Tellervo

A guide for users and developers

By Peter W. Brewer

For Tellervo versions 1.4.0 (server)
and 1.4.0 (desktop)

TELLERVO

A guide for users and developers

By Peter W. Brewer

Compiled: November 22, 2022

Compiled: November 22, 2022

Contents

I User Guide   3

    15.1   Getting help 81
    15.2   Support for future development 81

II   Developers guide   83

III   Appendices   121

Preface

The Tellervo application is primarily designed for the measurement of tree ring widths and the organization and curation of the data, metadata and physical samples for dendrochronological research. It is cross-platform (running on all Java 7 and later enabled operating systems including Windows, MacOSX and Linux) and open-source. It includes support for a variety of standard measuring platforms including Velmex, Lintab and Henson.

Tellervo can be run as either the full desktop application that communicates and stores data in a database server, or in the much simpler Tellervo-lite mode that stores data as simple data files on your computer.

This manual is divided into two main sections, the first for users, the second for developers. Tellervo is open source software (see the details of the license on pages 199–204), so you are welcome to inspect and edit the code. The second part of this manual will help you do that.

Over the years Tellervo (and it's predecessor Corina) have been developed by many people, all of whom are listed within the about box of Tellervo. We would like to thank the many people that have tested the applications especially: Charlotte Pearson; Carol Griggs; Brita Lorentzen; Jess Herlich; LeAnn Canady; Kate Seufer; and many undergraduate and postgradutes students at Cornell and the University of Arizona.

We hope that you find Tellervo useful and look forward to hearing your feedback.

Part I

User Guide

Chapter 1

Installation

1.1 Introduction

Tellervo is a cross-platform and open source application primarily aimed at dendro data collection and management.

Tellervo can be run in one of two 'modes': full Tellervo and Tellervo-lite. Tellervo running in full mode is an enterprise-style tool requiring access to an installation of Tellervo server. Tellervo server manages all data and metadata within a relational database, based on the Tree Ring Data Standard (TRiDaS – Jansma et al., 2010). Tellervo provides a host of advanced features such as: integrated 3D mapping; field data collection with mobile devices; graphing; metadata management; barcode-based curation; dendro data indexing; crossdating; secure access permissions system; collaborative sharing of data; amongst others.

For users looking for a simple tool to just measure dendro samples and edit existing data files, the cut-down Tellervo-lite may be a more appropriate option. Tellervo-lite is a standalone desktop application that doesn't require access to a server. It provides a very similar interface for measuring samples as the full Tellervo but instead of the data being stored and managed within the Tellervo server, they are saved to one of a wide variety of legacy dendro data file formats. The lack of relational database, however, means that many of the advanced functions offered by the full Tellervo (e.g. 3D mapping, metadata management; curation etc) are not available within Tellervo-lite.

Unless specified otherwise, this manual describes the functionality of the full Tellervo application with the exception of chapter 2 which outlines all the behaviour specific to Tellervo-lite.

Tellervo is made up of two packages: the Tellervo desktop application and the Tellervo database server. Tellervo was designed primarily for laboratories with multiple users, each running the Tellervo desktop application on their own computer connecting to a single central server containing the lab's data. The Tellervo server is only required if you are running the full Tellervo. If you intend to use Tellervo in lite mode then you can skip the server installation and move directly to section 1.3.

1.2 Server installation

To make use of all the features of Tellervo desktop you will require access to a Tellervo server. If you are running Tellervo in a lab where the Tellervo server has already been set up by your systems administrator, all you will require is the URL of your server and your login credentials. Otherwise you will need to install Tellervo server yourself.

How you install Tellervo server will depend on how you intend to use it. In medium to large laboratories, you would typically install the server package on a dedicated server. However, in smaller laboratories this is probably not necessary or feasible. In which case it is perfectly fine to install the server on the same machine that is connected

to your measuring equipment. This is likely to be the situation if you simply want to try out Tellervo, if you don't have a separate server, or if you do not work in a multi-user laboratory.

The Tellervo server is made up of a number of components, which unlike the desktop client, can't be easily combined together into cross-platform packages. Although all the constituent components are open-source and available for all major platforms, building and maintaining separate packages for each platform is too large a task for a small development team. To conserve resources, we therefore made the decision to utilize Virtual Machine technology to ensure that the Tellervo server could still be run on all major operating systems. This means that we can package the Tellervo server for a single operating system (Ubuntu Linux) and then distribute it as a Virtual Appliance that can be run as a program on your preferred operating system.

The Tellervo server is therefore available via three methods. The first is as a VirtualBox* Virtual Appliance which can be run on any major operating system and we strongly recommend that you stick with this option unless you are experienced with Linux and running servers. The second option is to intall the native Ubuntu package on an Ubuntu Linux server. The source code for the server is also available so it is perfectly possible for more experienced users to set up the Tellervo server to run natively on other platforms. To do this you will require a good knowledge of Apache 2, PHP and PostgreSQL. Choose the most applicable method and follow the instructions in the following sections.

1.2.1 Install as Virtual Appliance (recommended method)

To run the Tellervo server Virtual Appliance, you will first need to download and install VirtualBox from `https://www.virtualbox.org`. Installation packages are available for Windows, MacOSX, OpenSolaris and many Linux distributions.

Once you have VirtualBox installed, you will then need to download the Tellervo server from the Tellervo website `https://www.tellervo.org/download`. This package contains a bare-bones Ubuntu Linux server with everything required to run the Tellervo server installed and ready to use. As VirtualBox, the entire Ubuntu operating system and Tellervo server components are all open source there are no license fees to pay.

1. Open VirtualBox and go to *File → Import Appliance*

2. Press the choose button and locate the virtual appliance file that you downloaded from the website

3. Rename the server if you choose, then press the finish/import button

4. Once the server is installed, highlight it in the virtual machine list and press the 'settings' button

5. Go to 'Network' and choose 'Attached to Bridged Adapter'. You may also like to give the system more RAM if you have powerful machine

6. Click ok, then back on the main page, press the 'run' button

7. Read and accept the information about how to gain and release control of the keyboard in VirtualBox

8. The server will boot and eventually present you with a command line login screen. Log in with the following temporary details but please note that as is normal in command lines the password will be hidden:

    Username : tellervo

    Password : dendrochronology

9. Once you are logged in, the server will automatically ask you a series of questions. The configuration will finish by testing your new server (see figure 1.1).

---

*Note that the Tellervo appliance is provided in the open standard format OVA. You should be able to run the appliance in other Virtual Machine applications (e.g. VMWare, Citrix etc) but this is not tested or supported.

10. Note down the URL of your new Tellervo webservice as you will need to enter this when you start your Tellervo desktop client. If you need to know the URL at a later date you can run the tests again by typing:

    ✎   `tellervo-server --test`

11. You can now install and run the Tellervo Desktop application (see section 1.3)

To save on download size and disk space only the essential packages to make the server run have been installed. This means there is no graphical interface just a command line. Hopefully this should not be a problem as once set up, the only interaction needed with the Virtual Appliance will be through the normal Tellervo desktop application. If you would prefer to use a graphical interface to the server this can be easily installed. See chapter 13 for further details.

There are a number of limitations caused by distributing the server software in this way. The operating system has already been configured to use networking without a proxy server, so if your computer requires a proxy then you won't be able to connect to the Internet. Please contact the developers for more details.

VirtualBox has a number of methods for providing network connectivity to the Tellervo virtual appliance. The default setup described above is to use **bridged networking**. This makes the virtual server appear as if it was another physical computer on your network with its own IP address. This works best in most situations, but can cause problems if your network administration is particularly strict. Note for instance that some in-

Figure 1.1: Screenshot of VirtualBox running the Tellervo server. The console contains the results of the tests run at the end of the configuration routine.

stitutions require the network card MAC address of new computers been registered before they will be allowed access to a network. It is also not suited to those installing Tellervo Server on a laptop which is used on different networks, especially via wifi. If you have problems with bridged networking, then it may be better to use one of the alternatives listed below.

The first network alternative is to use **host-only networking**. To use this you simple select your server in VirtualBox, click settings then on the network tab choose host-only adapter. This setting means that your server will only be visible to the physical computer you are running Tellervo server on. This may not be a problem (it may even be desirable) if you are installing for your personal use but of course it is not suitable if you are intending to share the server with colleagues. Note though that your server will not have access to the Internet so will not be able to download updates or use remote backup tools etc.

The second network alternative is to use **NAT networking**. NAT stands for Network Address Translation and means that requests and responses sent to/from your physical computer are intercepted and forwarded on to the virtual Tellervo server. Instead of Tellervo server having its own IP address, Tellervo clients wanting to talk to the server send requests to your physical computer. This has the benefit of being more likely to work if your institution has strict network policies. The drawbacks are that networking is slower, it takes a little more effort, and perhaps requires a better understanding of systems administration to set up.

With your virtual machine powered off, select it in VirtualBox, click settings and go to the network tab. Change the network type to NAT and then click the port forwarding button. Add an entry to the list:

Name - TellervoHTTP

Protocol - TCP
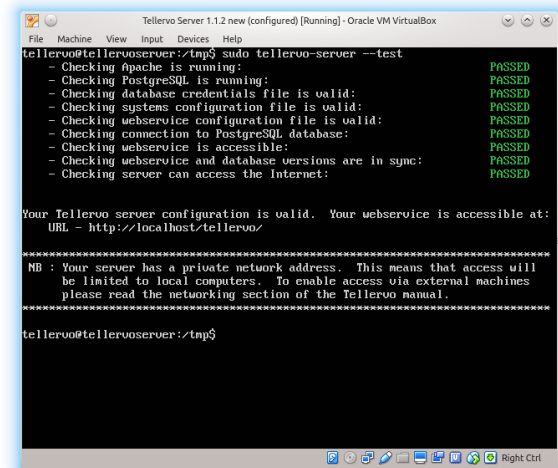
Host Port - 8080

Guest Port - 80

You can leave the other fields blank. The host port is the port that your Tellervo users connect to. The normal port that users connect to for web services is port 80, however, on most secure operating systems (e.g. Linux and OSX) you cannot port forward to ports below 1024, although it should be possible in Windows. If you are running Windows you may therefore be able to use a host port of 80. If you use 8080 (or any other port other than 80) you will need to include this in the URL of your webservice when you try to connect. The other complication of using NAT networking is that the URL provided by the Tellervo server configuration utility is incorrect as the server has no way of knowing that NAT is being used. The URL that you need to use to connect to is: `http://ip.of.your.physical.machine:8080/tellervo/`

1.2.2   Ubuntu native installation

If you are fortunate enough to be running Ubuntu then the native Ubuntu deb package is the best and easiest method for installing the Tellervo server, otherwise see section 1.2.1 to install the server as a Virtual Appliance. Note that due to limited resources we only support Tellervo Server on one version of Ubuntu at a time (typically the latest LTS release). Attempting to install Tellervo Server on a different version of Ubuntu and especially other distros such as Debian are likely to fail. If you really want to use a different operating system see section 1.2.3.

To install the Tellervo server in Ubuntu simply download the deb package from the Tellervo server `https://www.tellervo.org/download` and install with your favourite package manager. For instance, to install from the command line simply type:

```
sudo dpkg --install tellervo-server.deb
```

Almost certainly you'll be missing one or more required dependencies, in which cause dpkg will return an error. This can be fixed by running:

```
sudo apt install -f
```

This will install all the missing packages, and will automatically allow dpkg to run to completion. The package will automatically run a configuration script to assist with creating a database user, building the Tellervo PostgreSQL database, setting database permissions and setting up the Apache webservice. The configuration ends with a test routine to check all services are set up correctly and if so, will provide you with the URL of the newly configured Tellervo webservice.

1.2.3   Advanced install on other operating systems

The limited resources available for Tellervo development means that we have been unable to produce native installers for platforms other that Ubuntu. The VirtualBox method can be used on all modern operating systems so we strongly suggest you stick with this method. However, if you are an experience systems administrator and are feeling brave, it is possible to set up the Tellervo server manually.

The Tellervo server is essentially a PostgreSQL database running PostGIS and PLJava extensions accessed via a PHP webservice running on Apache 2. The current dependencies are listed in the control file at: `https://github.com/ltrr-arizona-edu/tellervo/blob/master/Native/BuildResources/LinBuild/ServerControl/control`. Note the interdependencies for PostgreSQL and PostGIS are very specific, so be careful to install the correct versions.

The basic procedure for installation is as follows:

- ▶ Install all dependencies
- ▶ Create PostgreSQL database

▶ Install PostGIS to database
▶ Install PLJava to database
▶ Restore Tellervo tables and data from template SQL file
▶ Set up a database user and provide access to the server in the pg_hba.conf file
▶ Give this user read and write permissions to the database
▶ Copy the webservice code into a web accessible folder
▶ Set up Apache to see this folder by creating an entry in the sites-enabled folder
▶ Restart PostgreSQL and Apache and check you can access the webservice from a web browser

A great deal of the complexity in setting up databases and running the server are hidden in the standard Ubuntu package through the tellervo-server script. The script is highly tailored to Ubuntu so is unlikely to work on another OS, but reading it may helpful.

1.3 Installing the desktop application

Installation packages for the Tellervo desktop application are available for Windows, MacOSX and Ubuntu Linux. Tellervo can also be run on other operating systems as long as they support Java 8 or later (see section 16.11, page 93 for more information).

To install Tellervo, download the installation file for your operating system from `https://tellervo.org/download`. The website should provide you with a link to the installer for your current operating system:

Windows – Run the setup.exe and follow the instructions.

Mac OS X – Download then open the zip file and drag the Tellervo.app into your applications folder. If you want to use a measuring platform you may need to run the 'fixserialportpermissions.sh' script located in the package.

Ubuntu Linux – A deb file is available which was designed for use on Ubuntu distributions but should work on any Debian based system. Install using your favorite package management system or from the command line like this: e.g.

```
sudo dpkg --install tellervo.xx.xx_all.deb
```

On Ubuntu and similar distributions, the package should add a Tellervo shortcut to your applications menu. Alternatively you can start Tellervo from the command line by typing tellervo.

Unix – The installation script will work on most flavours of Unix and Linux. You will need to install Java, RXTX serial port library and JOGL graphics libraries manually though.

Once you have installed your Tellervo Desktop application and you have access to a Tellervo server you are now ready to launch Tellervo for the first time.

1.3.1 First time launch

When you launch Tellervo for the first time you will be presented with a setup wizard (figure 1.2). Following the wizard to configure the main settings required before you can begin to use Tellervo. If you want to re-run this wizard at any time you can do so from the entry in the Help menu. You can also manually edit all these settings from the Tellervo preferences dialog which can be found in *Edit ➔ Preferences*.

The pages of the wizard include:

Tellervo mode – asks the user to choose whether they want to run Tellervo in full or lite mode.
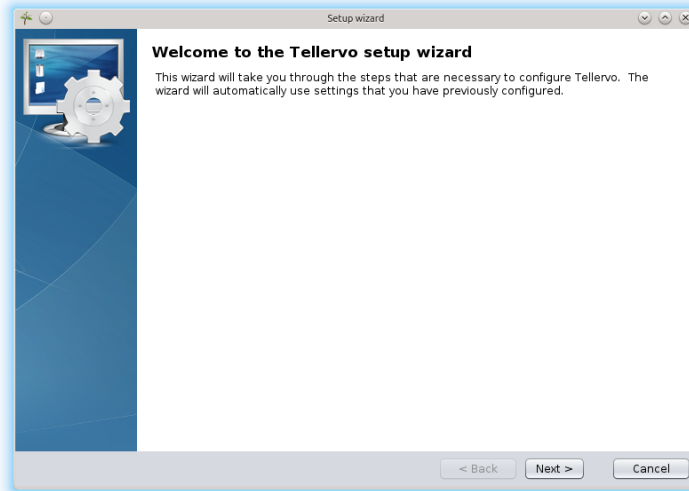
Figure 1.2: The Tellervo setup wizard will launch the first time you start Tellervo.

Network connection – this configures how your computer accesses the internet. Most users will be able to use the default 'Use system default proxy settings' option here, but if you know that your computer is behind a corporate proxy server you may choose to manually provide the settings.

Configuring the Tellervo server – Tellervo comes in two parts: the Tellervo desktop client that you are using; and the Tellervo server which runs the database that stores your data. If you are working in a lab your systems administrator may have already set up the Tellervo server and given you the URL to connect to. Alternatively, you may have already installed the Tellervo server yourself. If so the installation program should have given you the URL. If you don't have access to a Tellervo server yet, you should close this wizard, then go to the Tellervo website and download it.

Measuring platform configuration – the next page enables you to configure measuring platform hardware attached to your computer. Some measuring platforms have fixed settings in which case the port settings will be set automatically, but others can be changed in the hardware and must be set explicitly here. Use the 'Test Connection' button to make sure that Tellervo can successfully communicate with your platform.

Once you have completed the wizard you will be presented with a dialog (figure 1.3) for logging in to your Tellervo server.

The username and password details requested are your Tellervo login credentials (not your system or network credentials) provided to you by your systems administrator. If you are using your own Virtual Appliance server, the default admin user details are provided in section 13.5.1, page 72. The dialog gives you the option for saving your username and/or password if you prefer. We recommend using this feature only on personal machines. You may choose to cancel the login if you like and Tellervo will continue to load, however, you will not have access to the Tellervo database therefore very few functions will be available to you.
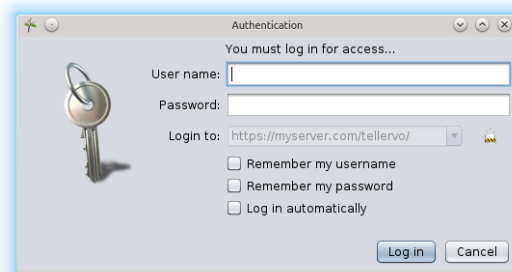
Figure 1.3: Tellervo server login dialog.

1.3.2 Mapping support

Tellervo includes 3D mapping for visualization of sam-
pling locations. Although this is not necessary for most
tasks, to make use of the mapping functions you will require a OpenGL 3D capable graphics card. To check whether
your computer already supports 3D mapping, open Tellervo, go to Admin, then Site map. Tellervo will warn you
if your graphics card is not supported.

All MacOSX computers should automatically support OpenGL. Most Windows and Linux computers made since
2006 should also support OpenGL, however, this does require proper drivers to be installed. In some cases Windows
computers may include a compatible graphics card, but may only have the default Windows video drivers installed.
If you are having trouble with the mapping in Tellervo make sure you have installed the most recent drivers for
your graphics card. Linux users may be required to install proprietary graphics drivers.

The mapping component of Tellervo makes use of NASA's open source World Wind Java. NASA's website
http://worldwind.arc.nasa.gov/ contains further information and instructions that you may find helpful if
you are having problems getting the mapping to work.

1.4 Upgrading Tellervo desktop

There are no special requirements for upgrading the Tellervo desktop client. You need simply install the new
version over the top of your previous version. Any personal settings will be maintained after the upgrade.

1.5 Upgrading Tellervo Server

Complete instructions for upgrading Tellervo Server run both natively or in Virtualbox can be found in section
13.3.

1.6 Uninstalling

We understand that Tellervo will never suit the requirements of all users, but as an open source product, we would
really appreciate feedback as to why it didn't work for you. Without this feedback it is difficult to prioritize future
development.

1.6.1 Tellervo desktop application

For Windows users, Tellervo desktop can be uninstalled using the standard add/remove programs feature in
control panel, or via the item in the Tellervo start menu. Mac users should simply delete the application from their
applications folder. Linux users should use their prefered package management tool e.g. from the command line:

```
    sudo dpkg --remove tellervo
```

1.6.2 Tellervo server

If you are running the Tellervo server as a virtual appliance simply follow the uninstall instructions for VirtualBox.
Please note that uninstalling the Virtualbox Tellervo server will delete your Tellervo database and all the data it
contains. Make sure that you export any data you need before uninstalling.

If you are running Tellervo server on a native Linux computer, you should use your preferred package mangement tool. To remove the server software but leave the database in place:

```
sudo dpkg --remove tellervo-server
```

To do a complete removal of the software and database then you should run:

```
sudo dpkg --purge tellervo-server
```

Chapter 2

Tellervo-lite

Tellervo-lite is the form that the Tellervo desktop application takes when you disable integration with the Tellervo server. The basic interface of Tellervo-lite is very similar to the full Tellervo, however, it lacks many of the advanced features such as 3D mapping, curation, and metadata management. If you are looking for a easy to use tool for simply measuring dendro samples and editing existing data files, then Tellervo-lite may be the best option for you.

Tellervo-lite is a 'mode' of the standard Tellervo application, not a smaller separate application. There is a unified installation procedure for installing Tellervo, and the switch to 'lite' mode is done during the configuration stage the first time you launch the application. If you have not installed Tellervo yet, please do so now by following the instructions on page 9.

2.1   Launching Tellervo-lite

The first time you launch Tellervo you will be presented with the setup wizard. The first question asks whether you'd like to use the standard Tellervo or Tellervo-lite. If you have decided to convert to Tellervo-lite mode at a later date (or vice versa), you can make the change by going to *Help* ➔ *Setup wizard* to relaunch the wizard, or by going to *Edit* ➔ *Preferences* and ticking the 'disable databse integration' checkbox on the network tab. Once you have launched (or relaunched) Tellervo in lite mode you will see the main screen shown in figure 2.1.

2.2   Tellervo-lite interface

The main Tellervo-lite interface is split into two parts. On the left is the series list, which contains all of the measurement series present within the current data file. On the right are two tabs: the data tab which contains the ring measurement values table and a graph representation; and the metadata tab which contains some basic metadata fields. Depending on the file format you intend to use, the series list may have one or more measurement series. For instance the Tucson RWL data format is capable of storing multiple series, whereas the Sheffield format can contain just one.

2.2.1   Data tab

The data tab contains three main panels: data table; graph; and remarks panel, all out which can be resized to suit your needs. The remarks panel enables you to tag individual rings with features, however, there is currently no support for saving this information in legacy files so it's not particularly useful in the context of Tellervo-lite and is therefore collapsed by default. The data table provides a way to view and edit the ring width data for your
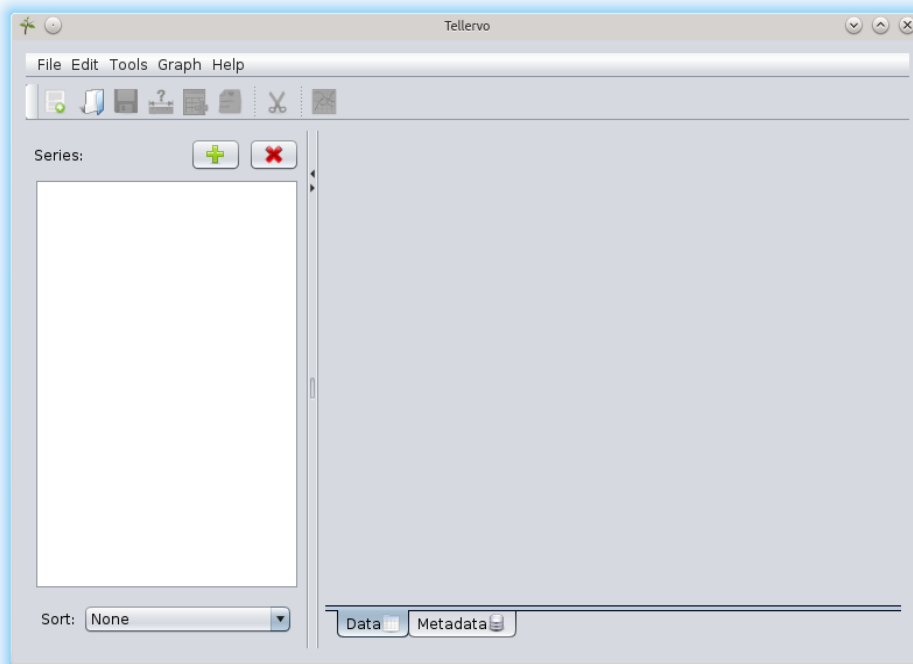
Figure 2.1: The main screen when Tellervo is launched in lite mode.

series, and is arranged in decades. You can enter and edit data within the table directly by clicking the cell and typing in a value.

The graph panel shows a basic chart of the ring width data and has it's own toolbar for adding/removing features, zooming etc.

At the bottom of the data tab is a status bar providing some summary information about the series. It includes: the calendar being used (whether absolute or relative); year and size of the currently selected ring; the variable being displayed (whole ring width; early/latewood width etc); the current measurement units; and some basic statistic. Clicking on the final three items, gives you options to change what is displayed.

2.2.2   Metadata tab

The metadata tab in Tellervo-lite provides a very generic and simplified method for entering the most common metadata to legacy dendro data files. The type of metadata stored in legacy data formats varies substantially, from essentially none (e.g. Belfast format) to quite rich (e.g. Heidelberg format). The time it would take to provide, maintain and support a metadata screen tailored to each format would be prohibitive. Besides, if you are interested in storing rich dendro metadata you should really be using the full version of Tellervo! The metadata screen in Tellervo-lite therefore provides the ability to add/edit a few of the very core metadata fields supported by most data formats.

At the top of the metadata tab is a pull down menu for specifying how the series are grouped. This is only relevant if you have multiple series in your file but will affect how the metadata fields behave and how files are saved depending on the output format you choose. For example, if you specify that the series are all from the same site/object, then then site/object name will be syncronised across all the series. Similarly, if the series are from the same tree, then the species field will be synchronised. If you choose to save a to a file format that expect all series

to be from the same site, yet you specify the series are from different sites, then Tellervo will save each series to a separate file.

2.3 Measuring a new sample

To begin measuring a new sample you should start by clicking the *File* ➜ *New* menu or by clicking the corresponding button on the toolbar. A new 'Untitled' series will appear in your sample list. Next, switch to the metadata tab and enter the name for this series and any other of the metadata fields that are desired. You can now switch back to the data tab. You can add as many series as you like by clicking the add series button at the top of the series list. You can also sort the series using the drop down menu at the bottom of the panel.

Before you begin measuring you need to tell Tellervo what sort of measurements you are making: whole ring widths; or early/latewood widths (the default is whole ring widths). To specify early/latewood widths you need to go to *Edit* ➜ *Measuring mode...* ➜ *Early and latewood widths*. If you use this menu after you have already measured some rings you will be warned that Tellervo will delete the data you have already collected. Once in 'early and latewood widths' measuring mode you will be able to choose which data is displayed in the table by clicking the variable box on the status line and choose between: Ring width; Earlywood width; Latewood width; Early/Latewood width.

