the network to find survey station locations. Both magnetic declination and grid convergence don't generally vary significantly over the area of a typical cave system - if you are mapping a very large cave system, or caves over a wide area, or are working close to a magnetic pole or where the output coordinate system is rather distorted, then you can specify `*declination auto` several times with different representative locations for different areas of the cave system - the one currently in effect is used for each survey leg.

For each `*declination auto` command cavern will (since Survex 1.4.2) report the range of calculated declination values and the dates at which the ends of the range were obtained, and also the grid convergence (which doesn't vary with time). This appears in the log - if you processed the data with aven you can view this by using "File->View Log". It looks like this:

```
1623.svx:20: info: Declination: -0.4° @ 1977-07-02 / 3.8° @ 2018-07-21, grid␣
↪convergence: -0.9°
 *declination auto 36670.37 83317.43 1903.97
```

It also (since Survex 1.4.16) reports the approximate true range of convergence values - by comparing this with the reported convergence values for the representative locations you can see if you might need to add more. This looks like:

```
Approximate full range of grid convergence: -0.9° at 1626.5.1 to -0.8° at 1624.133.
↪1-6
```

This is "approximate" because it's only computed for the North-most, South-most, East-most and West-most stations and it's possible the actual minimum or maximum not at one of these. It's unlikely to be much outside the reported range though.

We don't (currently) attempt to report a similar range for declination values (it's harder to do because declination also varies by date).

See Also
`*calibrate`, `*cs`

4.5.11 DEFAULT

Syntax

`*default calibrate`

`*default data`

`*default units`

Description

`*default` restores defaults for given settings. This command is deprecated - you should instead use: `*calibrate default`, `*data default`, `*units default`.

See Also

`*calibrate`, `*data`, `*units`

4.5.12 END

Syntax

`*end <survey>`

`*end`

Validity

valid for closing a block started by `*begin` in the same file.

Description

Closes a block started by `*begin`.

See Also

`*begin`

4.5.13 ENTRANCE

Syntax

`*entrance <station>`

Example

```
*entrance P163
```

Description

`*entrance` marks a station as an entrance. This information is used by aven to allow entrances to be highlighted.

4.5.14 EQUATE

Syntax

`*equate <station> <station>...`

Example

```
*equate chosspot.1 triassic.27
```

Description

`*equate` specifies that the station names in the list refer to the same physical survey station. An error is given if there is only one station listed.

See Also

`*infer equates`

4.5.15 EXPORT

Syntax

`*export <station>...`

Example

```
*export 1 6 17
```

Validity

valid at the start of a `*begin`/`*end` block.

Description

`*export` marks the stations named as referable to from the enclosing survey. To be able to refer to a station from a survey several levels above, it must be exported from each enclosing survey.

See Also

`*begin`, `*infer exports`

4.5.16 FIX

Syntax

`*fix <station> [reference] <x> <y> <z>`

`*fix <station> [reference] <x> <y> <z> <std err>`

`*fix <station> [reference] <x> <y> <z> <horizontal std err> <vertical std err>`

`*fix <station> [reference] <x> <y> <z> <x std err> <y std err> <z std err>`

`*fix <station> [reference] <x> <y> <z> <x std err> <y std err> <z std err> <cov(x, y)> <cov(y,z)> <cov(z,x)>`

`*fix <station>`

Example

```
*fix entrance.0 32768 86723 1760
```

```
*fix KT114_96 reference 36670.37 83317.43 1903.97
```

Description

`*fix` fixes the position of <station> at the given coordinates. If you haven't specified the coordinate system with `*cs`, you can omit the position and it will default to (0,0,0) which provides an easy way to specify a point to arbitrarily fix rather than rely on `cavern` picking one (which has the downsides of the choice potentially changing when more survey data is added, and of triggering an "info" message).

The standard errors default to zero (fix station exactly). `cavern` will give an error if you attempt to fix the same survey station twice at different coordinates, or a warning if you fix it twice with matching coordinates.

You can also specify just one standard error (in which case it is assumed equal in X, Y, and Z) or two (in which case the first is taken as the standard error in X and Y, and the second as the standard error in Z).

If you have covariances for the fix, you can also specify these - the order is cov(x,y) cov(y,z) cov(z,x).

If you've specified a coordinate system (see `*cs`) then that determines the meaning of X, Y and Z (if you want to specify the units for altitude, note that using a PROJ string containing `+vunits` allows this - e.g. `+vunits=us-ft` for US survey feet). If you don't specify a coordinate system, then the coordinates must be in metres. The standard deviations must always be in metres (and the covariances in metres squared).

You can fix as many stations as you like - just use a `*fix` command for each one. Cavern will check that all stations are connected to at least one fixed point so that co-ordinates can be calculated for all stations. If there is unconnected survey data then you'll get a warning (since Survex 1.4.10; in earlier versions this was an error) and only the connected data is processed.

By default cavern will warn about stations which have been `*fix`-ed but are not used otherwise, as this might be due to a typo in the station name. Uses in survey data and (since 1.4.9) `*entrance` count for these purposes. This warning is unhelpful if you want to include a standard file of benchmarks, some of which won't be used. In this sort of situation, specify `reference` after the station name in the `*fix` command to suppress this warning for a particular station. It's OK to use `reference` on a station which is used.

Since Survex 1.4.10 it's an error to specify `reference` without coordinates (e.g. `*fix a reference`) as this usage doesn't really make sense.

Note

X is Easting, Y is Northing, and Z is altitude. This convention was chosen since on a map, the horizontal (X) axis is usually East, and the vertical axis (Y) North. The choice of altitude (rather than depth) for Z is taken from surface maps, and makes for less confusion when dealing with cave systems with more than one entrance. It also gives a right-handed set of axes.

4.5.17 FLAGS

Syntax

`*flags <flags>`

Example

```
*flags duplicate not surface
```

Description

`*flags` updates the currently set flags. Flags not mentioned retain their previous state. Valid flags are `duplicate`, `splay`, and `surface`, and a flag may be preceded with `not` to turn it off.

Survey legs marked `surface` are hidden from plots by default, and not included in cave survey length calculations.

Survey legs marked as `duplicate` or `splay` are also not included in cave survey length calculations; legs marked `splay` are ignored by the extend program. `duplicate` is intended for the case when if you have two different surveys along the same section of passage (for example to tie two surveys into a permanent survey station); `splay` is intended for cases such as radial legs in a large chamber, or to walls and other features with a disto-x or similar device.

See Also

`*begin`

4.5.18 INCLUDE

Syntax

`*include <filename>`

Example

```
*include mission
```

```
*include "the pits"
```

Description

`*include` processes <`filename`> as if it were inserted at this place in the current file - i.e. the current settings are carried into the included file and any alterations to settings in the included file will be carried back again. There's one exception to this for historical reasons, which is that the survey prefix is restored upon return to the original file. Since `*begin` and `*end` nesting cannot cross files, this can only make a difference if you use the deprecated `*prefix` command.

If the filename contains spaces, it must be enclosed in double quotes.

An included file which does not have a complete path is resolved relative to the directory which the parent file is in (just as relative HTML links do).

The included file can be any filetype which cavern can process, so you can `*include compassdata.mak`, `*include compassdata.dat`, `*include wallsdata.wpj` or `*include wallsdata.srv` to allow processing mixed-format datasets.

If the filename as specified is not found, cavern will try adding a `.svx` extension, and will also try translating `\` to `/`.

4.5.19 INFER

Syntax

`*infer plumbs on`

`*infer plumbs off`

`*infer equates on`

`*infer equates off`

`*infer exports on`

`*infer exports off`

Description

`*infer plumbs on` tells cavern to interpret gradients of ±90 degrees as UP/DOWN (so it will not apply the clino correction to them). This is useful when you have data which uses this convention for plumbed legs.

`*infer equates on` tells cavern to interpret a leg with a tape reading of zero as a `*equate` which this prevents tape corrections being applied to them. This is useful when you have data which uses this convention for equating stations.

`*infer exports on` is necessary when you have a dataset which is partly annotated with `*export`. It tells cavern not to complain about missing `*export` commands in the parts of the dataset it is enabled for. Also stations which were used to join surveys are marked as exported in the 3d file.

4.5.20 INSTRUMENT

Syntax

`*instrument <instrument> <identifier>`

Example

```
*instrument compass "CUCC 2"
*instrument clino "CUCC 2"
*instrument tape "CUCC Fisco Ranger open reel"
```

Validity
valid at the start of a `*begin`/`*end` block.

Description
`*instrument` specifies the particular instruments used to perform a survey.

See Also
`*begin`, `*date`, `*team`

4.5.21 PREFIX

Syntax
`*prefix <survey>`

Example

```
*prefix flapjack
```

Description
`*prefix` sets the current survey.

Caveats
`*prefix` is deprecated - you should use `*begin` and `*end` instead.

See Also
`*begin`, `*end`

4.5.22 REF

Syntax
`*ref <string>`

Example

```
*ref "survey folder 2007#12"
```

Validity
valid at the start of a `*begin`/`*end` block.

Description
`*ref` allows you to specify a reference. If the reference contains spaces, you must enclose it in double quotes. Survex doesn't try to interpret the reference in any way, so it's up to you how you use it - for example it could specify where the original survey notes can be found.

`*ref` was added in Survex 1.2.23.

See Also
`*begin`, `*date`, `*instrument`, `*team`

4.5.23 REQUIRE

Syntax
`*require <version>`

Example

```
   *require 1.2.14 ; for *cs

::
```

(continues on next page)

(continued from previous page)

```
   *require 1.4 ; equivalent to 1.4.0

::

   *require 1 ; equivalent to 1.0.0
```

Description

`*require` checks that the version of cavern in use is at least `<version>` and stops with an error if not.

If your dataset requires a feature introduced in a particular version, you can add a `*require` command and users will know what version they need to upgrade to, rather than getting an error message and having to guess what the real problem is. We suggest noting the reason for the requirement in a comment after the `*require` command, like in the first example above.

4.5.24 SD

Syntax

`*sd <quantity list> <standard deviation> <units>`

Example

```
*sd tape 0.15 metres
```

```
*sd compass backcompass clino backclino 0.5 degrees
```

Description

`*sd` specifies the standard deviation of a measurement.

`<quantity>` is one of (each group gives alternative names for the same quantity):

- TAPE, LENGTH
- BACKTAPE, BACKLENGTH (added in Survex 1.2.25)
- COMPASS, BEARING
- BACKCOMPASS, BACKBEARING
- CLINO, GRADIENT
- BACKCLINO, BACKGRADIENT
- COUNTER, COUNT
- DEPTH
- DECLINATION
- DX, EASTING
- DY, NORTHING
- DZ, ALTITUDE
- LEFT
- RIGHT
- UP, CEILING
- DOWN, FLOOR

- LEVEL
- PLUMB
- POSITION

<`standard deviation`> is a positive real number, e.g. `0.05`.

<`units`> specifies the units, which must be appropriate for the quantity or quantities in the list (e.g. `metres` for distance, `degrees` for an angle). See `*units` below for full list of valid units.

To utilise this command fully you need to understand what a *standard deviation* is. It gives a value to the 'spread' of the errors in a measurement. Assuming that these are normally distributed we can say that 95.44% of the actual lengths will fall within two standard deviations of the measured length, e.g. a tape SD of 0.25 metres means that the actual length of a tape measurement is within ±0.5 metres of the recorded value 95.44% of the time. So if the measurement is 7.34m then the actual length is very likely to be between 6.84m and 7.84m. This example corresponds to BCRA grade 3. Note that this is just one interpretation of the BCRA standard, taking the permitted error values as 2SD 95.44% confidence limits. If you want to take the readings as being some other limit (e.g. 1SD = 68.26%) then you will need to change the BCRA3 and BCRA5 files accordingly. This issue is explored in more detail in various surveying articles.

See Also
`*units`

4.5.25 SET

Syntax
`*set <item> <character list>`

Example

```
*set blank x09x20
*set decimal ,
```

This example sets the decimal separator to be a comma. Note that here we need to eliminate comma from being a blank before setting it as a decimal - otherwise the comma in `*set decimal ,` is parsed as a blank, and cavern sets decimal to not have any characters representing it.

Here's an example of how to allow additional characters in station names:

```
*set names ?+_
```

After this ?, + and _ will be allowed in names (in addition to characters A-Z, a-z and 0-9 which are always allowed). `*set` replaces the previous setting so while `_` and `-` are both allowed by default, `-` no longer is after this command because it isn't in the list.

Description
`*set` sets the specified <`item`> to the character or characters given in <`character list`>.

The characters specified in <`character list`> may not be alphanumeric since these characters are used for station names and for readings.

In <`character list`>, `x` followed by two hex digits means the character with that hex value, e.g. `x20` is a space.

The complete list of items that can be set, the defaults (in brackets), and the meaning of the item, is:

BLANK (`x09x20,`)
Separates fields

COMMENT (`;`)
The rest of the current line is a comment

DECIMAL (`.`)
Decimal point character

EOL (`x0Ax0D`)
End of line character

KEYWORD (*)
Introduces keywords

MINUS (`-`)
Indicates negative number

NAMES (`_-`)
Non-alphanumeric chars permitted in station names (letters and numbers are always permitted).

OMIT (`-`)
Contents of field omitted (e.g. in plumbed legs)

PLUS (+)
Indicates positive number

ROOT (\)
Prefix in force at start of current file (use of `ROOT` is deprecated)

SEPARATOR (`.`)
Level separator in prefix hierarchy

4.5.26 SOLVE

Syntax
`*solve`

Example

```
*include 1997data
*solve
*include 1998data
```

Description
`*solve` distributes misclosures around any loops in the survey (in the same way that happens implicitly after reading all the data), then fixes the positions of all existing stations.

This command is intended for situations where you have some new surveys adding extensions to an already drawn-up survey which you wish to avoid completely redrawing. You can read in the old data, use `*solve` to fix it, and then read in the new data. Then old stations will be in the same positions as they are in the existing drawn up survey, even if new loops have been formed by the extensions.

4.5.27 TEAM

Syntax
`*team <person> [<role>...]`

Example

```
*team "Nick Proctor" compass clino tape
*team "Anthony Day" notes pictures tape
*team Wookey assistant
; Role not recorded
*team "Olly Betts"
```

Validity

valid at the start of a `*begin`/`*end` block.

Description

`*team` specifies the people involved in a survey and optionally what role or roles they filled during that trip. Unless the person is only identified by one name you need to put double quotes around their name.

The syntax of `*team` commands has been defined for a very long time, but prior to Survex 1.4.17 there weren't any checks of the syntax. Essentially `*team` used to be treated like a named comment line.

With Survex 1.4.17 and later you'll get a warning for an empty `*team` or if you open but fail to close double quotes around the person's name.

Roles are now checked against an allowed list (which is the same list that Therion uses, with the addition of `explorer` which Therion handles via a separate `explo-team` command). You'll get a warning if a role is not recognised.

These diagnostic messages were made warnings to avoid breaking processing of existing datasets which might contain `*team` lines which don't conform with the defined syntax, or with this newly adopted list of roles.