To begin measuring your sample you can now go to *Edit* ➜ *Start measuring*, press the start measuring button on the toolbar, or you can press F5. While measuring you should be provided with audible feedback for each ring measured with a more pronounced sound made every 10th ring. If there is a problem communicating with your measuring hardware, check your settings in the preferences dialog. If you still have problems contact the Tellervo developers by going to *Help* ➜ *Email developers*, making sure you include details of your hardware and any error messages that you receive.

If your sample is already dated and/or you already know how many rings your sample has, then you can initialise your data matrix using the button on the toolbar or the menu item in the edit menu. This gives you a dialog requesting date and ring number information which can be useful for those of you that use the skeleton plotting method prior to measuring.

Note by default Tellervo labels rings as relative years beginning in 1001 (a convention from the days of mainframe computers and punchcards). If your sample is dated, you should explicity tell Tellervo either using the initialise grid function prior to measuring, or by going to *Tools* ➜ *Redate* once you've finished.

Depending on the measuring platform hardware you have, you will see some variation of the measuring panel in figure 2.2. The left display holds the absolute position of the last ring boundary (for devices that measure cumulatively), the middle display holds the last recorded measurement width and the right display holds the current position of the measuring plaform (for devices that report live measurements). The right-hand display is useful for devices that don't have a physical display such as the Lintab.

Tellervo supports the measuring of rings both individually and cumulatively. We feel that it is easier and more accurate to measure rings individually, that is to say the device is reset to zero after each measurement. If a device accepts requests to reset measurements (e.g. Quadra Chek boxes) or if it automatically resets itself to zero after recording a measurement (e.g. EVE IO) then this procedure is used by Tellervo. In this case the user begins measuring by setting the display to zero, then moves the platform to the end of the ring, then either

Figure 2.2: Measuring control panel. The number of digital displays will vary between 1 and 3 depending on what type of measuring platform hardware you have.

presses the 'measure' button on the hardware device or the 'record' button on the screen.

> 💡 If your device does not have a physical 'measure' button you don't need to click on the 'record' button each time. Instead click on the 'turn on mouse trigger' button and then you can use your mouse's left button as a measure button regardless of where the mouse is on the screen. This means you don't need to lift your eyes from your microscope to ensure you are clicking the button correctly. When you're finished measuring press escape to exit this mode. Alternatively you can use the tab key to ensure the record button has focus and then you can use the spacebar on your keyboard.

Certain devices (e.g. Boekler Microcode boxes) do not listen for requests to reset to zero. In this case to measure each ring individually, you would need to manually reset the reading to zero following each measurement. This would of course be extremely tedious. In this situation Tellervo measures cumulatively from the beginning of the first ring and calculates the ring width based on the previous ring boundary position. With this method you must be careful not to knock your sample, and you must also take special care when altering radii to navigate around problem structures. If you do knock your sample, the best way to recover is to reset your platform to zero and press the measure button. Next, press the 'stop measuring button', manually fix the values in the data table, then begin measuring again from where you left off.

If you are in 'Early and latewood widths' measuring mode the measurements are made and sent to the data table in pairs. The first measurement should be of the earlywood of the ring, and the next value the latewood measurement. Whether you are currently measuring early or latewood is indicated as a message at the bottom of the measuring panel.

2.4   Loading existing files

If you have existing data files that you would like to examine and/or edit, these can be opened using the *File* ➜ *Open* menu or the equivalent button on the toolbar. Through the use of the TRiCYCLE conversion library (Brewer et al., 2011), Tellervo supports many legacy dendro data formats. Select the correct format from the format list in the dialog and then select the file you'd like to open. You can now make changes to the data and/or metadata just as if it was a new series. Ring width values can be changed manually just by clicking on the value in the table and typing in the new value, or you can use the *Edit* ➜ *Start measuring* menu to remeasure rings.

2.5   Inserting, deleting and remeasuring rings

If you find you need to insert or delete a ring in your dataset, the easiest thing to do is to right click on the cell in question and use the popup menu. You can also find similar options in the edit menu. You will notice that in the popup menu there is also the option of tagging a ring with various features (e.g. frost damage, false ring etc). This feature exists in Tellervo-lite, however, there is currently no support for saving this information to legacy data files.

If you want to remeasure a ring (or series or rings) you need simply highlight the ring in the data table and measure as normal. The new measurements will simply be pasted over the top of the old ones.

2.6   Saving

Saving your changes is simply a matter of going to *File* ➜ *Save* or *File* ➜ *Save as*, or using the corresponding toolbar icons. If you are saving to a new file, then you will need to choose the format you'd like to use from the file save dialog. There is no need to add the file extension for formats that have a standard extension as Tellervo will automatically add this for you.

If you have loaded an existing data file, you will be warned that overwriting it may result in the loss of some metadata. If your original file contained metadata not supported by the rudimentary fields in the metadata tab, then this metadata will be missing from the newly saved file. You may therefore prefer to use the *File ➔ Save as* option and save to a new file instead.

If you have multiple series in your file and you attempt to save to a format that only supports one series per file, Tellervo will warn you and ask whether you'd like to use the keycodes of the series as file names. If you answer yes the files will be saved within the specified folder with the keycodes as the file names, otherwise it will create a series of files with the same filename with a number suffix.

Chapter 3

Getting started

Once you have your Tellervo desktop application installed (see chapter 1) and you also have access to a Tellervo server (either via your lab network administrator or your own on as a Virtual Appliance) you are ready to start using Tellervo. Below are some basic instructions for performing common tasks in Tellervo followed by a number of more in-depth chapters.

3.1 Main window

When you launch Tellervo in full mode and login you will be presented with the Tellervo main window (figure 3.1). The screen is split into two main areas. On the left is a list of series that are currently loaded. On the right are a series of tabs that display data and information about the currently selected series. The tabs include:

Data – This is the main tab for showing ring-width data for the currently selected series. The tab is split into three: data tables contain the ring-width values; graph of the ring-widths; and a ring remarks panel where individual rings can be tagged with comments.

Metadata – The metadata tab records all the associated information about the currently selected series following the Tree Ring Data Standard (TRiDaS). Full details about metadata is given in chapter 5.

Components viewer – For series that are derived from other series e.g. indices, sums, chronologies etc, the components viewer displays all the series that are referenced by the currently selected series.

Dependents viewer – Displays all other series that rely upon the currently selected series. For instance if this series is used in one or more chronologies, then those chronologies will be displayed here.

Map – Contains a 3D map viewer with the location of all the series currently loaded and that have location data recorded in their metadata.

3.2 Measuring a new sample

Once your measuring platform has been configured, measuring your first sample is simple. To start a new measurement go to *File* ➜ *New* or click the 'new' icon on the toolbar. A dialog will appear where you can scan your sample's barcode, or press the button to enter metadata for your sample later. Barcodes minimize data entry errors and also speed up the process of measuring your samples. See section 10.2 for more information. Once you have scanned your barcode or pressed the button, you will then be presented with an empty Tellervo metadata screen.
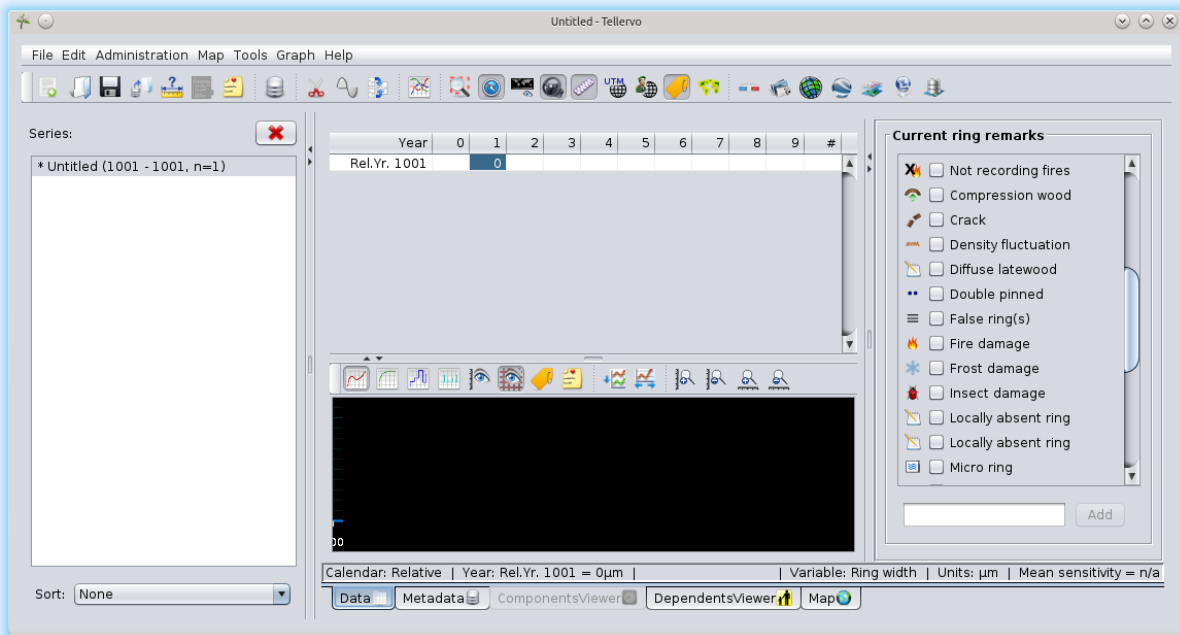
Figure 3.1: The main window in Tellervo. The screen is split into a two, with a list of open series in the workspace on the left, and a series of tabs on the right showing information about the currently selected series. The data tab is opened with a new series ready to receive measurements.

The next step is to fill out the metadata information. If you have used a barcode, nearly all of this metadata will be filled in for you, otherwise you will need to fill this out yourself. Details about metadata can be found in chapter 5, page 27. Once you have populated the metadata tab, you can then switch to the data tab.

Before you begin measuring you need to tell Tellervo what sort of measurements you are making: whole ring widths; or early/latewood widths (the default is whole ring widths). To specify early/latewood widths you need to go to *Edit* ➔ *Measuring mode. . .* ➔ *Early and latewood widths*. If you use this menu after you have already measured some rings you will be warned that Tellervo will delete the data you have already collected. Once in 'early and latewood widths' measuring mode you will be able to choose which data is displayed in the table by clicking the variable box on the status line and choose between: Ring width; Earlywood width; Latewood width; Early/Latewood width.

To begin measuring your sample you can now go to *Edit* ➔ *Start measuring*, press the start measuring button on the toolbar, or you can press F5. While measuring you should be provided with audible feedback for each ring measured with a more pronounced sound made every 10th ring. If there is a problem communicating with your measuring hardware, check your settings in the preferences dialog. If you still have problems contact the Tellervo developers by going to *Help* ➔ *Report bug on last transaction*, making sure you include your email address and any further information.

If your sample is already dated and/or you already know how many rings your sample has, then you can initialise your data matrix using the button on the toolbar. This gives you a dialog requesting date and ring number information which can be useful for those of you that use the skeleton plotting method prior to measuring.

Note by default Tellervo labels rings as relative years beginning in 1001. If your sample is dated, you should explicity tell Tellervo either using the initialise grid function prior to measuring, or by going to *Tools* ➔ *Redate* once you've finished.

Depending on the measuring platform hardware you have, you will see some variation of the measuring panel in figure 3.2. The left display holds the absolute position of the last ring boundary (for device that measure cumulatively), the middle display holds the last recorded measurement width and the right display holds the current position of the measuring plaform (for devices that report live measurements). The right-hand display is useful for devices that don't have a physical display such as the Lintab.

Tellervo supports the measuring of rings both individually and cumulatively. We feel that it is easier and more accurate to measure rings individually, that is to say the device is reset to zero after each measurement. If a device accepts requests to reset measurements (e.g. Quadra Chek boxes) or if it automatically resets itself to zero after recording a measurement (e.g. EVE IO) then this procedure is used by Tellervo. In this case the user begins measuring by setting the display to zero, then turns the platform to the end of the ring, then either presses the 'measure' button on the hardware device or the 'record' button on the screen.
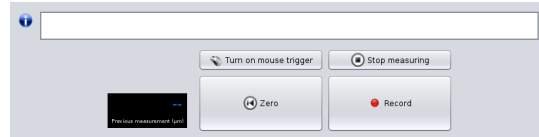
Figure 3.2: Measuring control panel. The number of digital displays will vary between 1 and 3 depending on what type of measuring platform hardware you have.

💡 If your device does not have a physical 'measure' button you don't need to click on the 'record' button each time. Instead click on the 'turn on mouse trigger' button and then you can use your mouse's left button as a measure button regardless of where the mouse is on the screen. This means you don't need to lift your eyes from your microscope to ensure you are clicking the button correctly. When you're finished measuring press escape to exit this mode.

Certain devices (e.g. Boekler Microcode boxes) do not listen for requests to reset to zero. In this case to measure each ring individually, you would need to manually reset the reading to zero following each measurement. This would of course be extremely tedious. In this situation Tellervo measures cumulatively from the beginning of the first ring and calculates the ring width based on the previous ring boundary position. With this method you must be careful not to knock your sample, and you must also take special care when altering radii to navigate around problem structures. If you do knock your sample, the best way to recover is to reset your platform to zero and press the measure button. Next, press the 'stop measuring button', manually fix the values in the data table, then begin measuring again from where you left off.

If you are in 'Early and latewood widths' measuring mode the measurements are made and sent to the data table in pairs. The first measurement should be of the earlywood of the ring, and the next value the latewood measurement. Whether you are currently measuring early or latewood is indicated as a message at the bottom of the measuring panel.

While you measure your sample you can flag features in a ring by right clicking on any cell in the table and selecting one or more of the standard notes (see figure 3.3).

Tellervo supports all standard TRiDaS remarks including: fire damage; frost damage; crack; false ring(s); compression wood; tension wood; traumatic ducts; single pinned; double pinned; triple pinned and many others. Rings that include remarks are indicated by the relevant icon in the data screen. Depending on your method of work, this can be useful for keeping track of sample pin holes. For instance, if a missing or false ring is discovered after a sample has been pinholed, the offset in pinholes can be easily seen without resurfacing the sample.
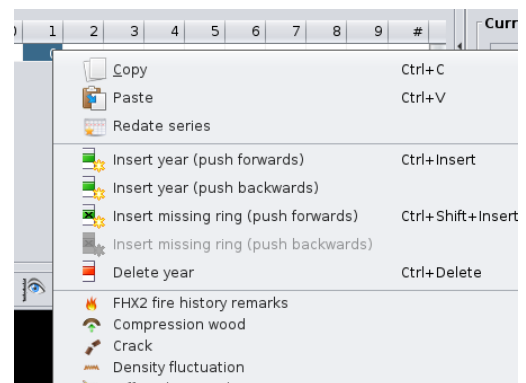
Figure 3.3: Right click context menu showing some of the options for adding remarks to rings.

In addition to the right click menu, you can also access ring remarks by opening the remarks panel using the button on the toolbar. This panel gives the the ability to add free text remarks to individual rings. One final method for adding remarks is specific to fire history researchers. Standard FHX-style remarks can be added to rings by pressing the equivalent FHX character code key in the relevant ring cells. For instance pressing a lower-case 'a' in a cell will add the 'fire injury in latewood' remark.

The data screen also contains a status bar at the bottom. By clicking on the units section, you can switch between micron and 1/100th mm units. Tellervo understands the units being supplied by the measuring platform, therefore changes here are purely for display purposes only. If you have a platform that measures in microns, but prefer to see the values in 1/100th mm then you can use this feature. There are also options on the status bar that enable you to choose one of a variety of summary statistics about your series.

Once you have finished measuring your sample, you should then go to *File ➜ Save* to save your series to the database.

3.3   Opening existing data

If you have used traditional dendrochronology software, you are probably used to opening existing dendro data files from your computer. Tellervo works in a different way. All data accessed by Tellervo is stored within the central Tellervo database rather than in files. The database provides many benefits over file based storage, most importantly it means there is a high degree of security and integrity in your data.[*]

To use data that you have stored in existing data files you must first *import* your data into the Tellervo database. This gives you the opportunity to clean-up your data! For details of how to import your data see chapter 9, page 47.

Once you have data in your database, either by importing existing data files or measuring new samples, you can access your data through the database browser. This is accessed through the *File ➜ Open* or *File ➜ Open multiple* menus and an example of the dialog is shown in figure 3.4. The same database browser dialog is used in multiple places throughout Tellervo, e.g. when adding additional series to graphs and when choosing chronologies to crossdate against.

The database browser is divided into two main parts. On the left are the browse and search tabs, and on the right is the series table. Selecting options in the browse or search tab populates the series table on the right with all the series that match the specified criteria.

The browse tab shows a heirarchical tree view of the contents of your Tellervo database based upon the TRiDaS data model. The panel will be pre-populated with all the objects in your database but it is possible to 'drill-down' by right clicking on an object and choosing 'Expand branch'. Expanding an object for instance, will show all the elements associated with that object, and expanding an element will show all the samples associated with the specified element. To better understand the TRiDaS terminology please read chapter 5, page 27.

By double clicking (or right clicking and choosing 'Search for associated series') on an item in the browse panel Tellervo will search the database for all series that are associated with the specified entity. The results of the search will be shown in the series table on the right of the screen. This table shows basic metadata about each search and is sortable by click on any of the column headers. To open a series, simply select one of these series and click 'OK'. If the database browser is open in 'multiple series' mode, then you can use the arrow buttons to select multiple series to open in one go.

There is also a 'Show options' button on the database browser dialog. This adds additional advanced methods for filtering the series table to help you find the data you are interested in.

---

[*]This doesn't mean you don't have to backup your data though! Whoever is in charge of maintaining your Tellervo database should make sure regular backups are made—preferably offsite.
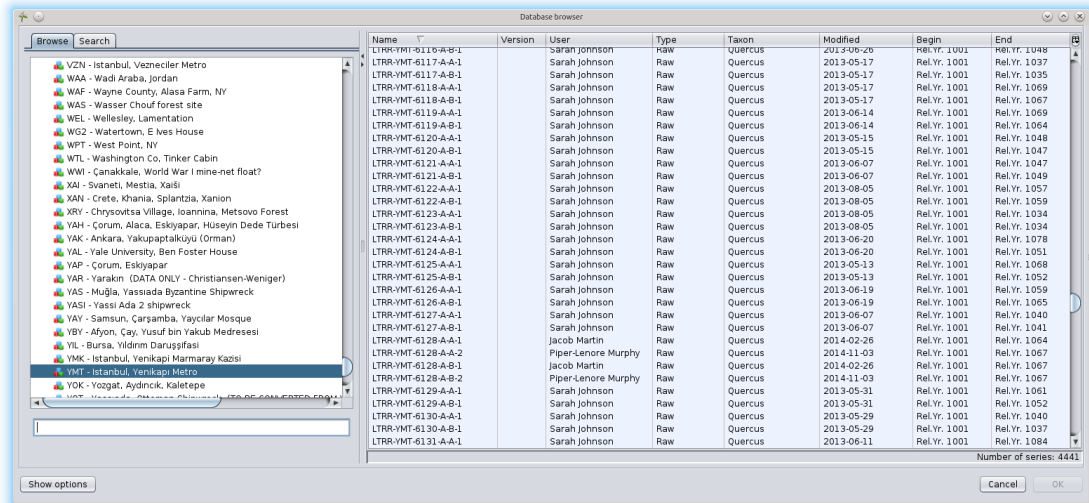
Figure 3.4: Screenshot of the database browser dialog.

At the bottom of the list of objects in the browse tab are two categories of 'tags': personal tags; and system tags. The tagging concept in Tellervo is borrowed from the world of social media. Just like you can tag a photo or blog post in social media tools with a tag, perhaps someone's name, a keyword, or location name, in Tellervo you can tag series. For instance with tags like 'needs checking', or 'in progress'. Any tags used will be displayed in the list and by double clicking on them, all series that have been tagged will be displayed. Tags can be personal (only visible to you) or system tags. The system tags are useful when you want to share tags with other users of your database.

3.4   Reconciling data

Tellervo has been developed not only for experience dendrochronologists, but as a tool for teaching students. It therefore includes a comprehensive 'reconciling' tool for supervisors to check the quality of measurements made by students. The reconcile dialog does a comparison of a measurement series made by a student with a references series of the same radius measured by the supervisor. The same dialog can also prove useful for comparing measurements from two experienced dendrochronologists when handling particularly difficult samples.

Open the series you would like to check as normal. Next go to *Tools ➜ Reconcile*. This will open a database browser window where you should then locate the reference series you'd like to compare against. The two series will then be opened in a reconcile window such as the one shown in figure 3.5.

The reconcile window contains: a data table for each of the series; a graph of the two series; and an information panel describing inconsistencies. The reconcile tool runs a number of tests:

Series length – this simply test checks that the two series are the same length

Trend agreement – this checks that the trend from year n to $n + 1$ is in agreement between the two series. For example if the ring-widths increase from 1950 to 1951 in the reference series then they should also increase in the test series. Where trends are different the pair of years are marked with a red border.

Error margin – this checks that the ring-width values in the reference and test series are within a 3% error margin. Where the ring-widths are outside this error margin, then cells are coloured red.
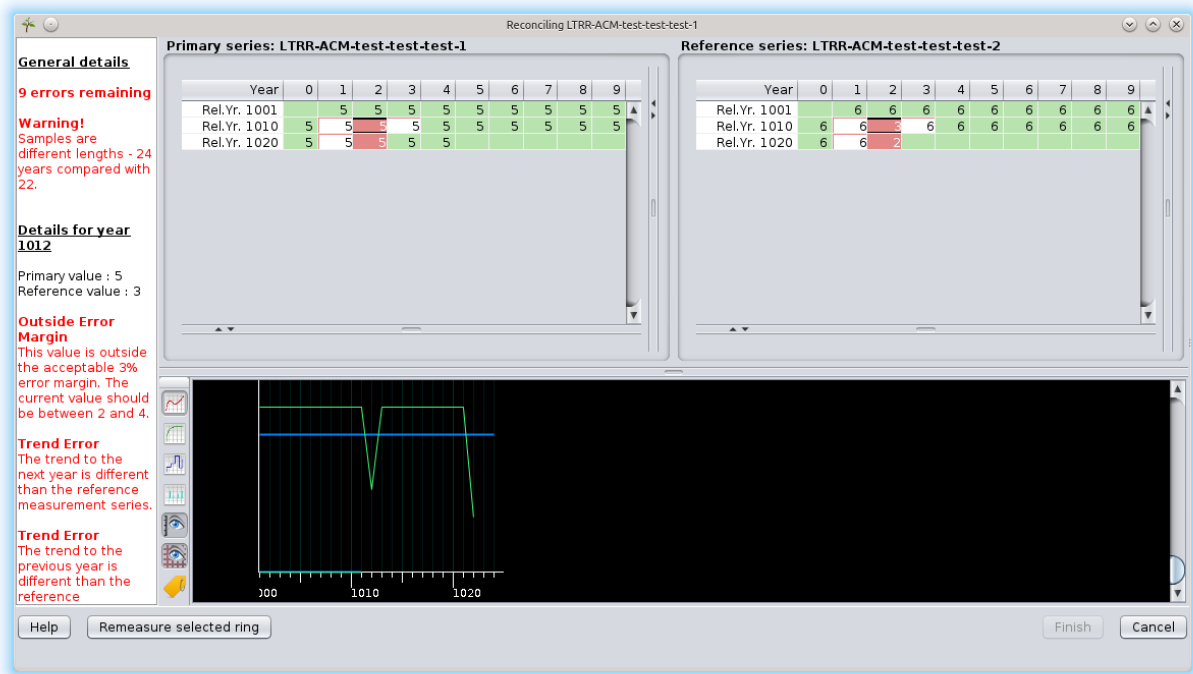
Figure 3.5: Example of the reconcile window showing the differences between two dummy datasets. The green cells show where the series are in agreement, and the red cells and cell-pairs with red borders are where there are conflicts.

In circumstances where the test series includes an extra, or is missing a ring then typically there will be a large amount of 'red' in the tables begin at the point of the error. Such errors are typically easiest seen in the graph.

Errors in the measurements can be fixed from within the reconciliation screen by directly editing the data tables. Right click popup menus can be used to insert or delete rings where necessary. There is also the option of remeasuring the current ring. The remeasuring tool allows you to remeasure the same ring multiple times to triple check you end up with an accurate measurement.

Once you have fixed all the errors you can then click the 'finish' button to commit the changes to the test series.

Chapter 4

Measuring platforms

Although it is possible to manually enter the ring-widths of your samples into Tellervo, it is normal to automate this process using a measuring platform. Tellervo supports the most common measuring platforms including Velmex and Lintab. However, please note that standard Lintab platforms use a proprietary communications protocol. Rinntech–the manufacturers of Lintab platforms–claim intellectual property rights over this protocol. During discussions between the Tellervo development team and Rinntech an agreement was reached whereby the Tellervo developers agreed not to release details of the protocol. In turn Rinntech has agreed to produce an adapter that can be attached to Lintab platforms so that they communicate with an open ASCII protocol. Users wishing to use Lintab platforms with Tellervo (or any software not developed by Rinntech) must therefore contact Rinntech and purchase an adapter.

Measuring platforms typically use serial ports to communicate to computers. In recent years computer manufacturers have been phasing out serial ports so you may need to purchase a serial-USB converter. Modern releases of MacOSX, Linux and Windows should support most serial-USB adapters out of the box, otherwise you must install the relevant drivers before continuing. Recent Lintab USB platforms use internal serial-USB converters so are treated in exactly the same way by Tellervo.

To begin, shut down your computer, attach your platform, then reboot and launch Tellervo. Next, go to the preferences window and open the hardware tab and you should see an interface that looks like figure 4.1. The same panel is shown in the setup wizard if this is the first time you have run Tellervo.

In the 'type' pull down menu, select the type of measuring equipment you are using. Note that this refers to the equipment that the computer is attached to, and not necessarily the measuring platform itself. For instance, Velmex platforms are typically connected through a Metronics digital readout device. Included in this list is the EveIO device which is an open-source device designed for the Cornell Tree-Ring Laboratory. Circuit drawings for this device can be obtained from the Cornell lab to enable Hensen measuring platforms to be used with Tellervo (and other software). If your measuring platform is not included in the list it should be relatively easy for us to add support so please get in touch and we'll see what we can do. Alternatively you could implement support yourself (either personally or by employing an independent developer). Technical details on how to do this are included in section 16.13, page 93.