`<role>` should be one of the following (grouped entries are just alternative names for the same thing). The intended meanings are noted to encourage consistent usage:

Role	Alias	Intended meaning
tape	length	Measured leg lengths
compass	bearing	Measured bearings
clino	gradient	Measured vertical angles
backtape	backlength	Like `tape` but for backsights
backcompass	backbearing	Like `compass` but for backsights
backclino	backgradient	Like `clino` but for backsights
instruments	insts	All instruments: both compass and clino; use for all-in-one instruments such as Disto-X.
counter	count	Topofil length measurements
depth		Measured differences in height between stations, e.g. underwater with a diver's depth gauge, or above water with a manometer
station		Added markers at stations
position		Recorded absolute positions of stations (e.g. fixed surface stations with a GPS)
notes	notebook	Recorded instrument readings
pictures	pics	Drew sketches
assistant	dog	General helper (e.g. held the end of the tape on stations)
altitude	dz	Recorded the altitudes of stations (e.g. with an altimeter)
dimensions		Measured all passage dimensions
left		Measured `left` passage dimension
right		Measured `right` passage dimension
up	ceiling	Measured `up` passage dimension
down	floor	Measured `down` passage dimension
explorer		Explored the area of cave being surveyed

See Also
`*begin`, `*date`, `*instrument`

4.5.28 TITLE

Syntax
`*title <title>`

Example

```
*title Dreamtime
```

```
*title "Mission Impossible"
```

Description
`*title` allows you to set a descriptive title for a survey. If the title contains spaces, you need to enclose it in

double quotes (“”). If there is no `*title` command, the title defaults to the survey name given in the `*begin` command.

4.5.29 TRUNCATE

Syntax

`*truncate <length>`

`*truncate off`

Description

Station names may be of any length in Survex, but some other (mostly older) cave surveying software only regards the first few characters of a name as significant (e.g. “entran” and “entrance” might be treated as the same). To facilitate using data imported from such a package, Survex allows you to truncate names to whatever length you want (but by default truncation is off).

Figures for the number of characters which are significant in various software packages: Compass currently has a limit of 12, CMAP has a limit of 6, Smaps 4 had a limit of 8, Surveyor87/8 used 8. Survex itself used 8 per prefix level up to version 0.41, and 12 per prefix level up to 0.73 (more recent versions removed this rather archaic restriction).

See Also

`*case`

4.5.30 UNITS

Syntax

`*units <quantity list> [<factor>] <unit>`

`*units default`

Example

```
*units tape metres
```

```
*units compass backcompass clino backclino grads
```

```
*units dx dy dz 1000 metres ; data given as kilometres
```

```
*units left right up down feet
```

Description

`*units` changes the current units of all the quantities listed to [<factor>] <unit>. Note that quantities can be expressed either as the instrument (e.g. `COMPASS`) or the measurement (e.g. `BEARING`).

<quantity> is one of the following (grouped entries are just alternative names for the same thing):

- `TAPE/LENGTH`
- `BACKTAPE/BACKLENGTH` (added in Survex 1.2.25)
- `COMPASS/BEARING`
- `BACKCOMPASS/BACKBEARING`
- `CLINO/GRADIENT`
- `BACKCLINO/BACKGRADIENT`
- `COUNTER/COUNT`

- DEPTH
- DECLINATION
- DX/EASTING
- DY/NORTHING
- DZ/ALTITUDE
- LEFT
- RIGHT
- UP/CEILING
- DOWN/FLOOR

<factor> allows you to easy specify situations such as measuring distance with a diving line knotted every 10cm (`*units distance 0.1 metres`). If <factor> is omitted it defaults to `1.0`. If specified, it must be non-zero.

Valid units for listed quantities are:

TAPE/LENGTH, BACKTAPE/BACKLENGTH, COUNTER/COUNT, DEPTH, DX/EASTING, DY/NORTHING, DZ/ALTITUDE in YARDS|FEET|METRIC|METRES|METERS (default: METRES)

CLINO/GRADIENT, BACKCLINO/BACKGRADIENT in DEGS|DEGREES|GRADS|MINUTES|PERCENT|PERCENTAGE (default: DEGREES)

COMPASS/BEARING, BACKCOMPASS/BACKBEARING, DECLINATION in DEGS|DEGREES|GRADS|MINUTES|QUADS|QUADRANTS (default: DEGREES)

FEET and YARDS use the international definition of a foot (exactly 0.3048m). If you want to use a different definition, you can specify it explicitly - for example, for the US survey foot:

```
*units tape 0.3048006096012192 meters ; US survey foot
```

or for the Indian survey foot:

```
*units tape 0.3047996 meters ; Indian survey foot
```

QUADRANTS are a style of bearing used predominantly in land survey, and occasionally in survey with handheld instruments. All bearings are N or S, a numeric from 0 to 90, followed by E or W. For example S34E to refer to 146 degrees, or 34 degrees in the SE quadrant. In this format, exact cardinal directions may be simply alphabetic. E.g. N is equivalent to N0E and E is equivalent to N90E. This unit was added in Survex 1.2.44.

GRADS are an angle unit where 400 grads = 360 degrees. They're also known as "neugrads" (or sometimes "gons").

Survex has long supported MILS as an alias for GRADS. However, this seems to be a bogus definition of a "mil" which is unique to Survex (except that Therion has since copied it) - there are several different definitions of a "mil" but they vary from 6000 to 6400 in a full circle, not 400. Because of this we deprecated MILS in Survex 1.2.38 - you can still process data which uses them but you'll now get a warning, and we recommend you update your data.

For example, if your data uses

```
*units compass mils
```

then you need to determine what the intended units actually are. If there are 400 in a full circle, then instead use this (which will work with older Survex versions too):

```
*units compass grads
```

If the units are actually mils, you can specify that in terms of degrees. For example, there are 6400 NATO mils in a full circle, so one NATO mil is 360/6400 degrees, and 360/6400=0.05625 so you can use this (which also works with older Survex versions):

```
; Compass readings are NATO mils (6400 = 360 degrees)
*units compass 0.05625 degrees
```

See Also
`*calibrate`

CHAPTER

FIVE

.SVX COOKBOOK

Here is some example Survex data (a very small cave numbered 1623/163):

```
2 1 26.60 222  17.5
2 3 10.85 014   7
2 4  7.89 254 -11
4 5  2.98  - DOWN
5 6  9.29 271 -28.5
```

You can vary the data ordering. The default is:

from-station to-station tape compass clino

This data demonstrates a number of useful features of Survex:

Legs can be measured either way round, which allows the use of techniques like “leap-frogging” (which is where legs alternate forwards and backwards).

Also notice that there is a spur in the survey (2 to 3). You do not need to specify this specially.

Survex places few restrictions on station naming (see “Survey Station Names” in the previous section), so you can number the stations as they were in the original survey notes. Although not apparent from this example, there is no requirement for each leg to connect to an existing station. Survex can accept data in any order, and will check for connectedness once all the data has been read in.

Each survey is also likely to have other information associated with it, such as instrument calibrations, etc. This has been omitted from this example to keep things simple.

Most caves will take more than just one survey trip to map. Commonly the numbering in each survey will begin at 1, so we need to be able to tell apart stations with the same number in different surveys.

To accomplish this, Survex has a very flexible system of hierarchical prefixes. All you need do is give each survey a unique name or number, and enter the data like so:

```
*begin 163
*export 1
2 1 26.60 222  17.5
2 3 10.85 014   7
2 4  7.89 254 -11
4 5  2.98  - DOWN
5 6  9.29 271 -28.5
*end 163
```

Survex will name the stations by attaching the current prefix. In this case, the stations will be named 163.1, 163.2, etc.