Next you must choose the port that your platform is connected to from the pull down menu. In Windows this will be a COM port, in Linux and Mac this will be a /dev/xxx port. Depending on the type of platform you choose, you may also need to set various communication parameters. If these boxes are enabled, please check the documentation that came with your measuring platform to ensure these values are set correctly.

To check whether your platform is working, click the 'Test connection' button (see figure 4.2) and attempt to measure a few rings. Different measuring platforms have different capabilities. For instance, some include a physical switch for firing measurement events, others also include switches for resetting measurements to zero. Some platforms (e.g. Lintab) also continuously report the measurement values to the computer. So depending on the hardware you use, Tellervo will present the you with slightly different options.
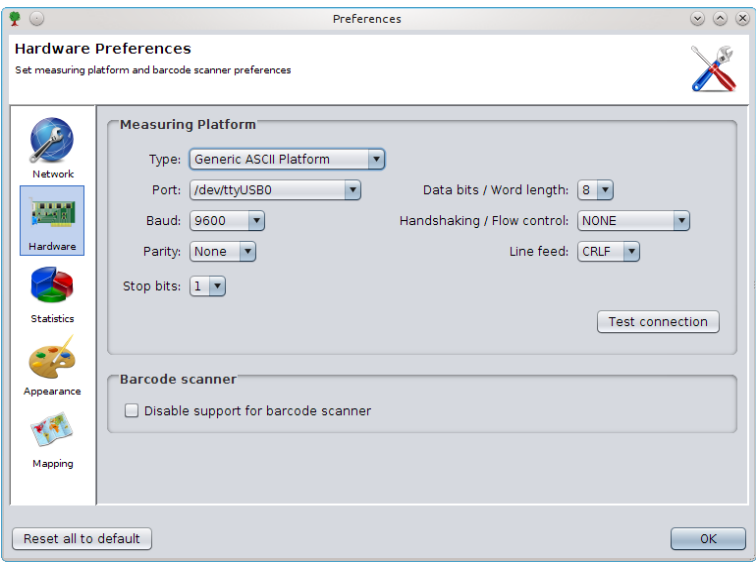
Figure 4.1: The hardware preferences dialog.

The test dialog includes information about the capabilities of your platform as well as a log window to show the raw information being received by Tellervo. If you are having trouble interfacing with your platform, you should send the communications log to the developers, along with as much information about your hardware as possible.

Once you are satisfied that you are getting the correct results from the measuring platform, click close on the test window and then close the preferences dialog to return to the Tellervo home screen.
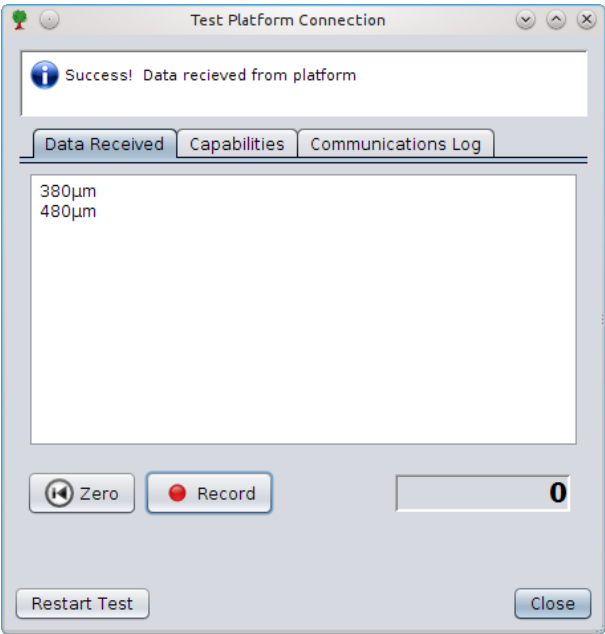
Figure 4.2: Testing the connection to a hardware measuring platform.

Chapter 5

Metadata

Metadata is 'data about data'. In Tellervo this means all the information associated with your physical samples and measurement series e.g. species, location, who measured it, dimensions, slope, soil type etc.

The metadata in Tellervo, and in fact the entire Tellervo data model, is based on the Tree Ring Data Standard (TRiDaS). Before you use Tellervo you may find it useful to read Jansma et al. (2010) so that you get a better understanding of the principles of TRiDaS, but a summary is also provided here.

5.1 Tree Ring Data Standard - TRiDaS

TRiDaS is an XML-based data standard for recording dendrochronological data and metadata. More than 80 dendrochronologists, computer scientists and specialists from research disciplines that rely on dendrochronology have so far contributed to its development, including dendroarchaeologists, art and architecture historians, ecologists, geologists and climatologists. The standard is therefore capable of recording the wide variety of metadata required by these different fields. TRiDaS builds upon other established standards, such as GML for the recording of locality information. The extensible nature of XML also means that TRiDaS can evolve to accommodate the changing needs of dendrochronologists over time.

TRiDaS includes a total of eight data entities: project; object; element; sample; radius; measurementSeries; derivedSeries; and value. Detailed descriptions of each of these entities are given below and their relationships are illustrated in figure 5.1.

> **A project** – is defined by a laboratory and encompasses dendrochronological research of a particular object or group of objects. Examples include: the dating of a building; the research of forest dynamics in a stand of living trees; the dating of all Rembrandt paintings in a museum. What is considered a "project" is up to the laboratory performing the research. It could be the dating of a group of objects, but the laboratory can also decide to define a separate project for each object. Therefore, a project can have one or more objects associated with it. Projects within Tellervo are managed with the *Admin ➜ Project Browser* interface (see section 5.5).

> **An object** – is the item to be investigated. Examples include: violin; excavation site; painting on a wooden panel; water well; church; carving; ship; forest. An object could also be more specific, for example: mast of a ship; roof of a church. Depending on the object type various descriptions are made possible. An object can have one or more elements and can also refer to another (sub) object. For instance a single file may contain three objects: an archaeological site object, within which there is a building object, within which there is a beam object. The list of possible object types is extensible and is thus flexible enough to incorporate the diversity of data required by the dendro community. Only information that is essential for

Figure 5.1: TRiDaS data model showing the relationships between data entities. Most of the entities having a simple hierarchical relationship (a project has one or more objects, an element has one or more samples.

dendrochronological research is recorded here. Other related data may be provided in the form of a link to an external database such as a museum catalogue.

An element – is a piece of wood originating from a single tree. Examples include: one plank of a water well; a single wooden panel in a painting; the left-hand back plate of a violin; one beam in a roof; a tree trunk preserved in the soil; a living tree. The element is a specific part of exactly one object or sub object. An object will often consist of more than one element, e.g., when dealing with the staves (elements) of a barrel (object). One or more samples can be taken from an element and an element may be dated using one or more derivedSeries.

A sample – is a physical specimen or non-physical representation of an element. Examples include: core from a living tree; core from a rafter in a church roof; piece of charcoal from an archaeological trench; slice from a pile used in a pile foundation; wax imprint of the outer end of a plank; photo of a back plate of a string instrument. Note that a sample always exists and that it can either be physical (e.g. a core) or representative (e.g. a picture). A sample is taken from exactly one element and can be represented by one or more radii.

A radius – is a line from pith to bark along which the measurements are taken. A radius is derived from exactly one sample. It can be measured more than once resulting in multiple measurementSeries.

A measurementSeries – is a series of direct, raw measurements along a radius. A single measurementSeries can be standardised or a collection of measurementSeries can be combined into a derivedSeries. The measurements themselves are stored separately as values.

A derivedSeries – is a calculated series of values and is a minor modification of the "v-series" concept proposed by Brewer et al. (2010). Examples include: index; average of a collection of measurementSeries such as a chronology. A derivedSeries is derived from one or more measurementSeries and has multiple values associated with it.

A value – is the result of a single ring measurement. Examples include: total ring width; earlywood width; latewood width. The values are related to a measurementSeries or a derivedSeries. In case of a measurementSeries the variable and its measurement unit (e.g. microns, 1/100th mm etc) are recorded as well. Tellervo supports both total ring width, and early/latewood values. Support for other variables is planned for a future version.

Working top to bottom, the TRiDaS entities are nested within each other. For instance a project contains one or more objects, which in turn contains one or more elements, and so on. The benefit of this is that you record data once and once only. In standard file-based dendrochronological software, when creating measurement series you are typically required to type the name of the site, the species of tree etc over and over again. This is not only time consuming, but very error prone.

Keeping data consistent is also difficult. For instance, if it was determined that a tree species was identified incorrectly, in existing file-based software, the user would need to locate all data series from this tree and manally update the metdata. This is not the case in Tellervo. A tree is represented just once in Tellervo and samples of this tree, and the subsequent measurement series reference this one entry. If metadata for this tree needs to be changed, the tree record is updated in just this one place. Because the measurement series obtain this information by reference, then all associated series are automatically kept up to date.
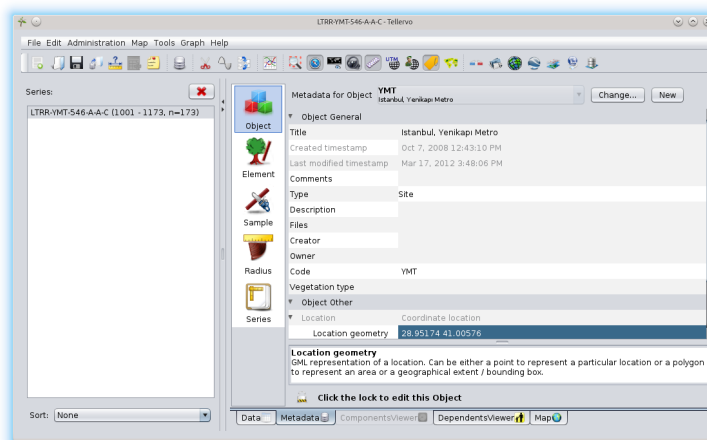
Figure 5.2: Example of the metadata dialog. The screen is showing the details of a TRiDaS object. Note that the location geometry field is highlighted and so a description of what is expected in this field is given below.

5.2 Entering sample metadata

The metadata for a series is viewed and edited on the 'Metadata' tab of the main window such as that shown in figure 5.2. You can see the interface is organized according to the TRiDaS data model with separate tabs for object, through to series.

When creating a new series, the metadata screens must be populated in order. This is necessary because of the nesting of entities described above. For instance, an element is associated with an object, so an object must be chosen before an element can be defined. Likewise, an element must be chosen before any samples of this element can be defined.

Much of the time the entities that you need will already be stored within the database. Instead of re-entering data, you simply need to select the existing entry from the database, saving a great deal of time. Depending on the situation, buttons will appear at the top of the dialog to let you 'choose' an entry from the database, 'revert' to the previously chosen entry, 'change' the existing entry to a different one from the database, or create a 'new' record.

Please note that the content of these metadata screens is kept read-only by default. To edit the values, you must first click the padlock icon to unlock the fields. When you have finished making changes you need to press the save button to write the changes to the database before moving to another metadata screen.

Very few of the metadata fields in the TRiDaS data model are mandatory, but a few are. In this case, these fields are highlighted with a red star. Note that whether a field is mandatory or not can depend on the other fields that have been filled in. For instance, the dimensions of an element are not required, but if dimensions are given then the units for these measurements must also be provided.

A number of the metadata fields are restricted with regards the values that you can enter. These are known as 'controlled vocabularies' in TRiDaS terms. Controlled vocabulary fields are represented by drop down menus. Similarly fields that expect numerical values (such as element dimensions) will only allow numbers. The final data entry method is through custom dialogs, for example there is a custom dialog for entering location information. This accepts coordinates in either decimal degrees or degrees minutes and seconds. Alternatively you can use data from a GPS handset by providing a GPS Exchange (GPX) format file containing the waypoints. The GPX format is the most common interchange format for GPS data. You can pick the relevant waypoint from the drop down menu. You can also preview the defined coordinates on a map using the 'view on map' button.
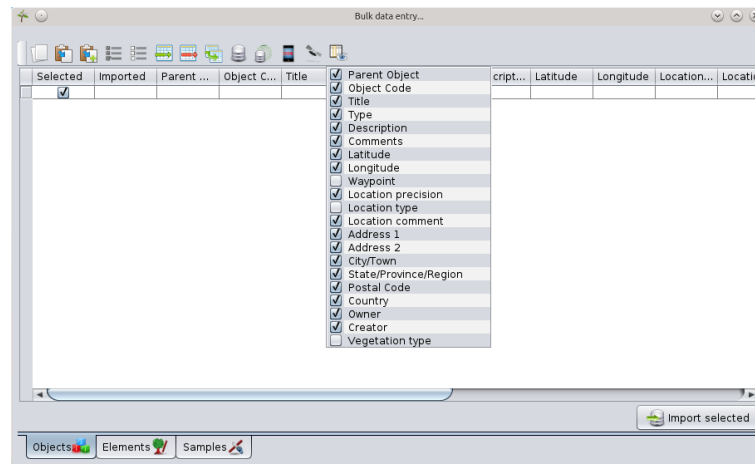
Figure 5.3: The bulk metadata entry screen. The 'show/hide columns' button has been pressed showing how the user can turn on and off particular columns.

> 💡 A popular open source GPS communication tool is GPS Babel. It is an easy to use application which can download data from the majority of GPS handsets. See `http://www.gpsbabel.org` for more information.

5.3   Entering bulk metadata

Entering metadata on a sample-by-sample basis works perfectly well, but does not necessarily fit best with the typical workflow of a laboratory. Samples do not typically arrive in a lab in ones and twos, rather in large quantities following a field excursion. In this case it is most efficient to enter all the metadata for the samples as they arrive. This is often best in terms of data accuracy as the metadata can be entered while the field notes are still fresh in the mind. If you have not yet been to the field, then we strongly recommend you use the ODK functionality in Tellervo (see chapter 6). This uses mobile devices with tailored forms to quickly collect metadata in the field, and have it entered directly into the database on your return.

To enable the efficient entry of lots of metadata stored in spreadsheets, field notes etc, Tellervo includes the bulk data entry interface. This can be accessed from the file menu and is illustrated in figure 5.3. There are three pages, one each for objects, elements and samples.

The interface is designed like a spreadsheet so as to be as familiar to users as possible. Each row of the table represents a new entry in the Tellervo database. The columns that are shown to the user are determined by the 'show/hide columns' button on the top right of the screen.

It is common for many of the metadata fields to be the same in a single field collection. For instance, when coring trees in a forest, they are often of the same species. Rather than requiring the user to repeatedly type the same data over and over, the 'copy row' button can be used to duplicate a record, and then the user can change the few fields that are different.

Another time saving feature is the ability to cut and paste back and forth between Tellervo and spreadsheet programs like Microsoft Excel. This can be particularly helpful if you already have large amounts of metadata entered into a spreadsheet. The best way to do this is to populate a row in Tellervo then copy this into Excel. This will then show you the order of columns and the format that Tellervo expects data to be in. If the data you try to paste data that Tellervo isn't happy with (e.g. text into a numbers field, or terms that don't match the Tellervo dictionary) then a summary table will list all the errors and allow you to cancel, or just paste the data that is valid.

When you have entered all the data you want, you can press the 'Import selected' button to write the records to the database. Start on the objects tab, then move to elements then samples.

5.3.1   Toolbar buttons

There are a number of additional features in the bulk data entry interface that can make data entry more efficient.

– Support for GPS data. The user can provide a GPS Exchange (GPX) format file containing the waypoint locations recorded in the field. Tellervo will add a waypoint column to the spreadsheet with a drop down menu which will automatically populate the latitude and longitude fields for the record.

– The samples page has the additional feature of creating barcode labels. Once you've finished importing your metadata, simply press the barcode button and a PDF file will be generated ready for printing.

– Another method that makes metadata entry even quicker is through the use of the Open Data Kit (ODK) features within Tellervo. ODK is an open source project that makes use of tablets and phones for data entry in the field. Details of how to use the ODK features in Tellervo are given in chapter 6.

– The Bulk Data Entry interface can be populated with records that are already in the database. This is a good way to check and edit metadata for a large number of records in one go, it also is a good way to export metadata out into other programs like Excel.

– Where records have coordinate information, Tellervo can use the Geonames service to lookup the country and closest town name and populate the table.

– The elements and samples tab have the option of using the 'quick fill' tool to prepopulate the table with new records. This is useful if you collect many trees and/or samples from these trees from a site all with similar metadata.

5.4   Metadata browser

The metadata browser interface provides a convenient way to view all the metadata within your Tellervo database. It can be accessed through the 'Administration' menu from the main screen.

The metadata browser contains two parts: a hierarchical representation of all TRiDaS entities in your database on the left; and a metadata viewer for the selected entry on the right. This interface is also the best method for fixing mistakes in your database.

Although Tellervo's database architecture maintains integrity within your data, it does come at the price of being a little more complicated to fix mislabelled series. For instance, what if you were to measure a series 'B' and assign it to sample ABC-138-A only later to realize you misread the label and it was in fact ABC-188-A. In a traditional file-based system, you would probably just need to rename the file you'd just created. In Tellervo however, you need to redefine the relationship of the series within the database and reassign it to the correct sample. This is best understood when looking at the hierarchical tree in the metadata browser. Hopefully you will see that you what you need to do is to move the series from its current position in the database to the correct one.

The reorganization of data in this way is achieved by right clicking on items in the hierarchical tree and choosing either 'merge' or 'reassign'. These functions are only accessible to database administrators. The reassign option simply moves an entry to a different parent in the database. The merge tool handles the more complicated situation where a entry has been made in two different places in the database and they need to be combined. For instance in the case described above if the erronously created ABC-138-A after (the correct) sample ABC-188-A had already

been created and populated. In this case the 'A' sample from ABC-138-A would be merged into ABC-188-A, bringing with it any radii and measurements. Hopefully the metadata for the merged series will be identical, but in the case of discrepancies, details are noted in the comments field of the finished entity. You should therefore take the time to check the finished entry and clarify any differences.

5.5   Project Browser

The project browser provides a similar interface as the metadata browser, with a list of all the projects within the database on the left and the details of the currently selected project on the right. Once projects are defined, objects can be assigned to projects in the metadata browser, bulk data entry screen, or main metadata tab.

5.6   Laboratory codes

Tellervo uses lab codes to refer to the hierarchical nature of the TRiDaS entities in the database. The separate parts of the code are delimited by hyphens and depending on the level of the entity you are referring to, will have a different number of parts. For instance, if you are referring to a tree (an 'element' in TRiDaS terminology) then the lab code will consist of just two parts: the object code and the element code. See figure 5.4 for an illustrated example.

Lab codes are used throughout Tellervo to describe TRiDaS entities. They can also be used in many places to specify entities that the user would like to choose. For instance, in the database browser, you can type the lab code for an object, element, sample, radius or series to search the system for all the series that match the specified entity. For instance entering 'ABC-5' would search for all series associated with element '5' from object 'ABC'.

Example: ABC-1-A-A-1

Element Code:
This tree grew at a site called 'ABC'. It was the first tree sampled at the site and received the element code of 1.

ABC-1

Element 1

Sample Code:
Two samples were taken from tree 1. These were labeled A and B. If there were more than two samples (tends to happen with charcoal), the third and fourth would be labeled C and D and so on. The lab code we are constructing is for the first sample, A.

ABC-1-A

Sample A   **Sample B**

Radius Code:
A sample may have many radii, the first radius receives the letter "A" and the second "B" and so on. For cores there is usually only one radius (A) to measure.

ABC-1-A-A

Radius A   **Radius B**

Series Code:
A radii will have at least 2 readings. The final letter in the lab code relates to the first or second reading. In this case we are constructing a code for the first reading '1'. The second reading would be '2' and so on.

ABC-1-A-A-1

Series 1   **Series 2**

Figure 5.4: Illustration of the how lab codes are built in Tellervo. Figure courtesy of Charlotte Pearson.

Field data collection

Accurate field notes are essential to ensure the greatest value is made of samples. With no metadata associated with a sample, it is of little more use than firewood. At the same time, though, time in the field is often very limited therefore there is pressure to record metadata as efficiently as possible.

Dendro field data collection is traditionally done through either handwritten field notebooks or paper forms. The trade-off between comprehensive metadata collection and time efficiency means that fieldnotes are often abbreviated with the best intentions to write them up fully in the evening or on the return to the laboratory. These sorts of pressures typically lead to the bare minimum of metadata being recorded.

In Tellervo, one solution to this problem makes use of the collection of open source tools known as Open Data Kit (ODK). ODK is a platform that utilises (Android) mobile phones and tablets as simple field data collection devices. It was originally designed to collect medical information in rural regions of developing countries and has proven to be extremely effective and reliable. Forms are designed and uploaded to the mobile device, and a simple sequence of questions enables the field researcher to enter text, categorical, numerical, GPS and photographic data. On return to base, the data is then uploaded from the device to a database for analysis. The system does not rely on mobile network signals that are typically weak or non-existent in remote field regions, rather storing the data on the device and then transfering the data at the end of the day once the researcher returns to base.

The hardware requirements for the mobile application are very low meaning it can run on entry level phones and tablets. The addition of rugged waterproof cases mitigates for difficult field conditions, and the storage of data on removeable flash memory cards means that data can often be recovered even when disaster strikes your device. Entering data via touch screens is now second nature to most people, enabling data entry speeds approaching or exceeding that of a traditional keyboard.

The field data collection system implemented in Tellervo is described in the following sections and by Brewer and Guiterman (2016).

6.1   Data transfer

There are two methods of transferring data from ODK mobile devices to the Tellervo database. The first and most convenient is via wifi and/or cellular network connections directly to the Tellervo server. This is quick and effective, however, it does require that your Tellervo server be available over the Internet. If you run your server on a private network or using the Virtualbox 'host-only' network configuration, then this will not be possible.

The alternative method is to manually transfer files back and forth between the computer running Tellervo Desktop and your mobile deivce. This can be done via a USB cable, by removing the SD-card from the device and reading it directly from the computer or via a variety of Android Apps. There are many methods for doing this and their availabilty will depend on the version of Android you are running.

6.2   Creating data entry forms

The metadata capabilities of Tellervo are extensive so producing a data entry form with all possible fields would be unhelpful. The fields required will vary dramatically depending on the researcher and the research questions being addressed. The Tellervo desktop application therefore has a tool for designing tailored forms for each field trip. This tool can be accessed via the *File ➔ Design ODK Form* menu (see 6.1).
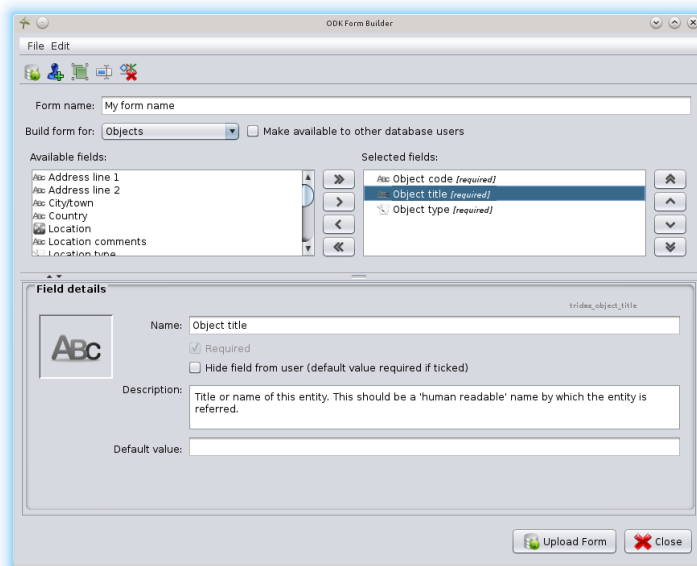
Figure 6.1: Screenshot of the Design ODK Form dialog.

The Design ODK Form dialog gives the user the ability to create two types of forms: the first for recording new objects (sites); and the second for recording elements (trees) and samples. The user should name the form they are creating, then select the form type, and then pick all the fields that they would like to include. Fields that are required by Tellervo and TRiDaS are pre-selected.

When you select a field you will see it's details below depending on the data type that it stores. You can then modifiy the field's properties. For instance you can change the name of the field which can be useful when the person using the ODK tool in the field is not familiar with the TRiDaS terminology. The field name 'element code' might not be intuitive whereas 'tree code' will be. There is also the option to alter the help text that goes with each field. This is another way to ensure the correct metadata is collected in the field. It is important though that you don't change the meanings of these fields as they will be mapped back to the original TRiDaS concept when you upload the data. For instance regardless of what you choose to name the 'element code' field, the contents of this field will be entered into the 'element code' field in the Tellervo database.

If you know in advance the most likely information that the ODK user will want to enter you can set a default value for a field. In this case the user can quickly confirm the entry is correct without having to enter it each time. You also have the option of hiding a field from the ODK user. This can be useful when you are certain of that the contents of a field. For instance if you are collecting in 'New Mexico' you might like to include the 'State' field, but set the default value and hide it from the user. This way the state is automatically recorded without the ODK user even having to confirm.

The ODK system has the ability to record a variety of data types. Depending on the data type a different interface is shown to the data collection user. For basic text fields, the user is shown the standard on-screen keyboard, similarly for numeric fields. ODK has the option of using the integrated GPS receiver, in which case the user has

the option of pressing a 'record location' button. It can also take photographs, video and sound recordings which can be associated as 'files' in the Tellervo database.

The final data type is the choice data type. In this case a user is displayed a list of options from which to choose. This is used to restrict the user to a predefined dictionary of terms (e.g. species, location type, object type etc). In the form builder, choice fields like this are prepopulated with the terms used by Tellervo. In the case of species, for instance, the list can be very long. Selected the correct item from a long list can be slow and tedious. The form builder tool therefore allows you to limit the list to a subset of terms. For example, you may know before your field visit that you will collecting one of just a handful of species of trees. Shortening the list to the likely options makes the data entry much more efficient.

All the predefined field types in the form designer correspond to the fields defined in the Tree Ring Data Standard. If you want to collect additional metadata not covered by TRiDaS then there is the ability to add user defined fields using the blue 'add user defined field' icon on the toolbar. Note, however, that at this time these fields are ignored by Tellervo when you import your metadata into the Tellervo database. To read data from these fields you should use the 'Create CSV File' option when downloading the data using the Bulk Data Entry ODK wizard (see section 6.4). Alternatively you can use other ODK compatible software such as ODK Briefcase or Kobotoolbox to extract to CSV, or Excel files.