We have a convention with the CUCC Austria data that the entrance survey station of a cave is named P<cave number>, P163 in this case. We can accomplish this like so:

```
*equate P163 163.1
*entrance P163
*begin 163
*export 1
2 1 26.60 222  17.5
2 3 10.85 014   7
2 4  7.89 254 -11
4 5  2.98  - DOWN
5 6  9.29 271 -28.5
*end 163
```

5.1 Join surveys together

Once you have more than one survey you need to specify how they link together. To do this use `*export` to make the stations to be joined accessible in the enclosing survey, then `*equate` in the enclosing survey to join them together. For example:

```
; CUCC convention is that p<number> is the entrance station
; for cave <number>.
*entrance p157
*equate p157 157.0

*begin 157

*export 157.0 ; tag bolt

*begin entpitch
*team "Olly Betts" Notes
*team "Jenny Black" Insts
*date 2012.08.05
*data normal from to tape compass clino
*export 0 5

1    0       3.226   202     -05
1    2       1.505   080     -13
1    3       2.605   030     +25
3    4       8.53    335     +80
4    5       7.499   060     +24

*end entpitch

*equate entpitch.5 pt2.1

*begin pt2
*team "Olly Betts" Notes
*team "Jenny Black" Insts
*date 2012.08.29
*export 1

1    2       5.361   054     -54
2    3       4.271   190     +00
2    4       4.634   138     +04
```

(continues on next page)

(continued from previous page)

```

*end pt2

*end 157
```

5.2 Surface survey data

Say you have 2 underground surveys and 2 surface ones with a single fixed reference point. You want to mark the surface surveys so that their length isn't included in length statistics, and so that Aven knows to display them differently. To do this you mark surface data with the "surface" flag - this is set with `*flags surface` like so:

```
; fixed reference point
*fix fix_a 12345 56789 1234

; surface data (enclosed in *begin ... *end to stop the *flags command
; from "leaking" out)
*begin
*flags surface
*include surface1
*include surface2
*end

; underground data
*include cave1
*include cave2
```

You might also have a single survey which starts on the surface and heads into a cave. This can be easily handled too - here's an example which goes in one entrance, through the cave, and out of another entrance:

```
*begin BtoC
*title "161b to 161c"
*date 1990.08.06 ; trip 1990-161c-3 in 1990 logbook

*begin
*flags surface
02    01      3.09   249    -08.5
02    03      4.13   252.5  -26
*end

04    03      6.00   020    +37
04    05      3.07   329    -31
06    05      2.67   203    -40.5
06    07      2.20   014    +04
07    08      2.98   032    +04
08    09      2.73   063.5  +21
09    10     12.35   059    +15

*begin
*flags surface
11    10      4.20   221.5  -11.5
11    12      5.05   215    +03.5
```

(continues on next page)

(continued from previous page)

```
11    13      6.14   205    +12.5
13    14     15.40   221    -14
*end

*end BtoC
```

Note that to avoid needless complication, Survex regards each leg as being either “surface” or “not surface” - if a leg spans the boundary you’ll have to call it one or the other. It’s good surveying practice to deliberately put a station at the surface/underground interface (typically the highest closed contour or drip line) so this generally isn’t an onerous restriction.

5.3 Reading order

The `*data` command is used to specify the order in which the readings are given.

5.4 Plumbs

Plumbed legs can be specified by using `UP` or `DOWN` in place of the clino reading and a dash (or a different specified `OMIT` character) in place of the compass reading. This distinguishes them from legs measured with a compass and clino. Here’s an example:

```
1 2 21.54 - UP
3 2 7.36 017 +17
3 4 1.62 091 +08
5 4 10.38 - DOWN
```

`U/D` or `+V/-V` may be used instead of `UP/DOWN`; the check is not case sensitive.

If you prefer to use `-90` and `+90` as “magic” clino readings which are instead treated as plumbs and don’t get clino corrections applied, etc, then see `*infer plumbs on`.

5.5 Legs across static water

Legs surveyed across the surface of a static body of water where no clino reading is taken (since the surface of the water can be assumed to be flat) can be indicated by using `LEVEL` in place of a clino reading. This prevents the clino correction being applied. (Note that unlike with plumbed readings, there isn’t a way to treat a clino reading of `0` as meaning `LEVEL` because `0` is a completely reasonable actual clino reading.)

Here’s an example:

```
1 2 11.37 190 -12
3 2  7.36 017 LEVEL
3 4  1.62 091 LEVEL
```

5.6 Specify a BCRA grade

The `*sd` command can be used to specify the standard deviations of the various measurements (tape, compass, clino, etc). Examples files are supplied which define BCRA Grade 3 and BCRA Grade 5 using a number of `*sd` commands. You can use these by simply including them at the relevant point, as follows:

```
*begin somewhere
; This survey is only grade 3
*include grade3
2 1 26.60 222  17.5
2 3 10.85 014   7
; etc
*end somewhere
```

The default values for the standard deviations are those for BCRA grade 5. Note that it is good practice to keep the `*include grade3` within `*begin` and `*end` commands otherwise it will apply to following survey data, which may not be what you intended.

5.7 Override accuracy for a leg

For example, suppose the tape on the plumbed leg in this survey is suspected of being less accurate than the rest of the survey because the length was obtained by measuring the length of the rope used to rig the pitch. We can set a higher sd for this one measurement and use a `*begin/*end` block to make sure this setting only applies to the one leg:

```
2 1 26.60 222  17.5
2 3 10.85 014   7
2 4  7.89 254 -11
*begin
; tape measurement was taken from the rope length
*sd tape 0.5 metres
4 5  34.50 - DOWN
*end
5 6  9.29 271 -28.5
```

5.8 Repeated Readings

If your survey data contains multiple versions of each leg (for example, pockettopo produces such data), then provided these are adjacent to one another, Survex 1.2.17 and later will automatically average these and treat them as a single leg.

5.9 Radiolocation Data

This is done by using the `*sd` command to specify the appropriate errors for the radiolocation "survey leg" so that the loop closure algorithm knows how to distribute errors if it forms part of a loop.

The best approach for a radiolocation where the underground station is vertically below the surface station is to represent it as a plumbed leg, giving suitable SDs for the length and plumb angle. The horizontal positioning of this is generally quite accurate, but the vertical positioning may be much less well known. E.g. we have a radiolocation of about 50m depth ±20m and horizontal accuracy of ±8m. Over 50m the ±8m is equivalent to an angle of 9 degrees, so that is the expected plumb error. 20m is the expected error in the length. To get the equivalent SD we assume that 99.74% of readings will be within 3 standard deviations of the error value. Thus we divide the expected errors by 3 to get the SD we should specify:

```
*begin
*sd length 6.67 metres
*sd plumb 3 degrees
```

(continues on next page)

(continued from previous page)

```
surface underground 50 - down
*end
```

We wrap the radiolocation leg in a `*begin`/`*end` block to make sure that the special *sd settings only apply to this one leg.

For more information on the expected errors from radiolocations see Compass Points Issue 10, available online at https://www.chaos.org.uk/survex/cp/CP10/CPoint10.htm

5.10 Diving Data

Surveys made underwater using a diver's depth gauge can be processed - use the `*data` command with the `diving` style to specify that the data is of this type.

5.11 Theodolite data

Theodolite data with turned angles is not yet explicitly catered for, so for now you will need to convert it into equivalent legs in another style - `normal` or `cylpolar` styles are likely to be the best choices.

If there is no vertical info in your theodolite data then you should use the `cylpolar` style and use `*sd` command to specify very low accuracy (high SD) in the depth so that the points will move in the vertical plane as required if the end points are fixed or the survey is part of a loop.

5.12 Change special characters

See the `*set` command documentation for details, and some examples.

CHAPTER

SIX

GENERAL: HOW DO I?