Once you have finished defining your form you need to save it either to the Tellervo server—if you are planning on using the standard wireless data transfer method—or to a file. You can upload the form definition to the server by pressing the 'Upload Form' button or alternatively if your server is not accessible over the internet you can save the form definition to an XML file on your computer by going to *File ➜ Save as ODK XML file*.

If you would like to save your form definition to reuse and edit later you should use the *File ➜ Save form design* option. Note you cannot load an ODK XML form definition file within the Tellervo form designer, only the Tellervo-specific '*.odkform form design' file.

6.3 ODK mobile application

Once you have generated your ODK form definition you are ready to transfer it to your mobile device. Install the ODK Collect App from the Google Play Store as you would any other application.

If your Tellervo server is accessible remotely then you need to configure it by going to the 'general settings' page. From here click 'configure platform settings'. The URL of your server is the same as the URL you enter when start Tellervo Desktop but with the addition of '/odk/' at the end (e.g. http://my.ip.address/tellervo/odk/). Your username is your normal Tellervo username and your password is the ODK specific password that you set in the *Adminstration ➜ Users and groups* page of Tellervo Desktop. The authentication schema that ODK Collect uses is not as secure as Tellervo. Using the same password for ODK and Tellervo would require downgrading the security on Tellervo, something we are not willing to do. Instead a second less secure password is assigned for use with ODK Collect. Note that this ODK password only provides limited access to load and save ODK form data to the server, it doesn't provide access to the Tellervo database. Once you have configured your server, you can click the 'Get Blank Form' button on the main menu of ODK Collect. This will list all the ODK forms you have defined in Tellervo and allow you to transfer them to your mobile device.

If your Tellervo server is not accessible over the Internet, you will need to transfer the ODK form definitions to your device manually. Whichever method
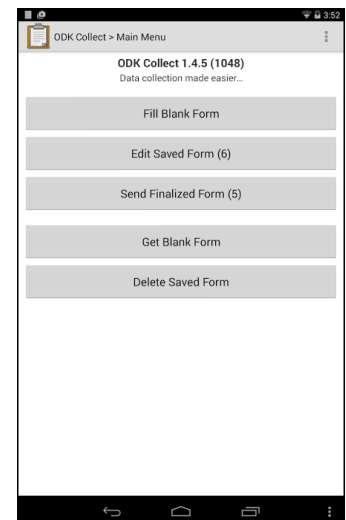
Figure 6.2: Screenshot of the main screen of the ODK Collect mobile application running on an Android mobile phone.

you use, the ODK XML form definition file needs to end up in the `odk/forms` folder of your device.

With your ODK form definition now installed on your mobile device you can press the 'Fill Blank Form' button on the ODK Collect main menu and you should see your form in the list. You can then create a form instance for each record that you'd like to create. The metadata for each instance is stored as a separate XML file on your phone in the `odk/instance` folder, along with any additional photo, video, or sound files that you created.

6.4   Importing ODK metadata into Tellervo

Once you have returned from the field, you can import your ODK metadata into Tellervo via the Bulk Data Entry interface (see section 5.3). On the Bulk Data Entry toolbar there is a button that opens up the ODK import wizard. The wizard has a number of steps each with descriptions to walk you through the process regardless of whether you are accessing your ODK data over the network or from a local folder. If you are doing the latter, you will need to copy the ODK form instances from your device to a local folder on your computer. The import wizard will then access the files from there.

Once you have completed the import wizard, the Bulk Data Entry screen will be populated with all the metadata you entered in the field. From here you can check, fix and augment the metadata before finally pressing the 'import selected' button on each of the object, element and sample pages. Once your metadata is safely imported into the Tellervo database you can then generate barcode labels for your samples using the button on the toolbar of the samples page.

6.5   Deleting ODK data

Both the ODK form definitions and form instances are by default left on the server. If you decide you no longer want a ODK form definition any more then you can go to *File → Delete form designs from server* from the main Tellervo window. Similarly you can delete form instances by going to *File → Delete form data from server*. Note there is no undelete option so you need to be certain!

Both form definitions and instances are also stored on your mobile device. To delete these too you can go to 'Delete Save Form' from the main menu of ODK Collect. From here you can delete some or all of the forms definitions and/or instances.

Chapter 7

Mapping

Tellervo includes an integrated open source 3D mapping system (based on NASA's award winning World Wind Java SDK) similar to the program Google Earth which you're no doubt familiar with. As mentioned in the installation chapter, this mapping system requires an OpenGL 3D capable graphics card. Before you can use the mapping in Tellervo, you must also have something to map! See the chapter on Metadata (page 27) for information about adding coordinates to your system.

There are two ways to map data from your database. First of all, you can see a map of all the sites (i.e. TRiDaS objects) by going to *Administration ➜ Site map*. This will give you a screen like this:

Figure 7.1: Screenshot showing an example of a site map.

You can also see a map of your current series if you have latitude/longitude metadata by clicking on the map tab on the main data screen. The map tab is a more recent implementation and contains a number of more advanced

features that the seperate *Administration* ➜ *Site map* does not. We recommend you use the map tab feature unless you prefer a full screen map.

7.1   Navigation

Figure 7.2: On-screen navigation controls.

You can navigate around your maps using the on screen controls (figure 7.2), by using your mouse and/or your keyboard. These controls enable you to explore your location information in 3D such as the example of Mount Vesuvius in figure 7.3.

7.1.1   Mouse with scroll wheel

Pan Left mouse button click and drag – all directions
Zoom Use the scroll wheel on the mouse or Left and Right mouse (both buttons) click and drag up and down
Tilt Right mouse button click and drag – up and down or use 'Page Up' and 'Page Down' on the keyboard.
Rotate Right mouse button click and drag – left and right Note: Crossing the top and bottom half of the screen
       while rotating will change direction.
Stop Spacebar
Reset Heading N
Reset all R

7.1.2   Single button mouse

Pan Left mouse button click and drag - all directions. L left mouse button click once to center view.
Zoom Hold 'Ctrl' on the keyboard and Left mouse button click and drag - up and down
Tilt Hold 'Shift' on the keyboard and Left mouse button click and drag - up and down or use "Page Up" and
       "Page Down" on the keyboard.
Rotate Hold 'Shift' on the keyboard and Left mouse button click and drag - left and right
Stop Spacebar
Reset Heading N
Reset all R

7.2   Interacting with data

Each marker on the map represents either a TRiDaS object or element in your Tellervo database. By clicking on these pins you can get more information from the database (see figure 7.3).
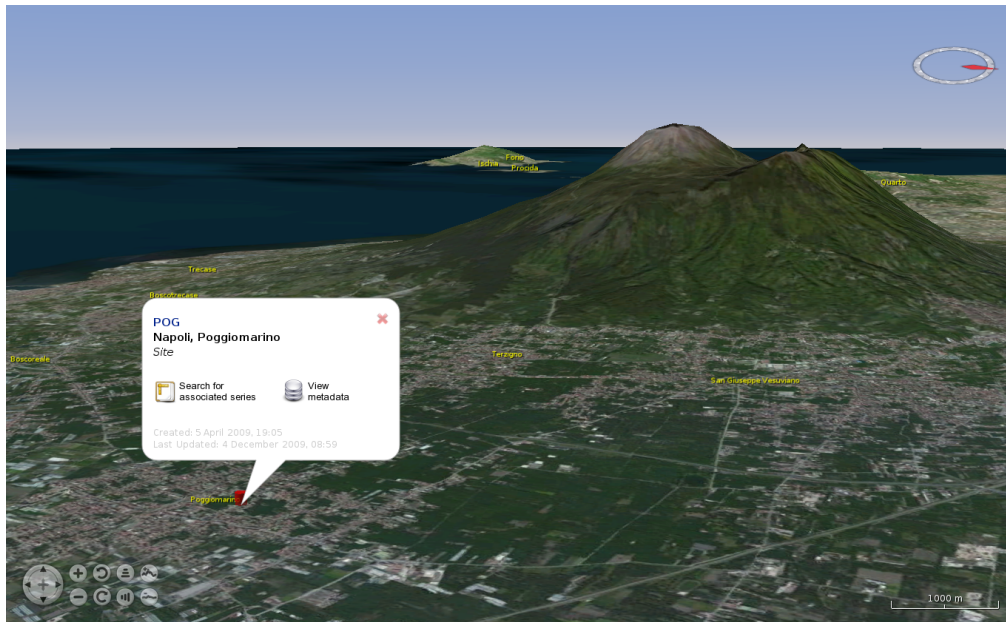
Figure 7.3: Screenshot of a map with information pin expanded

The example above shows the marker is of a site in Napoli called Poggiomarino (code name POG). You can see the option for searching for all series in the database associated with this site, and also the option for viewing all the metadata.

7.3 Map layers

Tellervo comes ready configured with basic map layers, including high resolution satellite imagery and basic political features. You can turn background layers on and off by going to *Map ➜ Add layers* or using the layer panel at the side of the map.

Map layers are downloaded on-the-fly so there is likely to be a delay when you initially visit a new region. However, up to 2Gb of map data can be cache locally to your hard disk, so on future visits, maps should load quickly.

7.3.1 Data layers

Data map layers (i.e. site and sample locations) are controlled with the layer list. When viewing series, you will have the option of adding layers containing points for all the other series at the current site, and showing all the sites in the database.

You can use the 'Add layer' button to add data layers of the following types:

All Tellervo objects – this adds a single layer containing all the objects within the Tellervo database.

Tellervo entity from database – this adds a layer containing the location of one record from the Tellervo database. This is specified by labcode e.g. ABC would add a pin for the site ABC, whereas ABC-1 would add a pin for the element ABC-1.

Elements from an object – this adds a layer containing all the elements for a specified object. The object is specified by labcode.

All ITRDB sites – this downloads the location of all sites currently available in the ITRDB database and adds them as a single layer.

ESRI Shapefile – this enables you to load an ESRI shapefile stored locally on your computer. Tellervo supports polygon, polyline and point files, although currently it does not enable you to style this data. Data for a layer is presented using a random color.

Google Earth KML/KMZ file – like the ESRI shapefile option this enables you to load spatial data from your computer.

7.3.2   Web Map Service (WMS)

The mapping system in Tellervo includes support for remote map servers that use the OGC Web Mapping Service (WMS) standard. If you go to *Map ➜ Add layers ➜ WMS layers*, you will get a dialog with a tab for each WMS server configured for your system. By default this includes the NASA Earth Observation and Jet Propulsion Lab servers. By ticking layers in this list you can add data layers to your map.

You can add map data from other WMS servers by clicking the '+' tab and entering the URL of the server you would like to use. This will give an additional tab with all the available map layers. This server will only be available for the duration of your current session so will need to be added each time you start Tellervo. If you would like a particular WMS server to be made permanently available, your Tellervo administrator can do this (see 'Managing map services', on page 76 for further details). Additional WMS servers added in this way will be available to all users the next time they connect to your Tellervo server.

Your system administrator may host a map server specifically for your lab, for instance, containing high resolution plans of an archaeological site that you are working on, or environmental data for your study region. Figure 7.4 shows an example overlay of sea surfaces temperatures loaded dynamically from the NASA EO server.
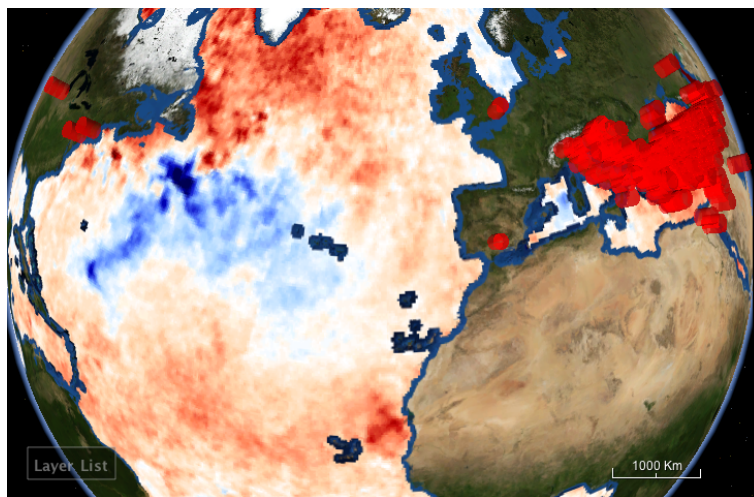
Figure 7.4: Map screenshot with a NASA sea surface temperature overlay dynamically loaded from the NASA WMS server.

7.4   Toobar buttons

 – The spatial search button enables you to draw a rectangular bounding box on the map surface to search for sites and trees. Once you draw your bounding box Tellervo will open a database browser window with

the bounding coordinates entered into the search tab. You can add additional parameters to further refine your search.

– Hide/show the north arrow / compass widget

– Hide/show the map navigation control widget

– Hide/show the inset overview map

– Hide/show the UTM map graticule

– Hide/show the MGRS map graticule

– Hide/show place names

– Hide/show political boundaries

– Turn the anaglyphic 3D mode on/off. Requires red/cyan glasses.

– Saves the current map as an image. See section 7.6.

– Add an ESRI Shapefile layer to the map

– Add a Google Earth KML layer to the map

– Add a GIS image layer to the map

– Add a Web Map Service (WMS) layer to the map

– Add a Tellervo database layer to the map. Adds objects or elements from the database via user defined parameters.

7.5   Layers list

Right clicking in the layers list (specifically over the Tellervo data layers rather than the background layers) provides a popup menu with a number of additional options. Layers can be removed and renamed from here. They can also be exported to Google Earth KML files (for use in Google Earth or other GIS applications). It's also possible to change the style of the map markers here through the 'Layer properties' dialog. This enables the user to change the colour, shape, size and transparency of the map markers.

7.6   Exporting maps

You can export maps by going to *Map ➜ Save current map as image*. For best results, maximize your map window first. You may also like to turn off various map widgets by going to the View menu. The exported image will include everything you can see on your map screen.

As well as exporting maps as images, it's also possible to export map data as Google Earth KML files. Simply right click on the layer in the layer list and select 'Export to KML'. Note this only works of Tellervo data layers, not background or other imported GIS layers.

Graphing

The graphing component is reused in many places throughout the Tellervo desktop application. The following description although based on the main graphing screen in Tellervo is largely applicable to all dialogs that include graphs (e.g. crossdating, indexing and reconciliation).

The main method for graphing your tree-ring data is by choosing an option from the Graph menu. Depending on the type of series you have open, the options available to you will be different. For raw measurement series, you will just have the option to 'Graph active series'. This will give you a simple graph of the current series that you have open. If you have a derived series open, then you may also choose 'Graph component series' which will plot all the series that go to create this series, or 'Graph all series' which graphs all the component series as well as the current series.

8.1 Controlling graphs

When newly created graphs are plotted according to the scale on the axes. A feature of Tellervo graphs though is that they can be manipulated directly on the screen. Both dendrochronology was computerized, dendrochronologists would plot rings manually on to graph paper. These paper graphs were then placed on lightboxes and moved around to enable comparisons. The graph function in Tellervo emulates this behaviour allowing users to click and drag graphs around to test for visual matches.

Figure 8.1 shows an example graph dialog. The mouse is hovering of the blue measurement series at relative year 1040 illustrating Tellervo's highlighting and guide line capabilities. A feature not shown in this screenshot is the
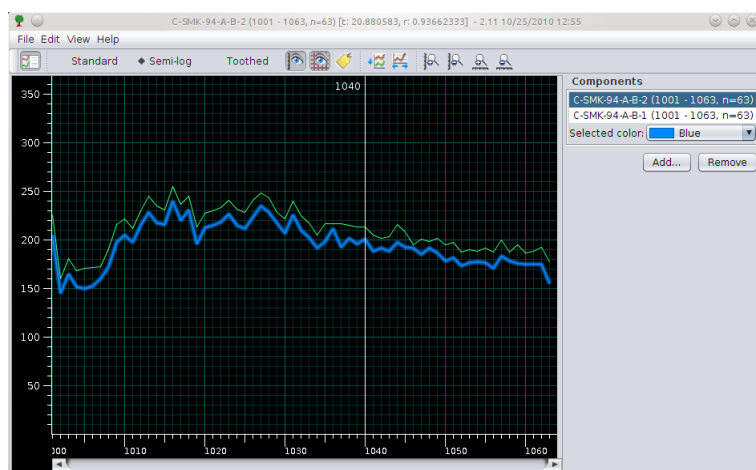
Figure 8.1: An example graph window containing two undated series of the same sample on a semi-log graph. Note the legend is visible with the options for adding or removing series.

illustration of sapwood rings. When sapwood rings are present the corresponding years on the chart are denoted via a heavier line.

The layout of graphs can be changed using both the toolbar buttons and menu options. The type of graph can be changed between a standard line graph, a semi-log graph and a toothed graph using the radio buttons. The remaining buttons are as follows:

	Zoom in on the horizontal axis
	Zoom out on the horizontal axis
	Zoom in on the vertical axis
	Zoom out on the vertical axis
	Toggle show/hide the grid lines
	Toggle show/hide the series labels
	Toggle show/hide the vertical axis
	Spread the series evening up the vertical axis
	Set the baselines of all the series to zero
	Resize graph to fit horizontally
	Toggle show/hide the legend

There are also a number of keyboard shortcuts that you might find useful:

Tab : Cycles through each graph component
Ctrl+W : Increase vertical scale
Ctrl+S : Decrease vertical scale
Ctrl+A : Increase horizontal scale
Ctrl+D : Decrease horizontal scale
Up arrow : Moves selected graph up by 10 units
Down arrow : Moves selected graph down by 10 units
+ : Moves selected graph up by 1 unit
- : Moves selected graph down by 1 unit
HOME : Scroll to first year of series
END : Scroll to last year of series
PAGE UP : Scroll left by one page width
PAGE DOWN : Scroll right by one page width
SPACE : Sets horizontal origin of all graphs to the same value

8.2   Exporting graphs

To export your graphs for use in reports you can go to *File ➜ Export plot as PDF file*, or *File ➜ Export plot as PNG file*. This presents you with a dialog for setting the colors, labels and size of the exported image. This functionality is due for an overhaul in the future to provide more flexible support for publication quality graphics.

Importing and exporting

Importing and exporting of dendro data in Tellervo is provided through the TRiCYCLE libraries. TRiCYCLE is a universal dendro data conversion application for converting back and forth between 24 supported data formats (Brewer et al., 2011). The open source libraries that provide the functionality to TRiCYCLE are incorporated directly into Tellervo providing support for all these formats.

Belfast Apple	Nottingham
Belfast Archive	ODF Spreadsheet
Besancon (including SYLPHE variants)	Oxford
CATRAS	PAST4
Cracow Binary Format	Sheffield D-Format (Dendro for Windows)
Comma delimited text files (CSV)	Topham
Corina Legacy	TRiDaS
DendroDB	TRIMS
Heidelberg (TSAP-Win)	Tucson (RWL and CRN)
KINSYS-KS	Tucson Compact
Microsoft Excel 97/2000/XP	VFormat
Microsoft Excel 2007	WinDENDRO

Table 9.1: List of the twenty-four formats supported by Tellervo. See appendices A–X (pages 123–195) for full descriptions.

9.1 Exporting data

Exporting data is initiated by the *File ➜ Export data menu*. If this is called from the main Tellervo data window, it will export the current series. If it is called from the Tellervo home screen, then it will present you with the database browser and allow you to pick one or more series to export. If you use the menu from within the main data editor then it will export

The export dialog contains two tabs. The first allows the user to choose the format that they would like to export to and the folder into which to save the result. Note that the user needs to specify a folder not a filename as many formats are unable to store more than one series in a file. When exporting derived series such as chronologies, the export dialog may therefore need to create multiple files. The second tab contains advanced options for altering the behaviour of the exporter:

What to export – This option enables the user to choose between exporting just the current series, or the current series and all associated series

Figure 9.1: Screen showing a series that has been exported to Besançon format. In the summary of the export at the bottom of the screen you can see the warning to the user that this format does not have the ability to represent relative dates properly.
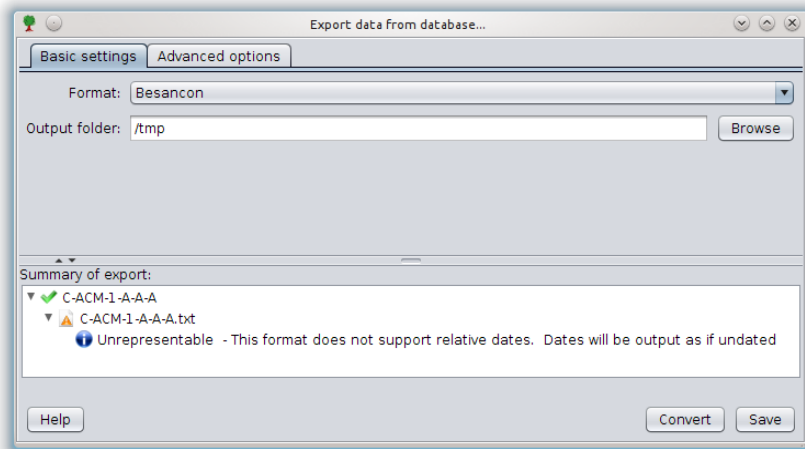
Grouping – This enables the user to choose to group files into a single export file if possible. For formats that do not support more than one series in a file, this option is ignored.

Naming – This configures how the output files are named. See section 9.1.1 for more details.

Encoding – This specified the character encoding to use in the exported text file. See section 9.1.2 for more information.

9.1.1   Naming conventions

The naming convention is used to determine how to name the output files. The naming convention relates to the filename itself and not the file extension. The file extension is specific to the output format chosen (e.g. Heidelberg files are .fh and TRiDaS files are .xml).

Numerical – This is the default naming convention. It uses the name of the input data file and appends an incrementing number if more than one output file is produced.

UUID – This gives all output files a random named based on Universally Unique Identifiers (UUIDs). This is a 36 character hexadecimal code which due to the astronomically large number of possible combinations is guaranteed to be universally unique. A typical filename will look like: 550e8400-e29b-41d4-a716-446655440000.

Hierarchical – This uses the hierarchical structure of the TRiDaS data model to provide a meaningful name for the output file. It joins together the title of each entity in the file beginning with the project name through to the series name. For files that contain multiple series, the name will contain details of all the entities shared by all the series in the file. For example, if a file contains several series from the same sample, then the file name will be projectTitle-objectTitle-elementTitle-sampleTitle. If the file contains several series from different samples of the same object, then the file would be projectTitle- objectTitle. If multiple output files end up with the same name then like the numerical convention described above, the files will have an incremental number appended to the end. Unfortunately, most input data files do not contain rich name information so files end up being called unnamedProject-unnamedObject-unnamedElement etc. This convention is therefore more appropriate when converting from TRiDaS to other formats.

Series code – This convention is only applicable to formats that contain just one series. The file is named according to the series code.

Series code (8 characters) – Same as 'Series code', however the file name is truncated to 8 characters if the series code is longer.

Keycode – Similar to 'Series code' but preferentially uses a keycode (supplied by some file formats) if available. If a keycode is not provided, then it falls back to using the series code.

Note that some formats (e.g. CATRAS) require the file name to be the same as a field within the file. In this case the naming convention is overidden, so no matter what convention you specify the filename will be the same. If you manually rename a CATRAS file you will come across errors when loading it in the CATRAS application.

9.1.2 Character sets

Character sets are the mechanism for pairing computer character codes with the character glyphs that we read. The widely used standard was originally ASCII, but this does not include diacritic characters, and characters specific to certain languages. There have since been many character encodings proposed (e.g ISO 8859-1 for Western Europe and ISO 8859-7 for Greece) as well as some that are specific to Windows and Mac operating systems (e.g. Windows-1252 and MacRoman). The character set that is becoming most widely used today is Unicode UTF-8. This is capable of representing the vast majority of characters (107,000+) while remaining backwards compatible for the 128 characters that ASCII is able to represent.

If an incorrect character encoding is used to interpret a file, normally the majority of characters will display correctly (where the character sets share the same encodings) but more unusual characters will be displayed incorrectly - typically square boxes or question marks.

The character encoding is set to the default for the operating system you are running. For instance on MacOSX this will be MacRoman and for Windows it will be Windows-1250. If you know your input file is in a different encoding you should set it in the input charset box. If your output file needs to be read on an operating system other than the one you are currently running, then you may like to override the writer charset. Please note that for certain writers, the character set used is part of the file specification (e.g. TRiDaS must be UTF-8). In this case your choice will be ignored.

The final complication with regards character sets is the line feed character(s). For historical reasons different operating systems use different characters to represent a new line. Depending on the software that is used to read a file, this can cause problems. Tellervo itself will automatically adapt to files with any type of line feed characters so reading files in Tellervo will never be a problem. When writing out files, Tellervo will use the default line feed for the operating system you are running, unless you choose a platform specific character set. For instance if you run Tellervo on Windows and choose a MacRoman writing charset, Tellervo will use Mac style line feeds.

9.2 Importing data

Although the TRiCYCLE import library can easily read the data from many data files, the process of importing files is complicated by the need to cross-map which sites, trees etc the data corresponds to and whether they are already in the database or not.

The data import process is accessed via the *File ➜ Import data* menu. Choose the file format that you would like to import, then select on or more files. If you are unsure what format your file is in, you can use appendices A–X (pages 123–195) to help you. You may also like to download TRiCYCLE* which includes a file identification tool in the help menu.

> ⚠ *The Tellervo system is currently limited to reading in raw data files. It does not allow for reading in of chronology files as these contain a more complex level of metadata. Support for chronology files will be included in a later release.*

---
*TRiCYCLE is available from `http://www.tridas.org/tricycle`
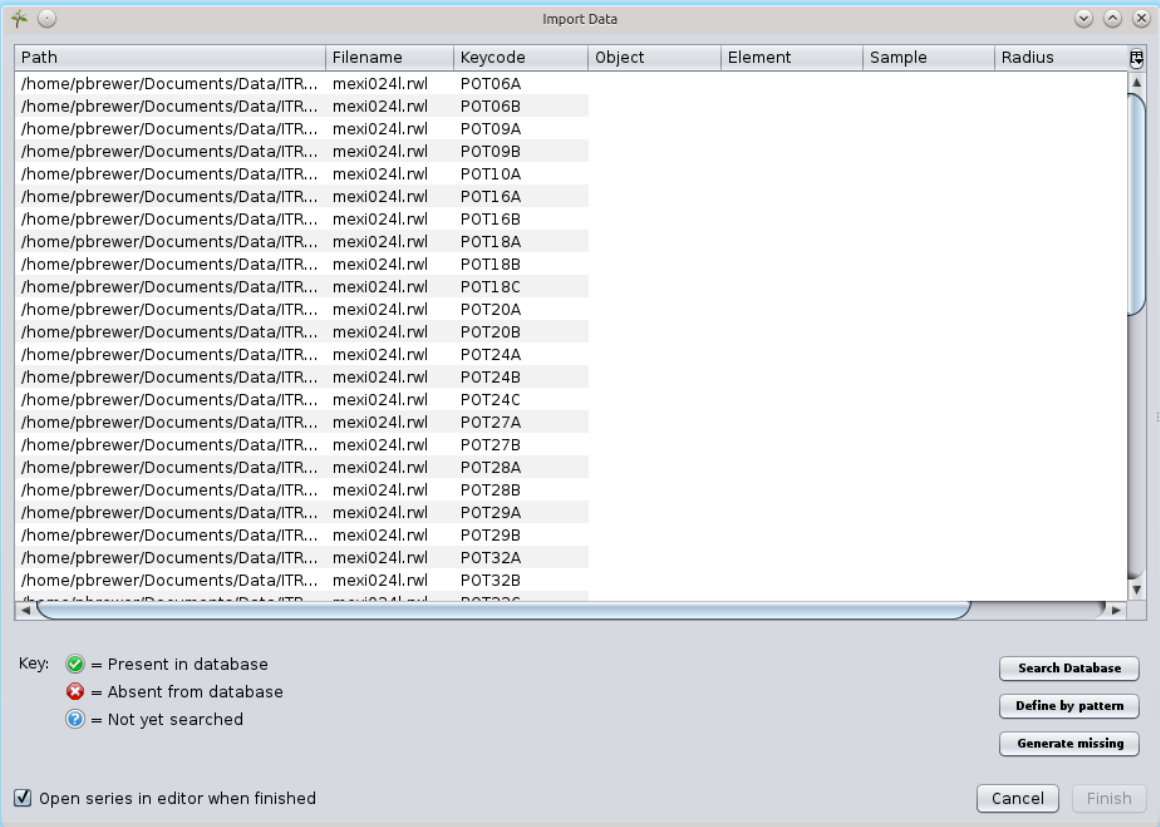
Figure 9.2: The data import dialog is used to assign data from legacy data files to the correct entries in the Tellervo database.

The Import Data dialog (figure 9.2) provides a spreadsheet-like interface to assign which object, element, sample and radius in the database each imported data series should be assigned to. It is possible to simply type the code for each of these entities into the table but for more than a few series this quickly becomes tedious. Instead, you can automatically populate this table by defining patterns based on the file names, folder names and/or rudimentary metadata from the files. If your data files follow a rigid naming convention this can dramatically speed up the import process. To use this feature click the 'define by pattern' button to show the define patterns dialog (figure 9.3).

Figure 9.3: Data patterns dialog used to define naming conventions to automate the mapping of legacy data to entries in the Tellervo database. The example here is for a series with a keycode of 'POT06A'. The first regular expression extracts all upper case alphabetical letters at the start of the keycode for the object code. The second regular expression extracts the digits for the element code. The fixed width pattern extracts the 6 character for the sample code.

For each of the object, element, sample, radius and series codes, you can define patterns for mapping your data to the database. These codes may be based in whole or in part on: file name; full folder path; final folder; and keycode extracted from the file. You can specify a variety of methods for extracting information from these fields:

All – This simply takes the entire field you specify. For example if the data file was called abc.rwl and you asked for 'all' of the filename field, then you would get abc.rwl

Fixed width – This allows you to specify a fixed number of characters specified either as an absolute number or a range. For example, if you were to specify 1-3 for the filename abc.rwl, then you would get abc. This is a simple method but relies on your data files all following the same convention and with the same number of characters.

Regex – This is a far more powerful method, but more complicated to master. Regex stands for regular expression and is an advanced language for defining patterns. A full description of the syntax is beyond the scope of this manual but there are many resources on the Internet to help.

Once you have set your method and pattern you can press the 'run tests' button to check your pattern matches what you expect. The test field will show the result of running the first series in your import through your pattern. If the results aren't as you expect you can modify your pattern and re-run the tests. The dialog also has the ability to load and save your pattern definitions. This is particularly helpful if you are lucky enough to have many data files that match a specific naming convention.

Once you are satisfied with your patterns you can click OK to return to the data import screen. The fields in the table will be populated with the matches as defined by your patterns with the entries marked by one of three icons. These denote whether a record with this code exists in the database, whether it is currently absent from the database, or whether the database has not yet been searched. Clicking the 'search database' will check the database for all unsearched records turning them from a blue question mark to a red cross or green tick accordingly.

Once you are satisfied that all entries in the table are correct you can go ahead and press the 'generate missing' button to create entries in the database for each record currently marked with a red cross, turning them green. Once the table is full of green ticks, the 'finish' button will become enabled. If the 'open series in editor when finished' tickbox is checked, then all these series will be opened in the main window. If your file does not explicity indicate what units the measurements are in, a dialog box will ask you to specify.

Please note that all the newly generated database entries will be skeleton records. You will need to check and update the metadata for each of these to ensure they fulfill the requirements of Tellervo. For example, many mandatory Tellervo fields are missing from legacy data files. Even fields that are present (such as taxon) are typically not standardised well enough to match the specific controlled vocabularies used in Tellervo.

Curation and Administration

10.1 Laboratory workflow

Tellervo includes a number of functions to assist you with the curation of your physical sample collection. To understand how these are designed to assist users, we must first consider the workflow within a laboratory.

In research laboratories, samples generally come to the lab in large batches following field collection. In this case the typical workflow may be as follows:

1. Collect samples and record field notes as accurately as possible
2. On returning to the lab enter field notes as soon as possible into the 'bulk data entry' interface
3. Print sample barcode labels
4. Prepare physical samples and label with barcodes
5. Assign samples to storage boxes
6. Measure samples, using barcodes to recall metadata from database
7. Crossdate samples / build chronologies
8. When all samples from a box are completed register box as archived and then store

For commercial labs offering dendrochronological dating as a service, samples more likely to arrive in smaller batches. In this case, the bulk data entry interface may not be the most efficient method for entering metadata. In this case the user may simply prefer to use the *File* ➔ *New* method for each sample.

Either way, the concept behind the curation of a collection in Tellervo revolves around the accurately recording as much metadata about a sample as possible, then labeling the physical sample with a label containing a barcode for Tellervo and sample code for the user. By entering a sample into the database as soon as it enters the lab, it can be traced throughout the workflow. When a chronology is built, it is easily to quickly and efficient locate all samples that have been used. By assigning samples to boxes, groups of similar samples (e.g. from the same site) can also be easily stored together and located quickly and efficiently.

10.2 Barcodes

Barcodes allow you to keep track of what samples you have and where they are stored. Although it is not essential to use the barcode functions, we strongly suggest you do because they save time and money, but most importantly they greatly reduce the scope for erroneous data entry. For instance, when measuring a sample a user simply scans its barcode and all the relevant metadata is retrieved from the database, rather than relying on them to enter data manually. Barcodes have been routinely used in the retail industry since the 1980s. They can be equally as useful in dendrochronology laboratories.

Tellervo creates and reads barcodes for samples, measurement series and boxes. Each barcode encodes the unique identification code stored in the Tellervo database for each of these entities. Due to Tellervo's use of universally unique identifiers (UUIDs), these codes are guaranteed to be unique opening the opportunity of labs to loan samples, much like libraries do with books. There are many styles (or 'symbologies') of barcodes in use today, but Tellervo uses one of the most common (Code 128) which is supported by the vast majority of barcode readers. For a detailed discussion on the specifications of the Tellervo barcode see section 19.4.

Basic barcode readers are now cheap and widely available, with basic devices retailing for a few tens of dollars. Most are characterized as 'keyboard interface devices' and work like an automated keyboard, typing in a string of characters when a label is scanned.

Within the Tellervo application, whenever the user is required to specify a box, sample or series, they have the option of typing the human readable lab code or scanning the barcode. By using the barcode, the user can be sure they are not entering typographic errors so we recommend using barcodes whenever possible.

The most important barcode is the label for the physical wood sample. These are easily generated through the *Administration* ➜ *Labels* ➜ *Sample labels* menu entry. Currently the layout of these labels is fixed, but in the future we aim to provide different styles.

10.2.1 Sample labels

Before labels can be generated, metadata entries the sample level must have been made in the database. This is typically done using the 'bulk data entry' interface (see page 31). If samples are already in the database, the user needs to select the object of interest in the label creation dialog to see all the available samples. It is then just a matter of selecting the samples of interest and moving them into the 'selected' column. Once the list is populated (samples from multiple objects can be included), then you can either click 'Preview' to see a PDF of the labels, or 'Print' to print directly.

 C-ACM-1-A

Figure 10.1: An example of a sample barcode produced by Tellervo for the Cornell lab. Note the label also includes the human readable code for the sample.

The current label style is designed to fit on standard core mounts and most samples. There are no widely available die-cut labels that fulfill this need, so the labels are intended to be printed on archival grade full page sheet labels (e.g. Avery® layout 6575), and then manually guillotined.

10.2.2 Box labels

The procedure for printing box labels is the same as for samples. Samples must have already been assigned to boxes before the label is printed (see section 10.3 for details). To print (or preview) box labels go to *Administration* ➜ *Labels* ➜ *Box labels*. The label style is designed to be printed on $5'' \times 8\frac{1}{8}''$ labels, two per sheet such as the Avery® 6579 layout. An example is shown in figure 10.2.

> ⓘ Until dynamic label styles have been implemented, box labels will print one per page. To make use of the second label on the page, the same sheet should be fed through the printer a second time.

10.2.3 Series barcodes

Series barcodes are printed at the top of a standard series report (see figure 10.3). These are produced through the *File* ➜ *Print*, or *File* ➜ *Print preview*, menus.

GR38

BOX a3b0761c-bfdf-11de-a207-e37bbc4501c3

Created: October 23, 2009 10:23 AM
Label updated: July 12, 2011 9:40 AM

Object	Elements	# Samples
KRR	1	1
KSR	1-12	14
KSY	1-14	14
KTM	1-13	13
KYP	1	1
	Grand Total	**43**

Comments:
No comments recorded

Figure 10.2: An example of a box label from the Cornell collection. The label provides a human readable name for the box (GR38), a barcode for accessing the box details within Tellervo, and a summary of the samples contained within the box.

C-YMT-1399-A-A-2

SERIES 191dd500-af8b-11df-9b25-1bfba0ced0da

Istanbul, Yenikapi Metro

Created: August 24, 2010 10:23 AM
Last Modified: August 24, 2010 10:23 AM

Measured by: Leann Canady
Supervised by: Charlotte Pearson

Ring widths:

1/100th mm	0	1	2	3	4	5	6	7	8	9
1001		• 132	79	55	160	111	172	198	177	130
1010	• 176	166	160	179	255	236	232	273	188	191
1020	• 182	109	113	92	73	75	69	84	98	121
1030	• 123	146	239	177	177	198	196	230	236	241
1040	• 208	161	161	206	246	253	173	189	164	185
1050	•• 149	112	143	145	108					

• = Single Pinned   •• = Double Pinned

Wood Completeness:

- Pith is incomplete.
- A total of 54 rings were measured.
- Heartwood is incomplete
- Sapwood is absent
- Bark is absent.

Interpretation:

- The first ring of this series begins in relative year 1001.
- The pith of this radius was laid down in exactly relative year 1001 and died after relative year 1055.

Element and sample details:

- Taxon: *Quercus*
- Element type: Post
- Sample type: Cross section

Figure 10.3: An example of a report showing barcode and basic metadata about a series.

10.3   Storage boxes

Tellervo uses the term 'box' to refer to the collection of samples you archive. Many labs (including Cornell) use cardboard bankers boxes to store samples once they are completed, but the same box concept could refer to draws or shelves in your collection.

10.3.1   Creating and editing boxes

Records for boxes in the system are created and edited through the *Administration ➜ Curation ➜ Box details* menu. To editing an existing box, you can scan the barcode label on the box, or select from the list. To create a new box, click the 'Create new box' button and enter its details. There is no restriction on what boxes should be called, but it is probably easiest if you use some sort of numerical sequence to assist with organizing the boxes in your store. The Cornell Tree-Ring Lab, for example, uses a two part name for each, the first being the year of collection, the second being a sequential number (e.g. 2009-11).

The contents tab lists all the samples that have been assigned to this box. To add new samples, simply click the 'Add sample to box' button and scan the sample's barcode.

10.3.2   Inventory

An important feature of any collection management system is the ability to perform an inventory on the collection. Even with the most robust system, samples will always go astray so its important to be able to periodically check that the boxes contain what you expect.

The 'Contents' tab of the Box details dialog contains a feature to assist with this. Next to the list of samples that are recorded as present, there is a temporary checklist column. By checking the boxes for each sample actually stored in the box it is easy to see which samples have been mislaid. If the 'Mark unchecked as missing from box' button is then pressed, the date and time the discrepancy was noted is then recorded in the comments field for the box.

10.3.3   Checking boxes in and out

Tellervo includes function for checking boxes in and out of a store, much like when a book is borrowed from a library. The *Administration ➜ Curation ➜ Check out box from store* and *Administration ➜ Curation ➜ Return box to store* menus do just this. You can either scan the box barcode or select the box from the drop down menu. These options record when a box is checked out/in and by whom. These details can be seen by users in the box details dialog.

10.3.4   Locating samples

As you might expect, Tellervo also includes a function for locating your physical samples. This is available in the *Administration ➜ Curation ➜ Find a sample* menu. There are three methods for locating a sample: via barcode; via lab code; and manually by object/element/sample.

If you have the sample in your hand and you simply want to know which box it should be returned to you can scan the barcode. If you are looking for a sample and you know its lab code then you can enter this instead. Alternatively, you can use the drop down menus to search for one or more samples at once. For instance, you can locate all the samples for a particular object and element.

Chapter 11

Indexing

Trees tend to put on big rings when they're young, and smaller rings when they get older. Some trees put on very large rings, while others put on very small rings. These variations in growth can make it difficult to crossdate samples. Some dendrochronologists therefore prefer to index or normalize their ring width data before combining into chronologies.

Indexing is a manipulation you can perform on your data to make it easier to crossdate.

The procedure for indexing is as follows:

1. You open a series (raw data)
2. You ask Tellervo to index it
3. Tellervo shows you some possible curves
4. You pick a curve (based on its graph, statistical scores, and your expectation of how the tree is growing)
5. Tellervo converts each year's ring width to a ratio of actual growth to expected growth for that year
6. You save the series (indexed data)

Indexing changes the units of a dataset. A raw sample has units of hundredths of a millimeter (0.01 mm) or microns. An indexed sample has units of parts per thousand (0.1%, or ‰).

This doesn't cause a problem with crossdating. The t-score normalizes all samples as part of its test, and the trend only cares if the values are increasing or decreasing. For more information on crossdating and chronology building, see chapter 12. It does, however, cause a problem with 'summing' since summing needs to take the average (what's the average of 1mm and 75%?). Therefore, the samples in a sum must be either all raw, or all indexed.

11.1 Types of index

There are a total of six different indexing methods available in Tellervo:

11.1.1 Exponential Index

This is the most commonly used index as it matches the way trees typically grow. Quickly when young and then gradually slower. An exponential index is therefore by far the most common index you'll use as 9 times out of 10 this will be the best choice.

This index tries to fit an equation of the following form to your data, searching for the best values of a, b and p.

▶ $y = a + be - px$

> ⓘ This is sometimes called a negative exponential index, because the exponent is negative. Tellervo doesn't require that the exponent is negative, but if it's not, using this index probably isn't such a good idea; it means the tree is generally getting bigger, not smaller.

The least-squares algorithm used comes from Cormen et al. (2001); the matrix solving function comes from Van Loan (1999).

Sometimes the exponential index does a lousy job. If a tree is living in a crowded area and the trees around it get cut down, suddenly it has much better growing conditions, so it might grow faster as it gets older, instead of slower. If you tried to use an exponential curve on a tree like this, it would exaggerate this growth, and useful data would get flattened out.

The result is you're looking at the growing conditions of this one tree, so it's not going to crossdate as well.

Alternatively, imagine you are working on a tree with a fire scar that has a few very large rings. An exponential index wouldn't take much notice of this, because most of the sample is still shaped like an exponential curve, but when you applied it they would be grossly out of proportion. For these types of samples, there are other indexing algorithms available.

11.1.2   Polynomial Index

When you ask Tellervo to perform a Polynomical Index it tries to fit a polynomial curve to your data using the following equation:

▶ $y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_{1x} + a_0$

You decide what degree polynomial, n, to use and Tellervo automatically finds the best values of a_0, $a_1 \ldots a_n$, to fit your data.

11.1.3   Horizontal Line Index

This only changes the magnitude not shape of the curve and is used when you would link to combine raw and indexed data together. It is a special case of polynomial where the horizontal line is equal to the average value.

▶ $y = x_{avg}$

This index is not used for crossdataing because dividing each value by the same value doesn't change the shape of the curve, only its magnitude. A horizontal line index is, however, useful because every element in a sum must use the same units, either raw or indexed. Therefore if you want to include a raw sample with an indexed sample then a horizontal line index can be used to convert the raw sample without otherwise altering the shape of the curve.

11.1.4   Floating Index

This is a running average of the 11 surrounding years. The adaptive index is generally used as a 'last resort' when both exponential and a high-degree polynomial have failed. It is simply the average of the eleven surrounding years:

▶ $ind_i = 1/11(data - i - 5 + data_{i-4} + \ldots + data_{i+4} + data_{i+5})$

This index was originally called floating average, probably in reference to the fact that the index curve "floats" around, not following any explicit $y = f(x)$-type formula. But people tended to call it floating, and then floating-point, which means something very different. You might still hear people calling this index by these other names.

11.1.5 High-Pass Filter Index

The high-pass index is a more general case of the adaptive index. Instead of simply taking the average of 11 values, it takes a weighted average. It's an example of a "high-pass" filter because high-frequency signals can pass through, but low-frequency signals are filtered out.

The default is "1-2-4-2-1", meaning:

▶ $ind_i = 1/10(data_{i-2} + 2 \cdot data_{i-1} + 4 \cdot data_i + 2 \cdot data_{i+1} + data_{i+2})$

This comes from Cook and Peters (1981) who used it as a discrete filter before moving to a cubic spline. Note that almost half $(4/10)$ of the computed value is simply its old value. The high-pass index is nearly the same as the input, so the χ^2 values are usually the lowest, therefore do not choose this index solely on a low χ^2 value.

11.1.6 Cubic Spline Index

Cubic splines are a very specific type of high-pass filter. A cubic spline curve is created by combining a collection of cubic (3rd degree polynomial) functions.

There are many methods for constructing cubic splines through a dataset. The algorithm used by Tellervo has a parameter, s, which controls how tightly the spline fits the data. A lower value fits the data more tightly, a higher value fits the data more loosely. Therefore, s=0 fits the data exactly while s=1 is a simple line. A good starting point for dendro data seems to be around $s = 1x1016$.

Cubic splines were first used for dendro by Cook and Peters (1981) using an algorithm from Reinsch (1967).

You can change the s-value used for the subic spline in the preferences. You might use a cubic spline in the same cases you would use a high-pass filter e.g. when the sample doesn't generally follow an exponential or polynomial curve very well, perhaps due to a fire scar.

11.2 Indexing data

To index your data, first you need to open the series you would like to index. Next choose *Tools* ➜ *Index* to display the indexing dialog (figure 11.1).
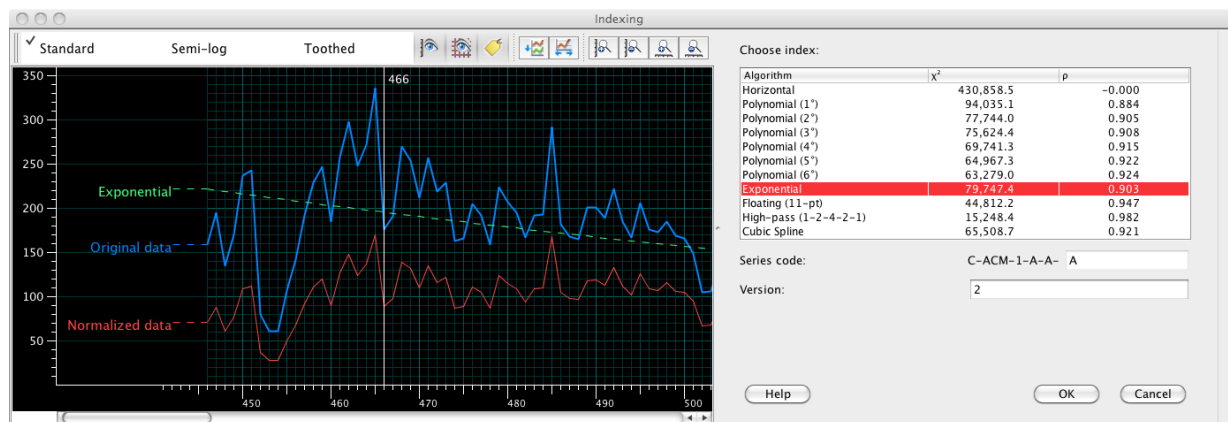
Figure 11.1: Indexing dialog showing the original data in blue, the exponential index of this data in green, and the normalized data in red.

From the indexing dialog you can then choose which type of index to apply to your data. The table on the right shows the available options along with the χ^2 and p values to help you choose the correct index to use. The graph shows your original data, the index line and the result of applying the index to the data and changes dynamically as you pick between different indexing methods. Once you have decided which index you want to use, select it, and click OK ensuring that you have given your data series a new version number.

Chapter 12

Crossdating and chronology building

All algorithms work in pretty much the same way. There's a "fixed" sample, and there's a "moving" sample. Imagine you have printouts of their graphs on translucent paper. The fixed graph is taped to a table, and you can slide the moving sample left and right. This is actually how it was originally done, on graph paper, with one inch per decade. Start with the moving sample to the left of the fixed sample, overlapping it by 10 years. Look at how well the graphs match: this is the first score that's computed. Slide the moving sample to the right one year and so on until you reach the end.

You could do it all simply by moving graphs and eyeballing the crossdates like this but there are hundreds of sites and millennia of chronologies you'll want to crossdate your samples against, so that would take a while. Tellervo has a few algorithms to find likely crossdates almost instantaneously. They aren't perfect, though, and all crossdates should be inspected visually to ensure they are a good fit.

12.1 Algorithms

Tellervo includes a total of five different algorithms for crossdating:

12.1.1 T-Score

The t-score is the classic crossdate. Unfortunately, every dendro program seems to have a slightly different implementation of t-score, so the numbers you get from Tellervo might not be exactly comparable to the numbers from other programs.

The version Tellervo uses is based on the algorithms given in Baillie and Pilcher (1973), though with some apparent bugs corrected (Ken Harris pers. comm.). In the following equations, $x_0, x_1, x_2, \ldots$ are the data of the fixed sample in the overlap, $y_0, y_1, y_2, \ldots$ are the data of the moving sample in the overlap, and N is the length of the overlap.

The first step is to make each dataset bivariate normal by replacing each value with the mean of the values around it, and then taking its natural logarithm. The preparation for the t-score is therefore done as follows and is done to both the fixed and moving series:

▶ $x_i \leftarrow \frac{x_{i-2}+x_i+x_{i+1}+x_{i+2}}{5}$

▶ $x_i \leftarrow ln(x_i)$

The student's T computation is then done as follows:

▶ $s_{xy} = \Sigma x_i y_i - N(x_i - x_{avg})(y_i - y_{avg})$

▶ $s_{xx} = \Sigma x_i^2 - N(x_i - x_{avg})^2$

- $s_{yy} = \Sigma y_i^2 - N(y_i - y_{avg})^2$
- $r = \frac{s_{xy}}{\sqrt{(s_{xx}s_{yy})}}$
- $t = r\sqrt{\frac{N-2}{1-r^2}}$

The *t*-score is an explorative statistic. There is no univerally accepted threshold above which a *t*-score is regarded as significant, however, Baillie and Pilcher (1973) suggest a value of 3.5. For more information see Wigley et al. (1987).

12.1.2 Trend

Trend is another popular crossdate statistic. It computes the percentage of years with the same trend (going-up- or going-down-ness). Scores greater than 60%-70% are good. Trend is also referred to as ufigkeitsko-Gleichläeffizient, Gleichläufigkeit and Eckstein's W.

The trend is the simplest crossdate. For each sample, it computes the trend of each 2-year interval (1001-1002, 1002-1003, and so on). The trend of a 2-year interval is simply whether the next ring is larger, smaller, or the same. The trend score is the percentage of intervals in the overlap which are the same. For example, a 75% trend (a very good score, by the way) means that for 75% of the intervals in the overlap, both samples went up in the same years and down in the same years.

If one sample stays the same, and the other increases or decreases, Tellervo considers that to be halfway between a same-trend and different-trend, and gives it half a point. Trend is a "non-parametric" algorithm, because it only takes into account if a given ring is bigger or smaller than the previous one, not by how much. To the trend, a drop of "100 1" looks exactly the same as a drop of "100 99". Two completely random samples will have a trend of 50%, on average. So you'd expect a trend must be greater than 50% to be significant.

According to Huber and Fletcher (1970), a trend is significant if:

1. $tr > 50\% + \frac{50}{\sqrt{N}}$ – For example a pair of samples with a 50-year overlap needs a $50 + 50\sqrt{50} = 57.1\%$ trend to be significant, but at a 400-year overlap need only a $50 + 50\sqrt{400} = 52.5\%$ trend. In practice, however, this doesn't tend to work terribly well. Using this scheme, there are typically about three times as many "significant" trend scores as *t*-scores, and users want this narrowed down a bit more. So take $\sigma = 3$ and use:

2. $tr > 50\% + \frac{50\sigma}{\sqrt{N}}$ – This gives about the same number of significant trend scores as *t*-scores.

Trends are also used in reconciliation. After they've been reconciled, both readings of a sample should have 100% trend.

12.1.3 Weiserjahre

The Weiserjahre algorithm is used for crossdating summed samples (chronologies) against single samples. All of the algorithms that have been mentioned so far only compare the ring widths. This works fine for raw samples, but when crossdating summed samples, there's a lot more information available, namely, the Weiserjahre data. Wouldn't it make sense to count a [20] 19×1 ring more heavily than a [1] $1 \div 0$ ring? 19 out of 20 samples think it's an increasing year, not just 1.