6.1 Create a new survey

You create a text file containing the relevant survey data, using a text editor, and save it with a suitable name with a `.svx` extension. The easiest way is to look at some of the example data and use that as a template. Nearly all surveys will need a bit of basic info as well as the survey data itself: e.g. the date (`*date`), comments about where, what cave, a name for the survey (using `*begin` and `*end`), instrument error corrections, etc. Here is a typical survey file:

All the lines starting with `;` are comments, which are ignored by Survex. You can also see the use of `DOWN` for plumbs, and `*calibrate tape` for dealing with a tape length error (in this case the end of the tape had fallen off so measurements were made from the 20cm point).

```
*equate chaos.1 triassic.pt3.8
*equate chaos.2 triassic.pt3.9

*begin chaos
*title "Bottomless Pit of Eternal Chaos to Redemption pitch"
*date 1996.07.11
*team "Nick Proctor" compass clino tape
*team "Anthony Day" notes pictures tape
*instrument compass "CUCC 2"
*instrument clino "CUCC 2"
;Calibration: Cairn-Rock 071 072 071,  -22 -22 -22
;       Rock-Cairn 252 251 252,  +21 +21 +21
;Calibration at 161d entrance from cairn near entrance to
;prominent rock edge lower down. This is different from
;calibration used for thighs survey of 5 July 1996

*export 1 2

;Tape is 20cm too short
*calibrate tape +0.2

1 2 9.48 208 +08
2 3 9.30 179 -23
3 4 2.17 057 +09
5 4 10.13 263 +78
5 6 2.10 171 -73
7 6 7.93 291 +75
*begin
*calibrate tape 0
8 7 35.64 262 +86 ;true length measured for this leg
```

(continues on next page)

(continued from previous page)

```
*end
8 9 24.90 - DOWN
10 9 8.61 031 -43
10 11 2.53 008 -34
11 12 2.70 286 -20
13 12 5.36 135 +23
14 13 1.52 119 -12
15 14 2.00 036 +13
16 15 2.10 103 +12
17 16 1.40 068 -07
17 18 1.53 285 -42
19 18 5.20 057 -36
19 20 2.41 161 -67
20 21 27.47 - DOWN
21 22 9.30 192 -29
*end chaos
```

6.2 Organise my surveys

This is actually a large subject. There are many ways you can organise your data using Survex. Take a look at the example dataset for some ideas of ways to go about it.

6.2.1 Fixed Points (Control Points)

The `*fix` command is used to specify fixed points (also know as control points). See the description of this command in the "Cavern Commands" section of this manual.

6.2.2 More than one survey per trip

Suppose you have two separate bits of surveying which were done on the same trip. So the calibration details, etc. are the same for both, but you want to give a different survey name to the two sections. This is easily achieved like so:

```
*begin
*calibrate compass 1.0
*calibrate clino 0.5
*begin altroute
; first survey
*end altroute
*begin faraway
; second survey
*end faraway
*end
```

6.3 Add surface topography

Survex 1.2.18 added support for loading terrain data and rendering it as a transparent surface. Currently the main documentation for this is maintained as a wiki page as this allows us to update it between releases.

This supports loading data in the HGT format that NASA offers SRTM data in. The SRTM data provides terrain data on a 1 arc-second grid (approximately 30m) for most of the world.

6.4 Overlay a grid

Aven is able to display a grid, but this functionality isn't currently available in printouts. You can achieve a similar effect for now by creating a `.svx` file where the survey legs form a grid.

If you want to do this, we suggest fixing points at the end of each grid line and using the `NOSURVEY` data style to add effectively elastic legs between these fixed points. This is simpler to generate than generating fake tape/compass/clino legs and is very fast for cavern to process. Some tips for doing this

Here's a small example of a 500mx500m grid with lines 100m apart:

```
*fix 0W 000 000 0
*fix 1W 000 100 0
*fix 2W 000 200 0
*fix 3W 000 300 0
*fix 4W 000 400 0
*fix 5W 000 500 0

*fix 0E 500 000 0
*fix 1E 500 100 0
*fix 2E 500 200 0
*fix 3E 500 300 0
*fix 4E 500 400 0
*fix 5E 500 500 0

*fix 0S 000 000 0
*fix 1S 100 000 0
*fix 2S 200 000 0
*fix 3S 300 000 0
*fix 4S 400 000 0
*fix 5S 500 000 0

*fix 0N 000 500 0
*fix 1N 100 500 0
*fix 2N 200 500 0
*fix 3N 300 500 0
*fix 4N 400 500 0
*fix 5N 500 500 0

*data nosurvey from to

0W 0E
1W 1E
2W 2E
3W 3E
4W 4E
5W 5E

0S 0N
1S 1N
2S 2N
3S 3N
4S 4N
5S 5N
```

6.5 Import data from other programs

Survex supports a number of features to help with importing existing data.

Unprocessed survey data in Compass or Walls format can be processed directly, and mixed datasets are support to aid combining data from different projects using different software. This also can help if you want to migrate data into Survex from another format as you can leave the existing data in its original format and just use Survex native format for new data, or if you prefer to convert everything to Survex native format it can be done in phases.

Processed survey data in Compass or CMAP formats can be viewed in `aven` and used with any Survex command line tool which takes processed survey data.

For data in formats without explicit support, you may be able to read the data by renaming the file to have a `.svx` extension and adding a few Survex commands at the start of the file to set things up so Survex can read the data which follows. For example, you can specify the ordering of items on a line using `*data` (see Survex Keywords above), and you can specify the characters used to mean different things using `*set` (see Survex Keywords above).

The `ignore` and `ignoreall` items in the `*data` command are often particularly useful, e.g. if you have a dataset with LRUD info or comments on the ends of lines.

6.6 See errors and warnings that have gone off the screen

When you run Survex it will process the specified survey data files in order, reporting any warnings and errors. If there are no errors, the output files are written and various statistics about the survey are displayed. If there are a lot of warnings or errors, they can scroll off the screen and it's not always possible to scroll back to read them.

The easiest way to see all the text is to use `cavern --log` to redirect output to a `.log` file, which you can then inspect with a text editor.

6.7 Create an Extended Elevation

You can create a simple extended elevation from `aven`, using the `File->Extended Elevation...` menu option. This takes the currently loaded survey file and "flattens" it.

Behind the scenes this runs the `extend` command-line program. This program offers more powerful features, such as being able to control which way legs are folded and where loops are broken, but currently these features are only accessible from the command line (the intention is to allow them to be used from `aven` in the future).

CHAPTER

SEVEN

LARRY FISH'S COMPASS

Survex can read Compass survey data - it supports survey data files and project files (`.DAT` and `.MAK files`), closed data files (`.CLP`), and processed survey data (`.PLT` and `.PLF` files). Survex 1.4.6 made significant improvements to this support so we recommend using this version or newer if you're working with Compass data.

7.1 Compass .MAK support

A Compass `.MAK` file defines a survey project to process, and specifies one or more `.DAT` files to process, along with coordinate systems and fixed points. You can process a `.MAK` file with cavern or aven as if it were a `.svx` file.

Survex understands most MAK file features. Known current limitations and assumptions:

- Survex handles the UTM zone and datum provided the combination can be expressed as an EPSG code (lack of any EPSG codes for a datum suggests it's obsolete; lack of a code for a particular datum+zone combination suggests the zone is outside of the defined area of use of the datum). Example Compass files we've seen use "North American 1927" outside of where it's defined for use, presumably because some users fail to change the datum from Compass' default. To enable reading such files we return a PROJ4 string of the form "+proj=utm …" for "North American 1927" and "North American 1983" for UTM zones which don't have an EPSG code. Please let us know if support for additional cases which aren't currently supported would be useful to you.
- The `@` command which specifies a base location to calculate magnetic declinations at is handled, provided the datum and UTM zone are supported (see previous bullet point). The UTM convergence angle specified as part of this command is ignored as Survex knows how to calculate it.
- Link stations are ignored. These have two uses in Compass. They were a way to allow processing large surveys on computers from last century which had limited memory. Survex can easily handle even the largest surveys on current hardware and ignoring them is not a problem in this case.