This is what the Weiserjahre cross does: for each possible overlap, it starts by counting the number of significant intervals of the master for that overlap. A significant interval is one with at least 3 samples, where at least 75% of them have the same trend. Then it computes the percent agreement (like the trend) between the master and the raw sample for only those significant years of the overlap. Of course, for the trend of the master, it doesn't

use the trend of the master; it uses the trend of the majority of its elements. They're usually the same, but not necessarily.

Another way to think about the Weiserjahre crossdate is: it's like a trend, but ignoring years where the sum has only 1 or 2 samples, or where there isn't an overwhelming trend in the sum. Also like the trend, the results are given as a percentage.

12.1.4   R-Value

The R-value, or correlation coefficient, is a crossdate which you'll almost never use. It's not terribly useful to dendrochronologists, but statisticians might want to know its value, so Tellervo makes it available.

The R-value is used in the T-Score, the T-score being defined in terms of the r-value and the overlap, N. If you look at the equations for calculating a T-Score you will see on the penultimate line:

▶ $r = \frac{s_{xy}}{\sqrt{(s_{xx}s_{yy})}}$

An r-value can range from 0.0 (no correlation) to 1.0 (perfect correlation).

12.2   Crossdating series

12.3   Managing chronologies

The Tellervo server

For basic day-to-day running of the Tellervo server, you simply need to make sure that the server is running. All other interaction and managment (creating users, granting permissions, accessing data) is done through the Tellervo desktop application. This section, however, outlines a number of aspects of the server that advanced users may find useful.

13.1 Backing up and restoring your database

As with any computer system it is important for you to back regular backups of your data to guard against hardware (as well as human!) errors. The two main methods for doing this are outlined below:

13.1.1 Backup whole Virtual Appliance

The simplest method is to make a copy of your entire Virtual Appliance, but this does have a number of drawbacks. The first is that you need to shut down your server before you can make the backup so this is only possible if server 'downtime' is not a problem for your lab. The second drawback is that it makes a copy of your entire server including the whole operating system, therefore each backup takes a lot more space.

1. Open VirtualBox
2. If you server is running you will need to do a full shutdown. From the server console type `sudo halt` then once it has halted you can close the console window and select 'Power off the machine'.
3. Select your virtual machine in the list on the left and go to *File ➔ Export Appliance*.
4. Follow the wizard, specifying a file where you'd like to back the server up to. Keep in mind that this will contain a complete copy of the server (including operating system) so could be 1Gb or more.

13.1.2 Restoring a Virtual Appliance backup

If you have followed the instructions in section 13.1.1 to backup your Virtual Appliance the steps to restoring your server are very similar to how your initially installed it. Simply open VirtualBox, then go to *File ➔ Import Appliance* and select the backup file that you made. Follow the wizard and it should restore your server. You can restore onto the same computer that was originally running the virtual machine (remember to give it a new name though if this is the case) or alternatively to any other computer with VirtualBox installed. This method can therefore be used to share entire databases.

13.1.3   Backup PostgreSQL database

The more standard way of backing up your database is to do a dump of the PostgreSQL database itself into a large text file. You can do this using the following command:

```
sudo tellervo-server --backup-db /folder/and/file/to/make/mybackup.dump
```

For example the following line will backup the database into a file called backup.dump in the tmp folder. Keep in mind that the tmp folder is cleaned each time the server is booted.

```
sudo tellervo-server --backup-db /tmp/mybackup.dump
```

You will want to transfer this backup to a remote location. Methods for doing this are given in section 13.2.

13.1.4   Restoring a PostgreSQL database

To restore an existing database dump file use the following command:

```
sudo tellervo-server --restore-db /tmp/mybackup.dump
```

The restore will confirm that you want to delete the current database and replace it with the backup. The restore procedure will create a fresh database, restore the database tables, restore your data and apply any outstanding database upgrades.

13.1.5   Backing up media files

If you have used the ODK field data entry system described in chapter 6, and requested to include media files then you will also need to make provision for backing these up. For reasons of efficiency these media files are not included in the PostGreSQL database. Links to these files are stored in the database, while the files themselves are stored in a web accessible folder.

13.1.6   Automated backups

If your server is left on, it's a good idea to schedule automated backups to ensure backups are made on a regular basis. Backup scripts can be written to use various backup rotations so that you have multiple versions of your backup over the last days, weeks or months allowing you to roll back to specific points in time. A good example would be to have nightly backups named by day of the week, as well as monthly backups taken on the first of the month. This will give you 19 backup files that will enable you to restore your database to the state it was in at any night of the last week or any month of the last year. An example of a simple nightly backup script would be:

```
day=`date +%A`
```

```
tellervo-server --backup-db /home/tellervo/Dropbox/$day.dump
```

Similarly a monthly backup script would look like:

```
month=`date +%B`
```

```
tellervo-server --backup-db /homne/tellervo/Dropbox/$month.dump
```

Save these scripts to files (e.g. at /home/tellervo/nightlybackupscript and monthlybackupscript) and then instruct the cron daemon to run them by editing the crontab using:

```
✎     sudo crontab -e
```

adding lines such as:

```
✎     @daily /home/tellervo/nightlybackupscript
```

```
✎     @monthly /home/tellervo/monthlybackupscript
```

Used in conjunction with tools like Dropbox or Amazon S3 (see section 13.2), these cron jobs will provide a robust mechanism for securing your data.

13.2   Copying files to/from server

There are times when you will need to transfer files to and/or from your Tellervo server. There are a number of ways to do this depending on your needs and skill level.

13.2.1   Virtualbox shared folder

Virtualbox includes a mechanism for sharing a folder on your physical computer with your virtual server. This is probably the easiest option as once mounted the folder can be seen on your server and files copied into and out of it as normal. In the Virtualbox Manager select your server and press the 'Settings' button. Now go to the 'Shared folders' tab and click the add folder button. Select the folder on your physical computer that you'd like to share and give it a name e.g. 'share'. Make sure you tick the 'make permanent' and 'auto mount' checkboxes. Now reboot your virtual server and your shared folder should be available at /media/share. If the folder has not mounted it may be because the tellervo user is not a member of the vboxsf group. In this case you'll need to enter the following before rebooting the server:

```
✎     sudo adduser tellervo vboxsf
```


13.2.2   Secure copy (SCP)

The most commonly used method for copying files to and from Linux computers is SCP. SCP is a secure transfer protocol based on SSH. You can use SCP to copy files to/from Linux and OSX computers or to Windows computers with a suitable SCP client the most common being WinSCP. To use SCP you'll need to install SSH on your server:

```
✎     sudo apt-get install openssh-server
```


13.2.3   Dropbox

Dropbox is a commercial application that most people are familiar with. No cost accounts can be set up that support up to 2Gb of data. There is a Linux command line client (i.e. without the usual graphical interface) that will syncronise files between your server and Dropbox clients on other computers. To install you'll need to enter the following commands:

```
✎     cd ~&& wget -O - https://www.dropbox.com/download?plat=lnx.x86_64 | tar xzf -
```

```
✎      /.dropbox-dist/dropboxd
```

You'll be prompted to visit a webpage to confirm your account details. Next you'll need a Python script to control the client:

```
cd ~&& wget -O - https://www.dropbox.com/download?dl=packages/dropbox.py
```

```
chmod a+x dropbox.py
```

```
./dropbox.py start
```

```
./dropbox.py autostart
```

The Dropbox client should then download any existing files in your Dropbox folder at /home/tellervo/Dropbox/. You can check the status of your Dropbox folder by typing:

```
./dropbox.py status
```

13.2.4   Amazon Simple Storage Service (S3)

Another commercial tool with good support for Linux is Amazon S3. Amazon provide a 'free useage tier' which stores up to 5Gb of data at no cost. While more complex to set up, S3 provides true commercial grade off-site backups. There is an open source command line client called s3cmd available from `https://github.com/s3tools/s3cmd`. Check out the documentation on Github and the Amazon AWS website for more information.

13.3   Upgrading the server

Before upgrading your server it's important to backup your database. We will always endeavour to make sure that nothing happens to your database, even if the upgrade fails for some reason (in which case the system should roll back to your previous version again), but things don't always go to plan. The method for upgrading your server will depend on what installation method you originally chose:

13.3.1   Upgrading VirtualBox Tellervo Server

If your current version of Tellervo Server is running in VirtualBox one of the easiest methods for upgrading is to backup the database (see section 13.1.3), download and install the latest Tellervo Server (see section 1.2) and restore your database within the new server (see section 13.1.4). This has the advantage of ensuring you are running the latest version of the Ubuntu operating system. If you would rather upgrade your Tellervo server in place, then you can also follow the instructions for upgrading a native Ubuntu server below.

13.3.2   Upgrading a native Ubuntu Tellervo Server

If you are familiar with the Linux command line you can upgrade your Tellervo Server using the dpkg tool. Please note that with limited resources we are unable to support multiple versions of Ubuntu as there can be many versions of the underlying dependencies that can conflict. As a rule we aim to support the most recent LTS version of Ubuntu. If you are running an older version of Ubuntu and end up with dependency problems please contact us.

For those running v1.3.2 or later of Tellervo Server the upgrade can be done by running the following command which will download and install the latest version of Tellervo Server in one go:

```
sudo tellervo-server --upgrade
```

If you are upgrading from an older version of Tellervo Server you'll need to download and install the latest version manually. First, you will need to get the latest Tellervo Server Ubuntu deb package from the Tellervo website. Save this to your server using one of the mechanisms described in section 13.2. Alternatively you can download directly to your server using the command line by typing:

```
    wget --output-document tellervo-server-x.x.x.deb
http://www.tellervo.org/download/?type=server\&os=linux\&version=latest
```

Regardless of how you get the latest deb file you will then need to install it by typing:

```
    sudo dpkg --install tellervo-server-x.x.x.deb
```

The upgrade should run and finish with a test to confirm all has worked correctly.

13.3.3 Upgrading the server operating system

Tellervo is designed to run on the Ubuntu operating system. Due to limited resources we only support one version of Ubuntu at a time, typically the most recent 'Long Term Support' (LTS) release. LTS releases are typically very stable and supported for five years, limiting the need to perform upgrades.

If you are running an older version of Tellervo server on a previous release of Ubuntu by far the simpliest method of upgrading is:

1. Backup your database (see section 13.1.3)

2. Remove Tellervo

3. Upgrade (or reinstall) Ubuntu

4. Install the latest Tellervo Server (see section 1.2)

5. Restore your database (see section 13.1.4)

13.4 Graphical Interface to the Virtual Appliance

For those of you that are unfamiliar with Linux, the basic command line prompt is not likely to be very comfortable. If you are interesting in looking at the server in more detail you may therefore prefer to install a full graphical interface. Unlike Windows, there are a number of different graphical interfaces (or desktops) to choose from in Linux, the most popular being Gnome and KDE. To install one of these you need to type one of the commands listed below. The first line installs Gnome and the second KDE. Windows users that are new to Linux may find KDE more familiar, but Apple users may be more at home with Gnome.

```
    sudo apt-get install ubuntu-desktop
```

```
    sudo apt-get install kubuntu-desktop
```

13.5 Security

The basic installation of the Tellervo server includes the standard configuration for Apache, PHP and PostgreSQL. Although these products are considered secure by default, there are a number of measures that can be taken to make them more so. If your server is only accessible within your local intranet (e.g. behind a robust firewall) then you may not feel it necessary to modify the standard setup. Precautions may be deemed more important if you

server is accessible from the internet. In this case it would be wise to contact your local network administrator for further information.

13.5.1   Usernames and passwords

There are a number of default usernames and passwords setup on your server. If your server is accessible for the internet we strongly advise you to change these defaults and anyone with knowledge of the Tellervo server could access and compromise your machine.

System user - these are the credentials you use to log in to the command prompt in your Tellervo Virtual Appliance. By default the user is 'tellervo' and the password is 'dendrochronology'. To change this log in to the command prompt and type `passwd` and follow the instructions. There is no easy way to recover this password if you loose it.

PostgreSQL database user - these are the credentials used by the webservice to read and write to the database and are set by the database administrator during the initial configuration of the Tellervo server. You are only ever likely to need this again if you want to directly access the database from a third party tool like PGAdminIII. You can reset this password from the Tellervo Virtual Appliance command prompt by typing `tellervo-server --reconfigure`

Tellervo admin user - these are the admin credentials that you use to log in with in your Tellervo desktop application. Be default the user is 'admin' and the password is 'qu3rcu5'. You should change these the first time you open the Tellervo desktop application by going to *Admin* ➜ *Change password*.

13.5.2   Authentication and encryption

Tellervo uses a relatively sophisticated method to ensure that unauthorised users cannot access the Tellervo database through the webservice. It is loosely based around http digest authentication and uses a challenge and response scheme. This makes use of cryptographic hashes (a relatively short digital fingerprint of some data but which cannot be decompiled to retrieve the original data) and nonces (a pseudo-random string used just once). All hashes used in the Tellervo webservice use the MD5 algorithm. This decision will be periodically reviewed to ensure that MD5 is the most appropriate and secure algorithm to use. Whilst an MD5 hash of a short phrase can be compromised, the length and randomness of the original data means with current cracking techniques this is very difficult. For a complete description of Tellervo's authentication procedure see section 19.1.

The default Tellervo server setup, however, uses standard HTTP protocol to communicate between the server and the desktop application. This is the same protocol used for the majority of web pages on the internet and a determined hacker could eavesdrop on this communication. Depending on how important and private you perceive your data you may choose to use Secure Socket Layer (SSL) to encrypt this communication. This is the same technology used by websites such as online banking. To make full use of this upgrade in security you will however also require a SSL certificate from an official licensing authority. These certificates typically cost several hundred dollars per year although some academic institutions make these available to their departments for free. It is possible to setup SSL with a self-signed certificate for free. This will encrypt your communications although it won't protect you from some other style of hacking such as 'man in the middle attacks'. If you use a self-signed certificate you will need to instruct users to turn of checking for valid certificates in Tellervo Desktop. This option is on the Network tab in *Edit* ➜ *Preferences*.

To configure SSL, from the server command line, first of all you need to enable the SSL Apache module:

```
sudo a2enmod ssl
```

```
sudo a2ensite default-ssl.conf
```

The default HTTPS configuration will use a self-signed certificate and key generated by the ssl-cert package. They are good for testing, but the auto-generated certificate and key should be replaced by a certificate spe-

cific to the site or server. For information on generating a key and obtaining a certificate talk to your network administrator. There is also some good information at `https://help.ubuntu.com/lts/serverguide/certificates-and-security.html`.

Once you have your valid certificate you will need to copy it into `/etc/ssl/certs/` and your key into `/etc/ssl/private/` and update the filenames in the Apache configuration file at: `/etc/apache2/sites-available/default-ssl.conf`. Make sure you also change the DocumentRoot variable to the location of your server folder `/var/www/tellervo/`.

Finally you will need to reload Apache by doing:

```
sudo service apache2 reload
```

If all has gone well your webservice will now be available at `https://your.server.ip/` (note the https rather than http).

13.6   Directly accessing the database

Although the Tellervo database is designed to only be accessed by the Tellervo desktop application via the Tellervo server's webservice, you may decide that you'd like to directly access the database yourself. For instance, you may like to write complicated SQL queries to probe your database in ways not currently supported by the Tellervo desktop client.

> ⚠ *Any changes made to the database may have drastic consequences. We strongly recommend that you never write changes directly to the database as this can cause loss of data and corrupt future upgrades to Tellervo.*

13.6.1   PGAdminIII

One of the easiest ways to access the PostgreSQL database is through the application PGAdminIII. This is a cross-platform open source application for communicating with PostgreSQL databases. You can install PGAdminIII on your desktop computer and access the remotely running database using your database user credentials.

For security reasons by default the Tellervo database cannot be accessed from computers outside of the Tellervo server. This may sound peculiar because the webservice can be accessed from computers anywhere on the web, but the database is actually accessed by the webservice, which is essentially a user running on the same computer as the database. To access the database *directly* from a remote computer you must therefore open access first. This is done by adding an entry to the file '/etc/postgresql/9.5/main/pg_hba.conf'. My personal command line text editor of choice is vim, but it is a little confusing to the uninitiated. If you are unfamiliar with command line text editing you are probably best to use pico:

```
sudo pico /etc/postgresql/9.5/main/pg_hba.conf
```

Scroll down passed all the comments, to the bottom of the file. Add the following line:

```
host all all IPADDRESS/32 md5
```

Make sure you replace IPADDRESS with the IP address of the computer you are trying to connect *from*. This is just one style of pg_hba.conf entry. There are many others which allow you to restrict to specific users, computers, networks etc. See the online PostgreSQL documentation for more details. Save your changes and exit by doing CTRL+X.

Next you need to make sure the PostgreSQL server is listening to requests from other IP addresses. To do this you need to edit the postgresql.conf file like this:

> ✎   `sudo pico /etc/postgresql/9.5/main/postgresql.conf`

making the following changes:

Old line - `#listen_addresses = 'localhost'`

New line - `listen_addresses = '*'`

Make sure you remove the hash character at the beginning of the line. Save the file and finally restart the Tellervo server:

> ✎   `sudo tellervo-server --restart`

You should now be able to access your database through PGAdminIII. To do this open the application and go to *File → Add server*. Specify your server's IP address is the host field, and your database username and password.

13.6.2   ODBC

It is also possible to connect to your Tellervo database via an ODBC connection. This allows limited access to the database from a variety of database applications including programs like Microsoft Access for which further details are given here. To use ODBC you will need to install the PostgreSQL ODBC driver (`http://www.postgresql.org/ftp/odbc/`) on your desktop computer.

Once you've installed the driver you can then open a blank database in Access and go to Files, Get external data then Link tables. In the file dialog box change the file type to ODBC Databases(). Next, select the PostgreSQL Unicode driver, then fill out the server details. You should then be able to open the tables and views from the Tellervo server database directly from within Access as if they were local tables. Be warned though that Access and ODBC have many limitations compared to PostgreSQL, especially with regards data types. For this reason we *strongly* recommend using this for read only purposes. Using the ODBC connection to write changes to your PostgreSQL database is quite likely to cause serious issues.

13.6.3   PSQL

The final, and most advanced method is to use the psql client on your server. This is a command line client which can be used to interrogate the database. If you're not already familiar with psql it is unlikely that this is a good method for you to use!

13.7   Tellervo server configuration

13.7.1   Standard server configuration

The Tellervo server can be configured using the command line tool that is installed on both the Virtual Appliance and native server installs. It is the same tool that is run at the end of the native server install, but can be run at any time to reconfigure or test your system. It must be run with superuser privileges therefore `sudo` is required before the command. For instance to get help on usage type:

> ✎   `sudo tellervo-server --help`

Possible options to pass the server are:

- ► '`--help`' – Display a list of the possible options
- ► '`--version`' – Display the version of the Tellervo server webservice and database currently installed
- ► '`--test`' – Run tests on the current configuration

Figure 13.1: Example of the output from the tellervo-server test.

- ▶ '--configure' – Configure the Tellervo server from scratch.
- ▶ '--reconfigure' – Reconfigure the Tellervo server. This should be done if the database name or user credentials change, or if the
- ▶ '--network' – Check network settings and reset if necessary
- ▶ '--emptylog' – Clean the log files and database table logs
- ▶ '--sysconfig' – Generate the systems configuration file
- ▶ '--set-db-pwd' – Reset the PostgreSQL database password
- ▶ '--set-admin-pwd' – Reset the Tellervo desktop login for the admin user
- ▶ '--start' – Start the Tellervo server
- ▶ '--stop' – Stop the Tellervo server
- ▶ '--restart' – Restart the Tellervo server
- ▶ '--upgrade-db' – Run database upgrade scripts
- ▶ '--add-wms-name' – Name of a web mapping server to add to database
- ▶ '--add-wms-url' – URL of a web mapping server to add to database
- ▶ '--del-wms-name' – Name of a web mapping server to delete from database

Figure 13.1 shows an example of asking the server to test the configuration, with all tests passed successfully.

The command line tool stores the majority of settings in the config.php file stored in the base directory of your Tellervo webservice. In theory you could make changes direct to this file, but we do not recommend this unless you know exactly what you're doing.

13.7.2   Advanced server configuration

In addition to the standard configuration options offered on the command line there are a number of other options that can be set. These are not accessible via the command line because as a rule they should only be altered by the Tellervo developers. They are primarily for use by the developers as an alternative to hard coding values within the server files. For instance, one such value is the TRiDaS version being used by the server. This value will only ever need to be changed alongside other substantial changes to the code.

13.8   Managing map services

There is currently no interface in Tellervo desktop that lets you specify the WMS mapping services that should automatically be available to your Tellervo users. Each user can add servers temporarily (see section 7.3) but these will disappear at the end of each session.

To add web mapping services for all uses to a database, you need to use the Tellervo server command line client. Two parameters are needed: the friendly name you'd like to give the server; and the URL. A new WMS can be added like this:

```
sudo tellervo-server --add-wms-name NAMEOFSERVER --add-wms-url URLOFSERVER
```

Similarly a WMS can be removed with the command:

```
sudo tellervo-server --del-wms-name NAMEOFSERVER
```

Publishing data

This chapter covers the publishing of data within your Tellervo database to international data repositories, namely: the International Tree Ring Data Bank[*] (ITRDB); Global Biodiversity Information Facility[†] (GBIF); and Integrated Digitized Biocollections[‡] (iDigBio).

The ITRDB is probably the most well known repository for tree-ring. Established in 1974, it is managed by the World Data Service for Paleoclimatology, at the National Centers for Environmental Information (NCEI) of the National Oceanic and Atmospheric Administration (NOAA). The ITRDB is one of multiple databases held by NCEI related to paleoclimate research including: ice cords; pollen; speleothem; corals; boreholes; and many others. As such, it is primarily focused on dendroclimatology specimens although it does contain broader cultural collections to.

The Global Biodiversity Information Facility (GBIF) is an international network and data infrastructure that seeks to provide anyone with open access to data about all types of life on Earth. Many contributing organizations provide data about specimens and species observations from museums, herbaria, and other scientific organizations from around the world. As a biodiversity database the focus of data within the GBIF repository are species observations and specimen records from all kingdoms of life.

Integrated Digitized Biocollections (iDigBio) is a US-based and funded organization that works closely with GBIF to support the digitization of biocollections in the USA. The National Science Foundation will typically make contribution of data to iDigBio a requirement of a grant where funding is provided to digitize collections.

14.1 Integrated Publishing Toolkit (IPT)

The Integrated Publishing Toolkit (IPT) is an open source tool that can assist with the publishing of data to GBIF and iDigBio. IPT can be installed as an optional extra to your Tellervo Server if you would like to have your data accessible to users around the world through these international repositories. If you don't have a server that is accessible from outside your organization, there are a number of organization that run instances of IPT that will accept dumps of your data manually. This is a relatively simple way to publish your data but does require that you manually export your data periodically. Installing your own instance of IPT alongside your Tellervo database means you can continually deliver data without intervention. IPT can be easily configured to publish portions, or the entirety of your Tellervo database to suit your needs. The Tellervo Server is preconfigured with all the datamapping completed for you, making the process relatively straightforward.

---

[*]ITRDB: `https://www.ncei.noaa.gov/products/paleoclimatology/tree-ring`
[†]GBIF: `https://www.gbif.org`
[‡]iDigBio: `https://www.idigbio.org`

14.1.1 Installing IPT

The IPT website at `https://www.gbif.org/ipt` contains extensive instructions for the installation of the IPT software including a manual and video tutorials. For the most up-to-date information please visit these pages. Some additional instructions and tips are provided below and are aimed at users who have Tellervo Server installed on Ubuntu either natively or as a VirtualBox appliance.

IPT is written in Java and is served via Tomcat, so the first step is to install this:

```
    sudo apt install tomcat9 tomcat9-admin
```

By default, Tomcat runs on port 8080 which makes for rather ugly URLs. As such many people prefer to configure port forwarding so that requests to the standard port 80 are forwarded on to Tomcat on port 8080. There are many ways to do this but to most straightforward is to add the following configuration to the relevant 'Directory' section of your /etc/apache2/sites/enabled/ file:

```
ProxyPreserveHost On
ProxyRequests Off
ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/
```

Enable Apache modproxy and restart the server as follows:

```
    sudo a2enmod proxy && sudo a2enmod proxy_http && sudo service apache2 restart
```

Next there are a number of steps required to get the Tomcat manager running. The first step is to add an admin user by including the following configuration line to /etc/tomcat9/tomcat-users.xml:

```
<user username="admin" password="password" roles="manager-gui,admin-gui"/>
```

Next you to edit the following two files:

```
/usr/share/tomcat9-admin/host-manager/META-INF/context.xml
/usr/share/tomcat9-admin/manager/META-INF/context.xml
```

Within these files you need to locate and comment out the IP restrictions link so:

```
<Context antiResourceLocking="false" privileged="true" >
 <!--<Valve className="org.apache.catalina.valves.RemoteAddrValve"
        allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />-->
</Context>
```

Next you need to allow Tomcat to write to /srv/ipt. This can be done by creating /etc/systemd/system/tomcat9.service.d/override.conf and adding

```
 [Service]
 ReadWritePaths=/srv/ipt
```

Next you need to restart Tomcat and deploy the ipt.war using the Tomcat Manage. Follow the IPT setup steps described in full at: `https://ipt.gbif.org/manual/en/ipt/2.5/initial-setup`

14.1.2 Connecting Tellervo to IPT

Once your IPT server is up and running, the next thing you need to do is to 'create a resource' using the IPT interface described in full at: urlhttps://ipt.gbif.org/manual/en/ipt/2.6/manage-resources. One of the major tasks in connecting Tellervo to IPT is the mapping between the Tellervo/TRiDaS data model to the data model used by IPT. The majority of this work has been completed for you through the creation of a view in the Tellervo database

named vwipt. On the 'source data' page in IPT, simply enter your database details and for the SQL statement use:

```
select * from vwipt;
```

On the 'Darwin Core Mapping' section, the majority of the fields will now be auto-mapped. You may chose to override some of these e.g. institutioncode; institutionid; datasetname, but please take time to understand the scope of each field before changing them.

14.1.3 Submitting data to GBIF

The IPT provides an internal mechanism to provide data to GBIF. Follow the steps to register your institution and your server.

14.1.4 Submitting data to iDigBio

iDigBio also supports the ingestion of data through IPT. Registering your institution and server with iDigBio can be done by emailing your server URL and details to data@idigbio.org.