  The other use is they provide a way to process surveys together which use the same station names for different stations. In this case we recommend writing a `.svx` file to replace the MAK file which wraps the `*include` of each DAT file in `*begin survey1`/`*end survey1`, etc so that stations have unique qualified names. Then the link stations can be implemented using e.g. `*equate survey1.XX1 survey2.XX1`. This example from the Compass documentation:

  ```
  #FILE1.DAT;               /no links
  #FILE2.DAT,A22,A16;
  #FILE3.DAT,A16,B14;
  ```

  would look like this:

  ```
  *begin file1
  *include FILE1.DAT
  *end file1
  ```

(continues on next page)

(continued from previous page)

```
*begin file2
*include FILE2.DAT
*end file2
*equate file1.A22 file2.A22
*begin file3
*include FILE3.DAT
*end file3
*equate file1.A16 file3.A16
*equate file2.B14 file3.B14
```

Note that the `.svx` version is able to more precisely represent what's actually required here - in the MAK version "you must carry `A16` into `FILE2` even though `FILE2` doesn't need it for its own processing". If you want the exact analog of the MAK version you can change the `A16` equate to:

```
*equate file1.A16 file2.A16 file3.A16
```

- The following commands (and any other unknown commands) are currently ignored: `%` (Convergence angle (file-level)), `*` (Convergence angle (non file-level)), `!` (Project parameters)

7.2 Compass .DAT support

A Compass `.DAT` file contains unprocessed survey data. You can process a `.DAT` file with cavern or aven as if it were a `.svx` file.

You can even use `*include compassfile.dat` or `*include compassproject.mak` in a `.svx` file and it'll work, which allows combining separate cave survey projects maintained in Survex and Compass.

One point to note when doing so (this tripped us up!) is that station names in Compass are case sensitive and so Survex reads `.DAT` files with the equivalent of `*case preserve`. The default in `.svx` files is `*case lower` so this won't work

```
*fix CE1 0 0 0
*include datfilewhichusesCE1.dat
```

because `CE1` in the `*fix` is actually interpreted as `ce1`. The solution is to turn on preserving of case while you fix the point like so:

```
*begin
*case preserve
*fix CE1 0 0 0
*end
*include datfilewhichusesCE1.dat
```

If you want to be able to refer to the fixed point from Survex data too then you can add in a `*equate` to a lower-case name to achieve that:

```
*begin
*case preserve
*fix CE1 0 0 0
*equate CE1 ce1
*end
*include datfilewhichusesCE1.dat
*include svxfilewhichusesce1.svx
```

Or if you're just wanting to link a Compass survey to a Survex one, you can use a `*equate` with `*case preserve` on:

```
*begin
*case preserve
*equate CE1 ce1
*end
*include datfilewhichusesCE1.dat
*include svxfilewhichusesce1.svx
```

Survex understands most DAT file features. Known current limitations and assumptions:

- The cave name, survey short name, survey comment and survey team information are currently ignored (because this information isn't currently saved in the `.3d` file even for `.svx` files).
- Survey date January 1st 1901 is treated as "no date specified", since this is the date Compass stores in this situation, and it seems very unlikely to occur in real data.
- Passage dimensions are currently ignored.
- Shot flag `C` in Compass causes flagged legs to not be subject to loop closure. Survex currently sets the SDs of such legs to 1mm, so flagged legs can still move slightly during loop closure. (Since 1.4.16; earlier versions ignored this flag entirely.)
- Shot flag `L` is mapped to Survex's "duplicate" leg flag.
- Shot flag `P` is mapped to Survex's "surface" leg flag. The Compass documentation describes shot flag `P` as "Exclude this shot from plotting", but the suggested use for it is for surface data, and shots flagged `P` "[do] not support passage modeling". Even if it's actually being used for a different purpose, Survex programs don't show surface legs by default so the end effect is at least to not plot as intended.
- Shot flag `S` is mapped to Survex's "splay" leg flag.
- Surveys which indicate a depth gauge was used for azimuth readings are marked as `STYLE_DIVING` in the `.3d` file.
- Compass seems to quietly ignore a shot with the same "from" and "to" station. This seems likely to be a mistake in the data so Survex 1.4.12 and later warn about this in a Compass DAT file (in native Survex data this is treated as an error, which is how older Survex versions treat it in Compass DAT files).

7.3 Compass .CLP support

A Compass .CLP file contains raw survey data after adjusting for loop closure. The actual format is otherwise identical to a Compass `.DAT` file, and Survex 1.4.6 and later support processing a .CLP file with cavern or aven as if it were a `.svx` file (the extra support is to recognise the `.CLP` extension, and to not apply the instrument corrections a second time).

You can even use `*include compassfile.clp` in a `.svx` file and it'll work, which allows combining separate cave survey projects maintained in Survex and Compass.

Usually it is preferable to process the survey data without loop closure adjustments (i.e. `.DAT`) so that when new data is added errors get distributed appropriately across old and new data, but it might be useful to use the `.CLP` file if you want to keep existing stations at the same adjusted positions, for example to be able to draw extensions on an existing drawn-up survey which was processed with Compass.

Another possible reason to use the data from a `.CLP` file is if that's all you have because the original `.DAT` file has been lost!

7.4 Compass .PLF/.PLT support

A Compass `.PLT` file contains processed survey data. The extension `.PLF` is also used for "special feature files" which have essentially the same format.

Survex supports both reading and writing these files, each of which are documented in separate sections below.

7.4.1 Reading Compass .PLF/.PLT

You can load these files with `aven` as if they were .3d files, and similarly for other Survex tools which expect a .3d file such as `survexport`, `extend`, `diffpos`, `3dtopos` and `dump3d`. (This support is actually provided by Survex's img library, so other programs which use this library should also be able to read Compass `.PLT` files without much extra work.)

Survex understands most PLT file features. Known current limitations and assumptions:

- Survey date January 1st 1901 is treated as "no date specified", since this is the date Compass stores in this situation, and it seems very unlikely to occur in real data.
- Passage dimensions are translated to passage tubes, but Survex may interpret them differently from Compass.
- Shot flag `C` is ignored. It only seems to be useful in unprocessed survey data.
- Shot flag `L` is mapped to Survex's "duplicate" leg flag.
- Shot flag `P` and plot command `d` are mapped to Survex's "surface" leg flag. The Compass documentation describes shot flag `P` as "Exclude this shot from plotting", but the suggested use for it is for surface data, and shots flagged `P` "[do] not support passage modeling". Even if it's actually being used for a different purpose, Survex programs don't show surface legs by default so the end effect is at least to not plot as intended. Stations are flagged as surface and/or underground based on whether they are at the ends of legs flagged surface or non-surface (a station at the boundary can be flagged as both).
- Shot flag `S` is mapped to Survex's "splay" leg flag. A station at the far end of a shot flagged `S` gets the "station on wall" flag set since the Compass PLT format specification says: "The shot is a "splay" shot, which is a shot from a station to the wall to define the passage shape."
- Stations with "distance from entrance" of zero are flagged as entrances.
- Stations which are present in multiple surveys are flagged as exported (like when `*infer exports on` is in effect in `.svx` files).
- Stations listed as fixed points are flagged as fixed points.
- If a PLT file only uses one datum and UTM zone combination and it is supported (the same combinations are supported as for MAK files) then they are converted to the appropriate EPSG code or PROJ4 string and this is reported as the coordinate system. Please let us know if support for additional cases which aren't currently supported would be useful to you. Files with multiple datums could be handled too, but we'd need to convert coordinates to a common coordinate system in the img library, which would need it to depend on PROJ. Please let us know if support for mixed datums would be useful to you.

7.4.2 Exporting Compass .PLT

Survex can also create PLT files via `aven`'s File->Export feature, and also from the command line via `survexport --plt`.

This export was originally added to allow importing data from Survex into Carto. The principal author of Carto has sadly died and it seems Carto is no longer actively developed, but we've left this support in place in case it is useful - the generated files can be used with Compass itself for example, though they are currently rather crudely structured. Here are some notes on this support:

- The whole Survex survey tree is exported as a single survey.