14.2 International Tree Ring Databank (ITRDB)

Chapter 15

Help and support

15.1 Getting help

At the moment your options for getting help are largely limited to contacting Peter Brewer! Once the user-base of Tellervo expands we will set up forums and mailing lists to assist.

15.2 Support for future development

Both Tellervo Desktop and Server are free software available under the General Public License v3 (see appendix Z). This means you are free to use Tellervo in both academic and commercial environments. However, when we talk about 'free software' (as the license explains) we are talking about freedom of use, not free as in price. Tellervo has inevitably cost a great deal to develop over the years and while you are not asking for a direct contribution, we do need your support for future development.

If there is particular functionality that you would like to see implemented in Tellervo, under the open-source model this can be done in a number of ways:

Implement the feature yourself! – If you are able to program in Java then we would be delighted to assist you to implement new features. You could do this in isolation* but we hope you will do this collaboratively with us and make the new feature available to the rest of our community. Please contact the developers and we will organize a developers Github account for you to contribute to the source code.

Request a feature from the developers – Contact the Tellervo developers and discuss the feature that you would like implemented. If the feature is relatively easy to implement and/or deemed useful for the laboratories that develop Tellervo then we may be able to implement the feature for you.

Pay a third party developer – If you know a third party developer that can make the changes for you then this is also possible. Again, we would ask that you do this in consultation with the existing developers so that any improvements can be contributed back to the community.

Collaborative development – If you have an idea for exciting new functionality we would be pleased to discuss the possibility of collaborative development–for example as part of a grant funded project. The chances of success when applying for infra-structure projects from federal agencies are much greater when proposed as part of a collaborative multi-laboratory project.

---

*Note that although the GPL license allows you to develop Tellervo separately, it does include clauses that require you to make the source code of the software you create also freely available under GPL or a compatible license. If you 'fork' the code in this way you will find it increasingly difficult to benefit from improvements made to the official Tellervo code.

Part II

Developers guide

Developing Tellervo Desktop

Tellervo is open source software and we actively encourage collaboration and assistance from others in the community. There is always lots to do, even for people with little or no programming experience. Please get in touch with the development team as we'd love to hear from you.

16.1 Source code

This section describes how to access the Tellervo source code, but as you are no doubt aware it is normal (if not essential) to use an integrated development environment for developing any more than the most simplistic applications. If you plan to do any development work, it is probably best to skip this section and move straight on to the 'Development environment' section which includes instructions for accessing the source code directly from your IDE. If, however, you just want to browse the source code please continue reading.

The Tellervo source code is maintained in a Git repository at github.com. The simplest way to see the source code is via the web viewer on the Github website: `https://github.com/ltrr-arizona-edu/tellervo`. You can also examine the Javadoc documentation of the code on the Tellervo website.

An overview of the development can be seen through the Tellervo Openhub pages at `https://www.openhub.net/p/tellervo`. Openhub provides graphics summarizing the code over time, including timelines of commits by user.

16.2 Development environment

The IDE of choice of the main Tellervo developers is Eclipse (`http://www.eclipse.org`). There are many other IDEs around and there is no reason you can't use them instead. Either way, the following instructions will hopefully be of use.

We have successfully developed Tellervo on Mac, Windows and Linux computers over the years. The methods for setting up are almost identical.

The first step is to install Eclipse, Java 7 JDK, Subversion and Maven. These are all readily available from their respective websites. On Ubuntu they can be install from the command line easily as follows:

```
sudo apt-get install eclipse subversion openjdk-7-jdk maven2
```

Once installed, you can then launch Eclipse. To access the Tellervo source code you may need to install one or more Git plugins. Go to *Help* ➜ *Install* new software then select the main Update site in the 'Work with' box, then locate the Git plugins under collaboration. You will also need to install the m2e Maven plugin to Eclipse. You may

need to add the Maven update site as this plugin may not be available in the main Eclipse repository. You can do this by clicking the 'Add' button and using the URL `http://m2eclipse.sonatype.org/sites/m2e`. Once installed you will need to restart Eclipse before continuing.

Next you need to get the Tellervo source code. Go to *File → Import → Git - Projects from Git*, then in the dialog select 'clone from URI'. Enter the URI of the Tellervo repository: `https://github.com/ltrr-arizona-edu/tellervo.git`, select the master branch and 'import existing projects' from the subsequent pages.

To launch Tellervo, you will need to *Run → Run Java application*. Create a new run configuration with the main class set to 'org.tellervo.desktop.gui.Startup'.

16.3   Dependencies

Tellervo uses the Maven build system due to it's excellent handling of dependencies. This has become more of an issue as the Tellervo project as grown, as it is now dependent on over 80 different open source libraries.

In an ideal world, any library that your code is dependent on should be available in central Maven repositories and downloaded and installed seamlessly as part of the build process. Maven should also handle transient dependencies (i.e. dependencies of dependencies) automatically. Therefore if a developer knows he needs the functions within a particular library, he simply needs to supply the details of this library without having to worry about the other libraries that this new library is in turn dependent on. Maven also manages versions much more efficiently. If a library is dependent on a particular version of another library this is specified within the Maven build mechanism. This means it is much easier to keep dependencies up-to-date without having to worry about the cascading issues that upgrades often have. In short, Maven is intended to save developers from 'JAR hell'.

In practice, life is not necessarily that simple. Although Maven assists developers in many ways, it also has its own particular quirks and annoyances. The main problem is how to handle the situation when the dependencies you need are not available in central repositories. To solve this you either need to install these jars into your local Maven repository, or make them available in a 3rd party Maven repository. For the ease of developement we have set up a Maven repository as part of the TRiDaS project which can be browsed at `http://maven.tridas.org/`. This repository is already configured within the Tellervo project so assuming this repository is still alive, then your Tellervo project should automatically build. If not, then you will need to install the few non-standard jars. These jars will continue to be maintained in the Tellervo Github repository and can be installed as follows:

- ▶ On your command line navigate to the Libraries folder of your Tellervo source code
- ▶ On Linux and Mac you can then simply run the MavenInstallCommands script
- ▶ On Windows you will need to manually run the commands located in this file

For the record, Tellervo currently depends upon the libraries listed in table 16.1. The table also specifies the licenses that these libraries are made available under.

16.4   Code layout

Tellervo has been actively developed since 2000, so has seen contributions by many different developers. Coding practices have also changed in this time so inevitably there are some inconsistencies with how the source code is organized. For instance, the most recent interfaces have been implemented using the Model-View-Controller (MVC) architecture whereas earlier interfaces contain both domain and user logic in single monolithic classes.

Library	License
Apache commons lang	Apache 2.0
TridasJLib	Apache 2.0
Batik	Apache 2.0
RXTXcomm	LGPL
JDOM	Apache 2.0
Swing layout	LGPL
Log4J	Apache 2.0
JNA	LGPL
Apache mime 4J	Apache 2.0
Commons codec	Apache 2.0
Http Client	LGPL
Http core	Apache 2.0
Http mime	Apache 2.0
Jsyntaxpane	Apache 2.0
L2fprod-common-shared	Apache 2.0
L2fprod-common-sheet	Apache 2.0
L2fprod-common-buttonbar	Apache 2.0
iText	GAPL
PDFRenderer	LGPL
DendroFileIO	Apache 2.0
Java Simple MVC	MIT
JGoogleAnalyticsTracker	MIT
gluegen	BSD
JOGL	BSD+ nuclear clause
WorldWindJava	NOSA
SLF4J	MIT
JFontChooser	LGPL
MigLayout	BSD
PLJava	BSD
PostgreSQL	PostgreSQL License (BSD/MIT)
Forms	BSD
JXL	LGPL
Netbeans Swing Outline	GPLv2

Table 16.1: Tellervo's primary and major first order dependencies along with the licenses under which they are used. Note there are a total of 82 libraries upon which Tellervo draws.

Library	License
Apache commons lang	Apache 2.0
Launch4J	BSD/MIT
NSIS	zlib/libpng
Ant	Apache 2.0
Eclipse	Eclipse Public License - v1.0
ResourceBundle Editor	LGPL
M2Eclipse	Eclipse Public License - v1.0
Subversive	Eclipse Public License - v1.0

Table 16.2: Additional tools/libraries typically used in the development of Tellervo.

Perhaps the most important inconsistency to understand is due to the transistion to the TRiDaS data model. In earlier versions of Tellervo used the concept of a 'Sample'* to represent each data file. Although large portions of Tellervo have been refactored to use the TRiDaS data model classes, there are still some places where the Tellervo Sample remain.

16.5   Multimedia resources

Tellervo includes infrastructure for multimedia resources such as icons, images and sounds within the Maven resource folder 'src/main/resources'. The most extensive is the Icons folder which contains many icons at various sizes ranging from 16×16 to 512×512 as PNG format files. The icons are accessed via the static Builder class. This has various accessor functions which take the filename and the size required, and return the icon itself or a URI of the icon from within the Jar.

16.5.1   Ring remarks

There are two types of ring remarks in Tellervo: TRiDaS controlled remarks and Tellervo controlled remarks. The end user does not know the difference between the two, the only difference between them is how they are handled behind the scenes. TRiDaS remarks are those designated in the TRiDaS schema, whereas Tellervo remarks are those defined specifically for Tellervo. They are represented differently in TRiDaS files like this:

```
    <tridas:remark normalTridas="double pinned"/>
<tridas:remark normal="Tellervo" normalStd="insect damage" normalId="165" />
```

To add a new remark type to Tellervo you will need to first enter it in the database table tlkpreadingnote specifying the vocabulary as '2' (Tellervo). To display a custom icon for this remark in the software, you will need to add a 16×16 and a 48×48 version to the resources an then add an entry to the TellervoRemarkIconMap in org/tellervo/desktop/remarks/Remarks.java. The 16×16 icon is used in the editor interface, and the 48×48 in PDFs.

16.6   Translations

There is internationalization infrastructure in place to enable Tellervo to be offered in multiple languages. This is done through the use of Resource Bundles, one for each language. Within the code, whenever a string is required, it is provided using the I18n.getText() function which then retrieves the correct string for the current locale. If no string is found, then the default language (English) string is returned. There is an Eclipse plugin to assist with this task called ResourceBundle Editor and it can be downloaded from http://eclipse-rbe.sourceforge.net. Once installed it provides a GUI that allows you to simultaneously update all languages at once.

The I18n.getText() function can be passed variables for insertion into the translation next e.g. file name, data value, line number etc. These can be passed either as a string array, or as one or more strings. The values are inserted into the translation string at the points marked 0, 1 etc. For instance, the translation string "File 0 exists. Rename to 1?" would accept two strings the first being the original filename and the second being the filename to rename to. For obvious reasons, only non-translateable strings should be passed in this way as they will be inserted indentically in all languages.

The Resource Bundle also includes support for menu mnemonics (to enable navigation of the menus with the keyboard) and accelerator keys (to enable keyboard shortcuts to bypass menus). Mnemonic are set by adding an

---

*To avoid confusing the original Tellervo class named 'Sample' will be referred to as 'Tellervo Sample' throughout this documentation. Within the code all TRiDaS data model classes are prefixed with 'Tridas' to help avoid confusion. The 'Sample' class is therefore not at all associated with the 'TridasSample' class.

ampersand before the letter of interest (e.g. &File for Eile) in the resource bundle. Accelerators are set by adding the keyword 'accel' with the key of interest inside square brackets after the resource bundle entry. Some examples include:

- ▶ &Graph active series [accel G]
- ▶ Graph &component series [accel shift G]

What key the 'accel' keyword refers to depends on the operating system Tellervo is being run on. In Windows and Linux it is normally 'ALT' wheras on a Mac it is usually the Apple command key.

There are currently minimal translations for UK English, German, French, Dutch, Polish and Turkish. These are by no means complete, and there are number of interfaces that are not internationalized at all. Further assistance is required from native speakers to complete this task.

16.7 Logging

Logging in Tellervo is handled by the SLF4J and Log4J packages. Rather than write debug notes directly to System.out, Log4J handles logging in a more intelligent way. First of all, each log message is assigned a log level which are (in order of severity) fatal, error, warn, info, debug and trace. Through a log4j.xml configuration file contained within the resources folder, we can control the level at which messages are displayed. For instance while we develop we would likely show all messages up to and including 'trace', but when we deploy we might only want to show messages up to and including 'warn'.

Log4J also enables us to log to several places (known as appenders), e.g. console, log file or a component within our application. It is also possible to change the level of logging depending on the log type, so minimal messages can be sent to the console but verbose messages to the log file. Tellervo has the following four appenders configured:

- ▶ Standard log file (tellervo.log) that rolls over up to 2mb of messages

- ▶ Submission log file (tellervo-submission.log) that contains the last 100kb of verbose messages and is used by the bug submission tool to enable users to notify developers of problems.

- ▶ Console – standard messages to the console when launched from command line

- ▶ Swing GUI – a swing component for displaying basic logs to the users in the application.

To alter the way these appenders are configured you need to edit the log4j.xml file. See the Log4J documentation for further information.

Using the logging framework is very simple. Just define a Logger as a static variable in your class like this:

```
private final static Logger log = LoggerFactory.getLogger(MyClassName.class);
```

where MyClassName is the name of the current class. Then you can log messages simply by calling log.warn('My message'), log.debug('My message') etc.

Before managed logging was introduced to Tellervo, debugging was often handled through the use of System.out and System.err messages. To ensure that these messages are not lost we use another package called SysOutOver-SLF4J. This redirects messages sent to System.out and System.err to the logging system. This is a temporary solution so when working on older classes, please take the time to transition these older calls to the proper logging calls. We can then remove the need for SysOutOverSLF4J.

16.8 Preferences

It is helpful to remember certain user preferences e.g. colors, fonts, usernames, URLs, last folder opened etc so that they don't have to do tasks repeatedly. This is achieved through the use of a preferences file. This file is stored in a users home folder and consulted to see if a preference has been saved, otherwise Tellervo falls back to a default value.

The preferences are accessed from the static member App.prefs. To set a preference you can do the following:

```
App.prefs.setPref(PrefKey.PREFKEY, "the value to set");
```

where PrefKey.PREFKEY is an enum containing a unique string to identify the preference, and the second value is the string value to set. There are other specific methods for different data types e.g. setBooleanPref(), setIntPref(), setColorPref() etc.

To retrieve a preference, you use a similar syntax:

```
App.prefs.getPref(PrefKey.PREFKEY, "default value");
```

When you get a preference the second parameter contains the default value to return if no preference is found. Like the setPref() method, there are also a host of getPref() methods for different data types.

16.9 Build script

Tellervo is built using Maven and is controlled through the pom.xml file stored in the base of the Tellervo source code. Previous versions of Tellervo used Ant but managing the increasing number of dependencies as Tellervo has grown became too onerous (see section 16.3 for more details).

Earlier versions of Tellervo were deployed using Java WebStart technology primarily because this is platform independent and requires just a single click for a user to install. However, this was replaced some time ago with native installers for the major platforms due to various complications associated with native libraries (see section 16.9.3) required for 3D graphics and serial port hardware. We have also found most users are more comfortable with the standard install procedures that they are used to on their operating systems.

While you develop, Maven should automatically build Tellervo for you in the background. Specific build commands are only required as you approach a release. We use the standard Maven 'life cycle' for building, packaging and deploying Tellervo. The method for doing this in Eclipse is by right clicking on the pom.xml file and selecting *Run as → Maven package* etc. If the option you want is not displayed, you will need to create an entry in the build menu by going to *Run → Run configurations*, then create a new Maven Build with the required 'goal'. The main goals are as follows:

clean - This deletes any previously compiled classes and packages in the target folder. It should only be necessary to run this occassionally if Maven has got a bit confused. If this is the case you may also need to force Eclipse to clean too by going to *Project → Clean...*

generate-sources - Runs JAXB to generated classes representing the entities within the Tellervo schema (see section 16.10 for further details). The classes are also generated for TRiDaS entities, but these are deleted in favour of using those provided by the TridasJLib library.

package - Depending on the profile specified (see below) this compiles Tellervo into a Jar file aor native Windows, MacOSX, Linux and Unix packages. These are all placed in structured folders within 'target Binaries' ready for deploying on a website.

install - This installs the compiled jar in your local Maven repository. This is normally used when you are building a library that is being used by another program. It is therefore not necessary for Tellervo.

deploy - This uploads the compiled jar into the maven.tridas.org repository. Note that you will need to either run this phase from the command line or by setting up a custom run configuration in Eclipse.

16.9.1 Building native installers

A pragmatic decision was made to use the closed source Install4J install for packaging. Although an open source tool is preferable (NSIS was used for some time), Install4J is the only tool we've found that gives us the one click package and coding signing for all three target platforms (Windows, OSX and Linux). Install4J is a commercial tool although the distributing company (EJ Technologies) provides free licenses to approved open source projects. The lead developed (Peter Brewer) has license keys to run Install4J for Tellervo. The drawback with this approach is that only Peter can produce the final releases.

Install4J includes a GUI application for generating and editing the configuration files used to compile the final packages. The configuration is saved in the tellervo.install4j file within the Native folder. During the 'package' phase of the life cycle, Maven runs Install4J and produces Linux, OSX, Windows, and Unix installer packages in the target/binaries folder.

Because this requires the closed source Install4J binary (and licence) as well as code signing certificates, building the native installers does not happen by default as this would make the build script non-portable. When the Maven package goal is run just a Jar file is created by default. To build all the native installers, you'll need to run package with profile set to 'binaries'.

16.9.2 Code signing

From Windows Vista and MacOSX 10.7 (Lion) onwards code signing has become increasingly important. Windows applications that are not signed result in terrifying warning messages, and in OSX by default they don't run at all. The idea behind code signing is that it provides some level of security for the user as code signing certificates can be revoked if an application is deemed to be malicious.

To sign an application you first need two code signing certificates: one for Windows and Java; and another for OSX as Apple only support certificates issued by themselves. A generic Windows and Java certificate can be purchased from a variety of suppliers (e.g. Verisign etc) the Apple certificate must be purchased annually through the Apple Developer Connection.

With the inclusion of the GateKeeper technology in OSX 10.7, code signing has become almost essential in OSX. The jar wrapped in native apps will fail to load under default security settings and the OS reports the file as 'damaged' because the JavaApplicationStub used has an existing signature and the contents of the package has changed. This signature must be removed or replaced with another before the application can be launched. If a self-signed or third party certificate is used then the GateKeeper will block the application saying that it isn't from a trusted developer. It is possible for users to go to the *System Properties → Security and Privacy* and set the application security level to 'any developer' but this is long and involved for novice users and results in a lot of warnings.

The best way to fix the issue is to sign the application using an Apple certified Developer ID certificate and GateKeeper will allow the application to launch with the default security settings. Obtaining this certificate is a fairly involved process and requires an annual subscription to Apple Developer Connection. Note that your certificate must begin with 'Developer ID' to work. Other Apple-provided certificates are used for distributing your application through the Mac App store and will not work.

The OSX code signing is only possible within OSX or on other platforms using the Install4J application. No other cross-platform code signing is currently available. This is one of the primary reasons for choosing Install4J to package Tellervo.

All the code signing is handled within Maven. The current set up in the Maven pom.xml file is configured for Ubuntu Linux so if you are developing on another platform you may need to make changes. Maven needs to know the location of the certificates as well as the keys to unlock them. This is done by making changes to your Maven settings.xml file (stored at /.m2/settings.xml in Linux). In this file you will need to modify your profiles section to read as follows:

```
1   <profiles>
2    <profile>
3      <id>default</id>
4      <activation>
5         <activeByDefault>true</activeByDefault>
6      </activation>
7      <properties>
8        <cert.password>mylongpassword</cert.password>
9        <pkcs12.filename>/full/path/goes/here/certificate.p12</pkcs12.filename>
10       <install4jc.binary>/full/path/goes/here/install4jc</install4jc.binary>
11     </properties>
12    </profile>
13  </profiles>
```

16.9.3 Native libraries

Although Tellervo is written in Java, it requires a number of native libraries to make use of OpenGL 3D graphics capabilities and to access the serial port of the computer. This libraries are different for each operating system, and they are also different for 32 and 64 bit machines. The correct libraries must be made available to the OS and are therefore typically installed outside of the jar file as part of the installation process.

On Windows these libraries take the form of Dynamic Link Libraries (DLL) files which are normally placed in the same folder as the executable:

- ▶ gluegen-rt.dll
- ▶ jogl_awt.dll
- ▶ jogl_cg.dll
- ▶ jogl.dll
- ▶ rxtxSerial.dll

On MacOSX the libraries come as JNILIB files and on Linux as .so files e.g.:

- ▶ libgluegen-rt.jnilib and libgluegen-rt.so
- ▶ libjogl_awt.jnilib and libjogl_awt.so
- ▶ libjogl_cg.jnilib and libjogl_cg.so
- ▶ libjogl.jnilib and libjogl.so
- ▶ librxtxSerial.jnilib and librxtxSerial.so

On Linux systems this are installed into the /usr/lib folder and on MacOSX they are included within the .app file.

We have experimented with techniques for packaging the libraries within the jar, then extracting the correct libraries based on architecture and dynamically loaded at runtime. This seemed to work relatively well for JOGL/Gluegen, but not rxtx. On certain graphics cards the JOGL/Gluegen libraries also caused a SIGSEGV fault. All native libraries are therefore now handled by the installer for the respective platforms.

16.10 Java Architecture for XML Binding - JAXB

Java Architecture for XML Binding (JAXB) is a technology that automatically maps Java classes to XML schemas and vice versa. It includes the ability to *marshall* data from Java classes to XML files and *unmarshall* data from XML files into Java class representations.

JAXB is used by TridasJLib to create Java class representations of the TRiDaS data model. It is also used directly in Tellervo to create classes for the Tellervo web service. Although the Tellervo webservice is based heavily on

TRiDaS (the two were developed in parallel), the Tellervo schema extends TRiDaS by including classes such as dictionaries and the 'box' concept which are required for a lab data management application.

The Tellervo JAXB classes are automatically built by Maven using the 'maven-jaxb2-plugin' and placed within the 'src/main/generated' folder. Please note that any manual changes to these classes will automatically be overriden the next time Maven is run. If you feel that changes are necessary to these classes then it is likely that one or more of the following needs modification:

- ▶ The Tellervo schema located in 'src/main/resources/schemas'
- ▶ The Tellervo JAXB bindings located in 'src/main/resources/binding'
- ▶ The specification for how JAXB is run located in the 'pom.xml' file

Please note that JAXB supports plugins and extensions for enhancing the classes that it produces. One thing to note in the Maven pom.xml is a nasty workaround when running JAXB. As the Tellervo schema depends on the GML and TRiDaS schemas, these classes are also built by JAXB. These classes however are already provided by the DendroFileIO library. It should be possible to use a feature called 'episodes' to handle this but this seems buggy and causes issues. For now, we use an antrun task to delete the duplicate classes immediately after they are produced.

16.11 Java version

Tellervo requires Java 8 or later to run. Support for Java 6 was dropped following it's end of life in 2013 and Java 7 support was dropped in August 2017. As always, we recommend you use the most recent version of Java possible for security reasons. Tellervo has been tested with Java distributions from both Oracle and OpenJDK, but development is focused using OpenJDK to follow the open source ethics of the project.

16.12 Developing graphical interfaces

Like the rest of the code, a number of different styles and methods have been used for the creation of interfaces in Tellervo. Many of the earlier interfaces were hand coded, but in recent years WYSIWYG graphical designers have been used to enable the creation of more complex designs. Most interfaces are now Swing-based although AWT widgets are used in places.

Some interfaces were created using the graphical designed in Netbeans IDE. These can be identified by the presence of companion .form files and warning comments in the code indicating which sections are autogenerated. The major drawback with the Netbeans form designer is that it cannot cope with externally made changes. If changes are made to the files outside of Netbeans, then the Netbeans form designer can no longer edit these files so please make sure you are certain this is how you want to proceed. The classes generated by Netbeans are typically used by a subclass via inheritance so that any changes can be external to the form designer generated files.

More recently the Google WindowBuilder Pro tool has been used for interface design. This has the benefit of (usually) being able to parse existing code enabling the modification of existing dialogs. WindowsBuilder does have its quirks though so make sure you keep up-to-date with new releases.

16.13 Supporting measuring platforms

The support for hardware measuring platforms has been designed to be as modular and extensible as possible. Adding support for additional measuring platform types should therefore be quick and painless!

To begin, you need to extend the abstract class org.tellervo.desktop.hardware.AbstractSerialMeasuringDevice. You can of course also extend the class implementation of another platform if you only need to modify a few settings. This is the case for both the QC10 and QC1100 devices which extend the GenericASCIIDevice class. The implementation code is identical for all three, but the derived classes set the port settings to the default values for the two QuadraChek boxes.

There are a number of methods that you will need to override from the base class. If you use Eclipse to generate the class it will create placeholders for all the relevant methods. The toString() method enables you to return the name for the device you are implementing, whereas all the is...() methods enable Tellervo to understand the capabilities of the device. For instance some devices will accept requests to zero the current measurement and/or request the current measurement value, while others will not (instead they rely on hardware buttons on the device itself). Some devices can have the port settings (such as baud, parity, stopbits etc) altered and the corresponding is...Editable() functions indicate whether this is possible. All user interfaces in Tellervo are modified in accordance with these methods and show the user only relevant buttons.

The guts of the work in the class are performed in the following methods:

setDefaultPortParams() – this method sets all the default port communications parameters. The abstract class already sets typical values so you only need to override this if they need to change.

doInitialize() – this method is run when the platform is initialized. If your platform needs to do any sort of handshaking then this is where this should be done.

serialEvent() – this method handles any events that are detected from the serial port. All new data received from the platform is decoded here. Values and errors are passed on via the fireSerialSampleEvent() method. Remember that all values should be sent as measurements in microns. If the platform has the ability to work in different units the UnitMultiplier value must be used to ensure the units set by the user are handled correctly.

zeroMeasurement() – if your platform responds to requests to zero the measurement value this is where you should implement this.

requestMeasurement() – if your platform responds to requests to send the current measurement value then you should implement this functionality here.

Once your new class is complete you need to inform Tellervo that it exists. To do this you need to register the device in org.tellervo.desktop.hardware.SerialDeviceSelector. You should then be able to launch Tellervo and test your new device in the preferences dialog. The relevant parts of the dialog will be enabled/disabled depending on how you set the corresponding is...Editable() methods in your class. The dialog also includes a seperate test window with a console for debugging the raw data received from the serial port.