- Compass station names can't contain spaces, so any spaces (and also ASCII control characters) are in station names are replaced by `%` follow by two lowercase hex digits giving the byte value (like the escaping used in URLs). `%` itself is also escaped as `%25`.
- The full Survex station name include survey prefixes is used - no attempt is currently made to shorten station names to fit within the 12 character limit documented for the Compass PLT format. If you export a single survey the names should be short enough, but exporting the whole of a complex survey project will likely give names longer than 12 characters.
- Anonymous stations are given a name `%:` followed by a number starting from one and incrementing for each anonymous station (Compass doesn't allow empty station names, and these invented names can't collide with actual station names). Since Survex 1.4.10 (1.4.6 implemented support for exporting anonymous stations to PLT, but with names which typically exceeded the documented 12 character limit of the format).
- Passage data is not included in the export (each exported leg has dummy LRUD readings of all `-9` which is needed to avoid a bug in some versions of Compass which don't cope with legs without LRUD).
- Survex's "surface" leg flag is mapped to Compass shot flag `P`. The Compass documentation describes shot flag `P` as "Exclude this shot from plotting", but the suggested use for it is for surface data, and shots flagged `P` "[do] not support passage modeling". Since Survex 1.4.10.
- Survex's "splay" leg flag is mapped to Compass shot flag `S`. Since Survex 1.4.10.
- Survex's "duplicate" leg flag is mapped to Compass shot flag `L`. Since Survex 1.4.10.
- The Datum and UTM zone information is not currently set in exported PLT files.

CHAPTER

EIGHT

DAVID MCKENZIE'S WALLS

Survex 1.4.9 and later can read Walls unprocessed survey data (.SRV and .WPJ files). Walls is no longer being developed, so the focus of support for Walls formats is primarily to help people with existing Walls data to migrate.

We've mostly implemented this support based on Walls documentation, but unfortunately the documentation of the SRV format is sometimes incomplete or incorrect, while the WPJ format seems to be largely undocumented. Sadly David is no longer around to ask, but we can at least test actual behaviour of `Walls32.exe` on example data.

As of 1.4.10, some large Walls datasets can be successfully processed (e.g. Mammoth Cave, the Thailand dataset from https://cave-registry.org.uk/, and Big Bat Cave). Behaviour is not identical and station positions after loop closure will inevitably be different, but large or apparently systematic errors are worth reporting. An easy way to compare is to export a Shapefile from `Walls32.exe` and overlay it in `aven`. The way to export is a bit hidden - after processing select the *Segments* tab, make sure the whole project is selected, and click the *Details / Rpts...* button which is towards the upper right. Click the *Shapefile...* button in the new dialog box, and select what you want to output (e.g. *Vectors*). Due to limitations in the Shapefile format each *Shape Type* selected here exports a separate Shapefile. To overlay a Shapefile in `aven` use *File->Overlay Geodata...*.

See below for a list of known limitations. We've mostly prioritised what to implement based on testing with real-world datasets so commonly used features are likely to be handled while more obscure features may not be.

- Survex reports warnings in some suspect situations which Walls quietly accepts. In general this seems helpful and they do highlight what look like genuine problems in existing datasets, but if there are particular instances which are noisy and not useful, let us know.

  Some of these warnings use the same wording as errors - most of these probably ought to be an error except Walls quietly accepts them so we don't want to fail processing because of them.

  If you want a way to suppress the "unused fixed point" warning, using the station in a `#NOTE` or `#FLAG` command counts as a "use" so you can suppress these with e.g. `#NOTE ABC123 /unused` for each such fixed point.

- If an isolated LRUD (LRUD not on a leg, e.g. specified for the start or end station of a traverse) is missing its closing delimiter, Walls will parse the line as a data leg where the "to" station starts with * or <, which will usually succeed without errors or warnings. A real-world example is:

  ```
  P25     *8 5 16 3.58
  ```

  Survex parses this like Walls does, but issues a warning:

  ```
  walls.srv:1:9: warning: Parsing as "to" station but may be isolated LRUD with␣
  ↪missing closing delimiter
   P25     *8 5 15 3.58
           ^~
  ```

  This warning will also be triggered if you have a "to" station which actually starts with * or <, if the station name was not previously seen. This condition provides a simple way to suppress the warning - just add a dummy `#NOTE` command before the line of data like so:

```
#note *8 ; Suppress Survex warning that this looks like broken LRUD
P25     *8 5 16 3.58
```

- Walls allows hanging surveys, apparently without any complaint, and as a result large Walls datasets are likely to have hanging surveys. A hanging survey used to be an error in Survex but since 1.4.10 a hanging survey is warned about and then ignored.
- `#FIX` - currently Survex does not support horizontal-only or vertical only fixes. These are currently given an SD of 1000m in that direction instead, but that is not the same as completely decoupling the fix in that direction.
- Variance overrides on survey legs are mostly supported, with the following limitations:
  - An SD of 0 is currently treated as 1mm (approximately 0.04 inches).
  - Floating a leg both horizontally and vertically (with `?`) replaces it with a "nosurvey" leg, which is effectively the same provided both ends of the leg are attached to fixed points.
  - Floating a leg either horizontally or vertically (with `?`) uses an SD of 1000m in that direction instead of actually decoupling the connection.
  - Floating the traverse containing a leg (with `*`) currently just floats that leg (so it's the same as `?`).
- Walls `#SEGMENT` is apparently in practice commonly set to a set of Compass "shot flags", so if a `#SEGMENT` value consists only of letters from the set `CLPSX` with an optional leading `/` or `\\` then we map it to Survex flags like so (since Survex 1.4.10):
  - `C` in Compass causes legs it is set on to not be affected by loop closure, but setting Compass flags in the Walls `#SEGMENT` is just a user-level convention so Walls won't do anything in response to `C`, so Survex ignores it too.
  - `L` is mapped to Survex's "duplicate" flag
  - `P` is mapped to Survex's "surface" flag
  - `S` is mapped to Survex's "splay" flag
  - `X` in Compass completely excludes the leg from processing, but setting Compass flags in the Walls `#SEGMENT` is just a user-level convention so Walls won't do anything in response to `X`. It seems in practice that `X` is sometimes used in Walls datasets to flag data to just exclude from the surveyed length, so we treat it as an alias for `L` and map it to Survex's "duplicate" flag.

  Other values of `#SEGMENT` are ignored.
- Walls `FLAG` values seem to be arbitrary text strings. We try to infer appropriate Survex station flags by checking for certain key words in that text (currently we map words `ENTRANCE` and `FIX` to the corresponding Survex station flags) and otherwise ignore `FLAG` values.
- `#NOTE` is parsed and the station name is marked as "used" (which suppresses the unused fixed point warning) but the note text is currently ignored.
- We don't currently support all the datum names which Walls does because we haven't managed to find an EPSG code for any UTM zones in some of these datums. This probably means they're not actually in current use.
- We currently assume all two digit years are 19xx (Walls documents it 'also accepts "some date formats common in the U.S. (`mm/dd/yy`, `mm-dd-yyyy`, etc.)' but doesn't say how it interprets `yy`.
- The documentation specifies that the `SAVE` and `RESTORE` options should be processed before other options. Currently Survex just processes all options in the order specified, which makes no difference to any real-world data we've seen. We need to test with Walls32.exe to determine exactly how this works (and if `RESET` is also special).
- LRUD data is currently ignored.
- The `TAPE=` option is currently quietly skipped, and tape measurements are assumed to be station to station.

- In TYPEAB= and TYPEVB=, the threshold is ignored, as is the X meaning to only use foresights (but still check backsights). Survex uses a threshold based on the specified instrument SDs, and averages foresights and backsights.
- UV=, UVH= and UVV= are all quietly skipped.
- The GRID= option currently gives an "Unknown command" warning, and is skipped. If your Walls data specifies a UTM zone then Survex will automatically correct for grid convergence.
- The INCH= option currently gives an "Unknown command" warning (unless the argument is zero, since Survex 1.4.10), and is skipped.
- Walls seems to allow \\ in place of / in some places (e.g. #FLAG). We aim to support this too, but it doesn't seem to be documented so may not currently be supported in the correct places.
- The inheritance of settings in WPJ files is probably not correctly implemented currently.
- The Walls documentation mentions a NOTE= option, but doesn't document what it does, and testing with Walls32.exe it doesn't seem to actually be supported!
- The two UPS zones for the polar regions (specified as UTM zone values of -61 and 61 in Walls) are supported with datum WGS84, but we do not have any real data to test this support with.
- Walls gives an error if an unprefixed station name is more than 8 characters long but Survex does not enforce this restriction.
- Walls documents *The total length of the three prefix components combined, including any embedded colon separators, is 127 characters* but Survex does not enforce any limit.
- In the option UNITS= the documentation says *CASE = Upper / Lower / Mixed* but it seems actually any string is allowed and if it starts with a letter other than U or L then it's treated as Mixed. Since Survex 1.4.10.
- Walls explicitly documents that *Unprefixed names [. . .] must not contain any colons, semicolons, commas, pound signs (#), or embedded tabs or spaces.* but it actually allows # in station names (though it can't be used as the first character of the from station name as that will be interpreted as a command. Since Survex 1.4.10.
- Walls ignores junk after the numeric argument in TYPEAB=, TYPEVB=, UV=, UVH=, and UVV=. Survex warns and skips the junk. Since Survex 1.4.10.
- Walls allows the clino reading to be completely omitted with ORDER=DAV and ORDER=ADV on a "wall shot" (leg to or from an anonymous station). Supported since Survex 1.4.10.
- If a station is used with an explicit Walls prefix (e.g. PEP:A123) then it will will be flagged as "exported" in the .3d file. This is currently applied even if the explicit prefix is empty (e.g. :A123). Since Survex 1.4.10.
- Walls allows a station with an explicit prefix to have an empty name, e.g. PEP:. The Walls documentation doesn't mention this, though it also doesn't explicitly say the name can't be empty. This quirk seems unlikely to be intentionally used and Survex doesn't allow an empty station name, so we issue a warning and use the name empty name (which has a space in, so can't collide with a real Walls station name which can't contain a space) - so PEP: in Walls becomes PEP.empty name in Survex. Since Survex 1.4.10.

If you find some Walls data which Survex doesn't handle or handles incorrectly, and it is not already noted above, please let us know. If you can provide some data demonstrating the problem, that's really helpful. It's also useful to know if there are things listed above that are problematic to help prioritise efforts.

CHAPTER

NINE

BOB THRUN'S CMAP

Survex can read CMAP processed survey data, commonly known as CMAP XYZ files. CMAP no longer seems to be used, but we've kept the support in place so it's there if anyone finds old data and wants to view it.

Support was added long ago (Survex 1.0.4) but you really want to use Survex 1.4.9 or later due to a feet to metres conversion bug in all versions before this, which scaled all returned coordinates from CMAP XYZ files by a factor of about 10.76.

CMAP XYZ files contain a timestamp. CMAP was originally written for computers where the clock was just set to localtime so it seems likely this timestamp is in localtime, but it does not specify a timezone. Survex assumes it's in UTC, which is at least fairly central in the possibilities, but may mean timestamps are off by up to about half a day. The timestamps in example files all use two digit years. It's not documented how CMAP handles years 2000 and later, so years < 50 get 2000 added to them, years 50 to 199 inclusive get 1900 added to them, and years >= 200 are used as is (so year 0 is 2000, year 49 is 2049, year 50 is 1950, year 99 is 1999, year 101 is 2001, and year 125 is 2025).

CMAP XYZ files don't seem to contain any station flags. There's a single character "type" which acts as a leg flag of sorts, but it seems the meaning is up to the user so we don't try to interpret it. We assume all the data is underground (so all stations get the "underground" flag).

There are two variants of CMAP XYZ files. CMAP 16 and later default to producing the "shot" variant (with extension `.sht`), which is well supported by Survex. CMAP 16.1 was released in 1995 so you're probably much more likely to encounter `.sht` files.

Older CMAP versions produced the "station" variant (with extension `.adj` for adjusted data and `.una` for unadjusted data). Survex only supports reading stations from this format - the survey legs linking them are currently ignored. This wasn't implemented originally because there seemed to be a mismatch between the documentation of the format of these files and the accompanying example files which we never managed to resolve (the `1st` column is documented as *"Station number where the name of the current station first appeared. If this is not a closure station, the number will be the current station number."* but in the sample files it seems to just be the current station number, even when there are loops).

Implementing reading based on what seems to be a buggy specification really needs a broader collection of sample files. Given this format seems to be obsolete that seems unlikely to happen now, but if you have a collection of old CMAP files you are keen to be able to read with Survex then please get in touch.

CHAPTER

TEN

USING THE IMG LIBRARY

The code Survex uses to read and write `.3d` files is provided as the img library which allows it to be reused by other programs which are written in C, C++ or another language which is able to call C functions.

Using this code means you can take advantage of any revisions to the 3d file format by simply rebuilding your software with the updated `img.c` and `img.h`, whereas if you write your own code to parse 3d files then you'll have to update it for each revision to the 3d file format.

Using the img library also allows you to read old revisions of the Survex .3d format and also other processed survey data formats: currently it supports reading Survex `.pos` files, and processed survey data from Larry Fish's Compass and from Bob Thrun's CMAP).

It also allows reading a subset of the data in a file, restricted by survey prefix.

If not able to use the img library, parsing text output from `dump3d` provides many of the same benefits. If you take this approach, you may find the `--legs` option more convenient as it gives you a `LEG` line for each leg rather than `MOVE` for the first station in a traverse then `DRAW` for each subsequent station.

10.1 Using img in your own code

The expected way to use img is that you copy `src/img.c` and `src/img.h` into your source tree, and periodically update them (they usually evolve fairly slowly).

It's not currently packaged as a separate library for Debian or anywhere else I'm aware of (there's a very low number of applications which link to it and the effort to do that seems more usefully directed; we also don't guarantee ABI stability, but it should be upwardly API compatible).

The current img code makes decisions about the availability of standard C library headers, functions and types based on the C/C++ standard version the compiler claims to support (via the `__STDC_VERSION__` and `__cplusplus` macros). In general these tests should be sensible but a bit conservative (e.g. `snprintf()` was widely supported before it was standardised by C99) but currently the highest standards tested for are C99 and C++11 and modern compilers likely default to at least these. If you want more control you can probe in your build system and define various macros named with a `HAVE_` prefix - e.g. `HAVE_SNPRINTF` to indicate that `snprintf` is available.

10.2 API

The API is documented by comments in `src/img.h`.

See `src/imgtest.c` for an example program using img as you would from outside of Survex. Also `src/dump3d.c` shows how to access all the different types of data returned (that's using img as it is in the Survex code so has different `#include` lines, opens the file in a more complex way, and has Survex-specific translation from img error codes to Survex message numbers - none of these are relevant for using img outside of Survex).

If anything is unclear, please ask on the mailing list and we can clarify.

CHAPTER

ELEVEN

MAILING LIST

The best way to contact the authors and other Survex users is the Survex mailing list - for details visit: https://survex.com/maillist.html

We'd be delighted to hear how you get on with Survex and welcome comments and suggestions for improvements.

And we'd love you to contribute your skills to help make Survex even better. Point out areas of the documentation which could be made clearer, or sections which are missing entirely. Download test releases, try them out, and let us know if you find problems or have suggestions for improvements. If there's no translation to your language, you could provide one. Or if you're a developer, *"Say it with code"*. There's plenty to do, so feel free to join in.

CHAPTER TWELVE

FUTURE DEVELOPMENTS

Now that Survex has reached version 1.0, we are continuing progress towards version 2, in a series of steps, evolving out of Survex 1.0. The GUI framework is being based on aven, with the printer drivers and other utility programs being pulled in and integrated into the menus.

Aven is built on wxWidgets, which means that it can easily support Linux, other Unix-like platforms, Microsoft Windows, and macOS.

More information on our plans is on the web site.