16.14   Writing documentation

The documentation in Tellervo is written in the well established typesetting language $\LaTeX\,2_\varepsilon$. $\LaTeX$ is a great tool for producing high quality documentation with a good structure and style. Unlike standard WYSIWYG (what you see is what you get) word processing applications like Microsoft Word, $\LaTeX$ uses simple plain text code to layout a document so that it is often described as WYSIWYM (what you see is what you mean)! The style of a $\LaTeX$ document is handled separated enabling the author to concentrate on content. By removing the possibility for authors to tinker with font sizes etc, $\LaTeX$ forces you to create clear, well structured documents. For further details see `http://en.wikibooks.org/wiki/LaTeX/`.

The master document is 'Documentation/tellervo-manual.tex' and imports each chapter file. To build the documentation you will need a editor to update and compile to PDF. One of the easiest editors is the online service Sharelatex.com. Alternatively, if you want to write offline: on Linux I would suggest Kile; on MacOSX TeXShop;

and on Windows WinEdt. To add or edit bibliography entries you will also need a BIBTEX editor such as JabRef or BibDesk.

LATEX has fantastic cross-referencing and citation functionality built in. Please follow the lead of the existing documentation!

16.15   Recording screencast tutorials

One way we are trying to support users is by making clear video screencast tutorials of the major workflows in Tellervo. I will describe the steps require to use a combination of ffmpeg, x11grab and Pitivi to do this in Linux, but there are many other tools available so if you prefer to use them please be my guest. I will, however, note that making the tutorials high resolution is extremely important as lower quality video codecs make it very difficult to read screens. So far I've been unable to use the simple wrapper tools like recordMyDesktop to record at high resolution.

The highest resolution video accepted by Youtube and Vimeo at the time of writing is HD720 (1280×720). We will therefore work towards recording our tutorials to this level. By natively recording a section of our desktop of this size we remove the need for shrinking and distoring our video so it is best to ensure all the action goes on within a rectangle of this size. I've created a desktop wallpaper with these dimensions marked from the upper left hand corner.

To do the recording we need to install the following:

```
sudo apt-get install ffmpeg pitivi x264
```

16.16   Making a new release

Making a new release should be a relatively quick and simply process, but there are still a few things to remember:

- ▶ Make sure this documentation is up-to-date!
- ▶ Update the logging appenders to an appropriate level so that the user is not swamped by debug messages
- ▶ If this release relies upon a certain version of the Tellervo server, make sure you set this correctly in '/src/main/java/org/tellervo/desktop/versioning/Build.java'. This is important to ensure that users aren't working against an old version of the server which could have unexpected side-effects.
- ▶ Increment the build version number in the pom.xml
- ▶ Increment the build version in the tellervo.install4j file
- ▶ Update the splash screen and background graphics.
- ▶ Check the code in Eclipse and eliminate as many warnings as possible.
- ▶ Make sure the developers metadata is correct in the pom.xml. Add any new developers that have joined the project since the last release.
- ▶ Run Maven package.
- ▶ TEST!
- ▶ Deploy to maven.tridas.org by running Maven deploy.
- ▶ Copy '/target/binaries' to the http://www.tellervo.org/download/ folder. The new release will automatically be added to the options for download.

▶ If this new release should be the recommmended release for internal and/or external uses, alter the index.php page to reflect this.

Developing Tellervo Server

The Tellervo server is made up of a PHP webservice run by Apache, connecting to a PostgreSQL database.

The Tellervo webservice is written entirely in PHP. Like the Desktop Client, the server is developed with Eclipse so most of the setup steps are identical (see chapter 16). You will, however, probably want to install the PHP development plugin so that you get syntax highlighting etc. See the Eclipse PDT website (`http://www.eclipse.org/pdt/`) for further information.

17.1 Webservice

Before making any changes to the webservice you will need to understand its architecture first. Chapter 18 contains details of the communications specification.

17.1.1 Creating new series

Due to the complications arising from the virtual measurement concept, creating new series in Tellervo is necessarily more complicated than any other of the TRiDaS entities. The workflow required to create a new series is illustrated in figure 17.1.

17.2 Server package

The Ubuntu server package is built by Maven at the same time as the desktop package (see section 16.9) during the package goal.

The server packaging is done as a secondary execution of the JDeb plugin. JDeb is configured in the pom.xml by including all the files that need to be copied along with where in the target file system they should be placed. The database files are installed to '/usr/share/tellervo-server' and the webservices files to '/var/www/tellervo/'.

The metadata for the deb file is included in the control file located in Native/BuildResources/LinBuild/ServerControl. JDeb makes use of Ubuntu's excellent package management system to handle the dependencies. Adding or editing dependencies is simply a matter of changing the 'depends' attribute control file.

The ServerControl folder also contains scripts called preinst, postinst, prerm and postrm, which are launched before and after installation, and before uninstalling. These files are called with different parameters depending on whether this is part of a fresh install, an upgrade, or an aborted install. There are a number of rules that the resulting deb package should follow (e.g. if a program is configured twice, then the second run should know and
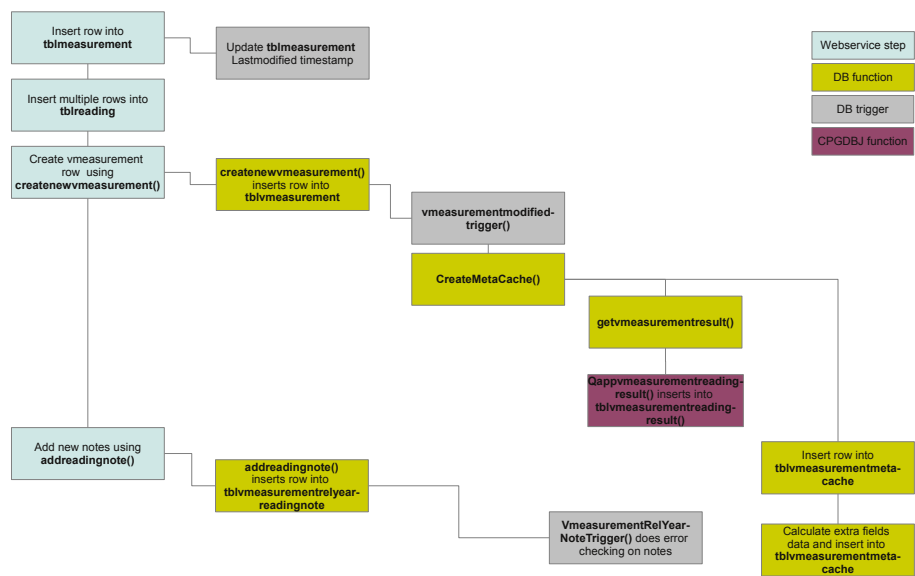
Figure 17.1: Illustration of the steps that happen during the creation of a new measurement series. The stages are presented top to bottom in the approximate order in which they are executed. The majority of the processing is done as a result of the database function createnewvmeasurement() being called by the webservice.

understand about previously provided details), the details of which can be found in the Debian Policy Manual*, along with information how and when each of the pre and post scripts is run. Hopefully this side of the server packaging will not need to be touched again, but if you are making changes and are doing anything more than simple tweaks, please consult the Debian policy documentation.

If changes are required, figure 17.2 highlights the order that the pre and post scripts are run and with what parameters.

The postinst script is used to trigger the interactive script that helps the user configure the Tellervo server (described further in section 17.2.1). The steps are as follows:

- ▶ Check the user running the script is root as we're doing privileged functions
- ▶ Generated scripts from templates
- ▶ Configure PostgreSQL database, creating users and/or database if requested otherwise obtaining details if they already exist
- ▶ Configure PostgreSQL to allow access to the specified database user
- ▶ Configure Apache to access the webservice
- ▶ Verify setup by checking Apache and PostgreSQL are running, that the webservice is accessible, the database is accessible and that various configuration files can be read
- ▶ Print test report to screen

17.2.1   Tellervo server script

At the heart of most of the configuration and control of the Tellervo server is the tellervo-server script. This is a command line PHP script that is launched after installation and can be re-run by the user to make changes to the configuration. Although such a script would normally be written in Bash or similar, we decided to go with PHP because of the requirement to interact with the Tellervo PostgreSQL database.

The script isolates the common tasks performed into functions. It uses the getopt() function to read both long (e.g. –blah) and short (e.g. -b) arguments from the command line. These depending on the arguments given, the script then calls the relevant functions.

To comply with standard protocols, the script uses the exit() function to return whether the requested task was successful or not. Returning zero means the script was successful, and returning any other integer means the script failed. This is important so that the package management system knows when things have gone wrong, and can then attempt to roll back if possible.

The script includes a number of helper functions and classes that you may find useful when modifying the script:

echoTruncatedString($str, $length – echos a string to the console but truncates it to $length if necessary. If the string is shorter than $length, then it is padded with spaces. This is useful to ensure the following text is displayed aligned, e.g. test results.

requireRoot() – check whether the user running the script has root privileges.

checkServiceIsRunning($service) – checks whether the named service is running on the system. This is performed by checking whether the provided string is present in the response from the shell command 'ps ax'.

setConfigVariable($var, $value) – does a search and replace for a placeholder variable in the config.php file, replacing with $value. Placeholders should be stored in the config.php template as `%%VARIABLENAME%%`.

promptForPwd($isCreating=TRUE) – is an interactive script for getting a password from the user. It checks that the password is strong and asks for it twice to check for typos.

---

*Debian Policy Manual – `http://www.debian.org/doc/debian-policy/` but a more accessible description is found at `http://wiki.debian.org/MaintainerScripts`

Standard installation procedure

```
 ┌──────────────┐   FAILED   ┌────────────────────┐   FAILED
 │ preinst install ├─────────►│ postrm abort-install │
 └──────┬───────┘            └─────────┬──────────┘
        │ OK                            │ OK
        │                               │
   Files are unpacked                   │
        │                               │
        ▼                               │
 ┌──────────────┐  FAILED               │
 │ postinst configure │                 │
 └──────┬───────┘                       │
        │ OK                            │
        │                               │
  "Installed"    "Failed-Config"  "Not Installed"  "Half Installed"
        │                               │          "Reinst required"
        ▼                               ▼
 ┌──────────────┐            ┌────────────────────────┐
 │ Successful exit │        │ Exit with error message  │
 └──────────────┘            └────────────────────────┘
```

Upgrade from version 1.2-3 to 1.2-4

```
 ┌──────────────────────────┐ FAILED ┌──────────────────────────────┐ FAILED ┌──────────────────────────────┐ FAILED
 │ 1.2-3->prerm upgrade 1.2-4 ├───────►│ 1.2-4->prerm failed-upgrade 1.2-3 ├──────►│ 1.2-3->postinst abort-upgrade 1.2-4 │
 └────────────┬─────────────┘         └──────────────────┬───────────┘          └────────────────────────────┘
              │ OK                          OK           │                                │ OK
              │                                          │
              ▼                                          ▼
 ┌──────────────────────────┐ FAILED ┌────────────────────────────┐ FAILED
 │ 1.2-4->preinst upgrade 1.2-3 ├─────►│ 1.2-4->postrm abort-upgrade 1.2-3 │
 └────────────┬─────────────┘         └─────────────┬──────────────┘          "Half Installed"
              │ OK                          │ OK                                   (1.2-3)
              │                             ▼                                  "Reinst required"
     Files are unpacked         ┌────────────────────────────┐ FAILED
              │                 │ 1.2-3->postinst abort-upgrade 1.2-4 │
              │                 └─────────────┬──────────────┘
              │                          │ OK                    "Unpacked"
              │                                                    (1.2-3)
              │        "Installed"
              │          (1.2-3)                   "Installed"
              │                                      (1.2-3)
              │                 ┌────────────────────────┐
              │                 │ Exit with error message  │
              │                 └────────────────────────┘
              ▼
 ┌──────────────────────────┐ FAILED ┌──────────────────────────────┐ FAILED ┌──────────────────────────────┐ FAILED
 │ 1.2-3->postrm upgrade 1.2-4 ├──────►│ 1.2-4->postrm failed-upgrade 1.2-4 ├──────►│ 1.2-3->preinst abort-upgrade 1.2-4 │
 └────────────┬─────────────┘         └──────────────────────────────┘          └───────────────┬──────────────┘
              │ OK                          OK                                          │ OK
     Old files are deleted                                                              ▼
              │                                                            ┌────────────────────────────┐ FAILED
              ▼                                                            │ 1.2-4->postrm abort-upgrade 1.2-3 │
 ┌──────────────────────────┐ FAILED                                      └───────────────┬──────────────┘
 │ 1.2-4->postinst configure 1.2-3 │                                             │ OK
 └────────────┬─────────────┘                                                    ▼
              │ OK                                                  ┌────────────────────────────┐ FAILED
              │                                                     │ 1.2-3->postinst abort-upgrade 1.2-4 │
       "Installed"           "Failed-Config"                        └───────────────┬──────────────┘
         (1.2-4)               (1.2-4)           "Installed"                 │ OK
              │                                    (1.2-3)      "Unpacked"
              │                                                   (1.2-3)          "Half Installed"
              ▼                                                                       (1.2-3)
 ┌──────────────┐                        ┌────────────────────────┐              "Reinst required"
 │ Successful exit │                     │ Exit with error message  │
 └──────────────┘                        └────────────────────────┘
```

class Colors – can be used to display coloured text on the console. Useful for highlighting errors and test results.

17.3 Handling version dependencies

In an ideal world, the API for how clients talk to the Tellervo server would never change. Unfortunately, we don't live in an ideal world! New features in Tellervo will require changes to the API, as will changes to TRiDaS. In anticipation of such changes, the Tellervo server includes a mechanism for detecting when a client is too old to handle the API that it is using. In this case the server will refuse to handle the request. A similar complementary mechanism is in place in the client for instances when a client is attempting to talk to an older server that it no longer supports.

Although the Tellervo desktop client is the primary software that talks to the Tellervo server, there are others including PAST4 and a prototype R client. As such, it is important to include the ability to specify the oldest versions of clients that are able to connect, and also to be able to specify different versions for different types of clients.

It is also necessary to include the ability to allow or disallow access to the server by unknown client applications. If a new program is written by other developers and it attempts to access the server it could contain bugs (or even malicious code[†]) that interferes with the server. For a production instance of the server this is obviously undesirable, therefore the systems configuration option 'onlyAllowKnownClients' is set to TRUE.

17.3.1 Client requiring a recent server

In the event that changes are made to the client that means it needs a particular version of the server to run, then this information is specified in the static string App.earliestServerVersionSupported. This value should be a three part version number e.g. 1.1.1. No other changes are required. When the user tries to connect to a server with a version prior to the minimum version specified, they will be shown an error message explaining that they need to get the Tellervo administrator to upgrade the server.

17.3.2 Server requires a recent client

In the event that changes are made to the server which mean that a recent version of the client is required, then this information is stored in the database in the tblsupportedclient table. The 'client' field should contain a unique portion of the HTTP_USER_AGENT header provided by the client. The version number field should be a 2 or 3 part number e.g. 1.1 or 1.1.1. In the event that a user tried to connect to a Tellervo server using a client that is too old, then they will be given a clear error message informing them that they need to upgrade.

17.4 Handling server configuration

The Tellervo server is configured using two main PHP files: config.php and systemconfig.php. The configuration is split into two primarily because the config.php values are considered to be editable by the server administrator, whereas those in systemconfig.php should normally only be edited by Tellervo developers.

If you want to make configuration options editable by the administrator of the Tellervo server, then these should be implemented within the config.php file. There is a config.php.template file which is used to construct the config.php file on the users system. Simply adding hardcoded entries to this file is the simplest way when a default value is

---

[†]Keep in mind though that a user with the necessary privileges would need to provide this new program with their credentials for it to make changes to data.

appropriate. If the value of your field needs to be generated either by asking the administrator a question (e.g. name of lab), or dynamically at the time of installation (e.g. IP address of the server) then this template file should contain placeholder values which can then be replaced by the tellervo-server configuration script. For instance the config.php.template file contains a placeholder for the hostname of the server like this: `$hostname = "%%IP%%";`. The value is set by the tellervo-server script using the function `setConfigVariable($var, $value)`. Keep in mind though, that during an upgrade, the config.php is maintained and not replaced. If you make additions to the config.php.template you will also need to make provision for handling changes to the end users existing config.php.

If you want to add new configuration fields that don't need to be edited by the system administrator, these should be handled in the systemconfig.php file. The systemconfig.php file is automatically generated during installation/upgrade of the server from entries in the database table tblconfig. This means that any changes to the system configuration can be handled as part of the database upgrade simply by adding new rows or editing existing rows in tblconfig. Each entry in this table is made available to the webservice as a global variable once the tellervo-server script has been run. For instance the row containing key=wsversion and value=1.0.0 is available as the variable $wsversion within the webservice.

17.5   Making a new release

As mentioned in section 17.2, the server package is created at the same time as the desktop binaries as part of the Maven package procedure. There are, however, a number of steps you need to undertake to make sure this goes smoothly.

▶ Make sure this documentation is up-to-date!

▶ Increment the `<serverversion>` tag in the pom.xml file

▶ Make sure that any upgrades that need to be made to the database are included in a new and unique SQL file stored in Databases/db-upgrade-patches. Each file from this folder is run by the installer unless it has previously been run. If it can't be run in a transaction make sure the extension is .notransaction.sql.

▶ If this version of the server needs a particular version of the client then you'll need to set this value in the tblsupportedclient table by including a relevant SQL statement in your db-update-patches script e.g.:

> ✏   `UPDATE tblsupportedclient SET minversion='2.13' WHERE client='Tellervo WSI';`

▶ Run maven package to produce the server deb file.

▶ Create a clean Ubuntu VirtualBox installation with the user 'tellervo' and the password 'dendrochronology'

▶ Install the deb file and dependencies as normal, configuring the server with temporary settings

▶ Run `sudo tellervo-server --deploy` to remove the temporary settings and set up ready for users first run

▶ Shutdown the virtual appliance then export. Add in all the metadata as required.

▶ Copy the virtual appliance file and the native deb file to the web server for release. The server PHP code will automatically pick up the extra files and offer them to users.

▶ Update the download/index.php file to reflect the 'preferred' version choice for users.

▶ TEST! If users are running this as an upgrade, then we need to ensure this goes smoothly. Although they are told to backup their database before running we should assume they've ignored the warning and that we are altering precious data. Test both a fresh install and an upgrade from the previous version.

17.6   Administering the Maven repository

The following information is only necessary for the lead-developer and outlines the steps necessary to install and maintain the central Maven repository for Tellervo. This Maven repository should provide all other developers with the libraries required to develop Tellervo and which are bundled in the release packages.

The repository tool that we currently use is Apache Archiva. Installation is relatively simple:

1. Download the zip bundle from the Apache Archiva website

2. Unzip and place on the server in a suitable location (e.g. usr/share/apache-archiva)

3. Run 'sudo bin/archiva start'

If you have an existing backup of the Archiva database then you can place this in the data folder and you should be good to go. If not you will need to do the following steps to configure the repository from scratch:

1. Go to 'http://www.tridas.org:8080/archiva/' in your web browser and set up the admin account. If you're setting this up on another domain remember you'll need to change the repository URLs in both the distributionManagement and repositories sections of your pom file

2. In the repositories tab you need to configure both releases and snapshot repositories

3. Set up users with 'Repository Manager' permissions for each user that would like to deploy to the repository. They will need to configure their .m2/settings.xml file to do this

4. Set up the guest user to have 'Repository Observer' permissions for each repository. This means that people can anonymously access artefacts from the repository

5. Add the following remote repositories:

   Geotk – Identifier: geotk; URL: http://maven.geotoolkit.org/

   Geomajas repository for JPedal – Identifier: maven.geomajas.org; URL: http://maven.geomajas.org/

   maven.iscpif.fr – Identifier: maven.iscpif.fr; URL: http://maven.iscpif.fr/snapshots/

   thirdparty.maven.iscpif.fr – Identifier: thirdparty.maven.iscpif.fr; URL: http://maven.iscpif.fr/thirdparty/

6. Add proxy connectors for the above repositories

7. Run the MavenDeployCommands.sh script to deploy the handful of repositories that we need that are in *no* repositories

8. You will need to run a similar file to deploy dependencies for TRiDaSJLib. See the tridas source code for details

The remote repositories contain libraries maintained by others that are not (at the time of writing) in the central Maven repositories. We include them here to ensure they are cached in our repository and so are available to our developers even if these external repositories go down. Our new repository will be populated with these external artefacts when a developer first requests them. They are retrieved from the external repositories and cached in ours.

It is possible to manually deploy artifacts to the repository using the web interface, but this is slow and tedious. We normally deploy direct from Eclipse using the maven deploy goal.

Webservice specifications

This chapter outlines the basic syntax required for talking to the Tellero webservice and is largely aimed at developers who want to create new clients to interact with the Tellervo server.

The Tellervo database is accessed solely through the webservice interface. A webservice is an interface designed to be accessed by programs that send requests and receive responses. Tellervo uses a style of HTTP+POX (Plain Old XML) to send and receive requests via an HTTP POST. In simple terms the Tellervo client sends an XML document that describes the request via POST to the Tellervo server. The server then reads the XML request, performs the request and then compiles the information that has been requested, finally returning the information to the client as another XML document. The syntax of the XML document containing the request and response is determined by the Tellervo XML schema and makes heavy use of the TRiDaS XML schema for describing dendrochronological entities.

The Tellervo schema is the final authority on what is allowed in a request or response so if you find a conflict between this documentation and the schema then it is most likely because the documentation is out-of-date, or it's simply incorrect. As you become familiar with the schema you'll probably find it easier to refer to it rather than this documentation to understand what is expected.

> 💡 One of the best ways of understanding the structure of requests and reponses is to use the XML monitor in the Tellervo Help menu to view the documents being sent and received by the Tellervo client.

18.1 Basics of sending requests

The webservice accepts requests via the POST mechanism of HTTP. The simplest way to understand this is to think of this as a standard webpage. If you add the following HTML code to a webpage it will give you a simple form. If you type your XML request document into the form and click submit the Tellervo server will respond with an XML document containing the answer to your query.

```
1  <form method="post" action="http://name.of.your.server/tellervo/">
2    <textarea name="xmlrequest"/>
3    <input type="submit" value="submit" >
4  </form>
```

A page with just such a form is available in the root of your Tellervo server installation as the page post.php.

When the response is viewed in a web browser, it will be rendered using a style sheet, but if you view the source code, you'll see the underlying XML document. A simple HTML form like this can be handy for helping you send some rough requests to the server to help you understand how things work.

The Tellervo client (or any other client that wants to talk to the Tellervo server) does exactly this. It sends XML formatted request documents to the server as POST requests, and the server returns an XML document containing the reply. The client then reads the XML reply and displays the result to the user in any way it sees fit.

18.2   Standard request/response

The request XML document you send the server needs to validate against the Tellervo XML schema (available in the schema folder of your Tellervo server or in the /src/main/resources/schemas/ folder in the Subversion repository. The schema is a detailed representation of what tags are allowed, when they are obligatory and what possible values they can contain. We strongly suggest getting hold of an XML validation tool to help you check that the requests you send the server are valid. The server will do the same and respond with an error message if it is not valid, but the textual error is a lot harder to understand than the graphic display of a desktop validation tool.

The general layout of the request file is as follows:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tellervo xmlns="http://www.tellervo.org/schema/1.0" xmlns:gml="http://www.opengis.net/gml"
       xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:tridas="http://www.tridas.org/1.2.2">
3    <request type="">
4      ...
5      ...
6    </request>
7  </tellervo>
```

- ▶ **Line 1** contains the XML declaration and tells the server that we're using UTF-8 character encoding. This is the only encoding currently supported.

- ▶ **Line 2** starts the root tag and defines the namespaces used by Tellervo. In this example the default namespace is the Tellervo schema itself, but it also refers to the TRiDaS, GML and XLink namespaces that Tellervo also makes use of. See section 18.2.1 for more details.

- ▶ **Line 3** begins the request tag which contains the request itself.

When you send such a request XML document to the Tellervo server the typical response returned is structured as follows:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tellervo xmlns="http://www.tellervo.org/schema/1.0" xmlns:gml="http://www.opengis.net/gml"
       xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:tridas="http://www.tridas.org/1.2.2">
3    <header>
4      ...
5    </header>
6    <content>
7      ...
8    </content>
9  </tellervo>
```

The header contains standard information about the request and also includes error and warning messages. The content tag contains the actual data being returned.

18.2.1   Namespaces

For newcomers to XML, namespaces and their definition can be quite confusing. Namespaces are used in XML to enable us to incorporate multiple schemas within the same XML file.

Schema	Namespace
Tellervo	http://www.tellervo.org/schema/1.0
TRiDaS	http://www.tridas.org/1.2.2
GML	http://www.opengis.net/gml
XLink	http://www.w3.org/1999/xlink

Table 18.1: The namespaces used in the Tellervo schema

In the case of Tellervo, that means we can use entities defined in both TRiDaS (to describe dendro data) as well as for instance GML (to describe location information). Rather than reinvent the wheel, we use established standards like GML so that we can leave the experts in each field to handle their own datastands. We need a method of clarifying which schema each entity in the XML file is referring to and for that we use namespaces.

Namespaces are typically defined in the root tag of the XML file using the attribute 'xmlns'. Multiple namespaces can be described by adding prefix definitions. For example xmlns: tridas ='http://www.tridas.org/1.2.2' means that any tag prefixed by 'tridas:' in the XML document refers to an entry in the TRiDaS schema. Namespaces are URIs and are typically URLs. There is quite often documentation about the schema at the namespace URL, but this is not necessarily the case. Information is certainly not retrieved from this location. The namespace just needs to be unique and must match what is defined by the schema. The prefix itself though can be whatever you want. If you have just a few prefixes, then it is common to just use a single character prefix, perhaps 't' for TRiDaS to keep the file small. Some people prefer to be more explict though and using longer prefixes to ensure there are no confusions. Table 18.1 lists the namespaces used by Tellervo.

If the majority of your XML file contains tags from one namespace, with just a handful of tags from another, you may prefer to define a default namespace. In this case all tags from the default namespace do not require prefixes. The following namespace declarations are therefore all valid and equivalent:

▶ <tellervo xmlns="http://www.tellervo.org/schema/1.0" xmlns:gml="http://www.opengis.net/gml" xmlns: xlink ="http://www.w3.org/1999/xlink" xmlns:tridas="http://www.tridas.org/1.2.2">

▶ <t: tellervo xmlns:t="http://www.tellervo.org/schema/1.0" xmlns:gml="http://www.opengis.net/gml" xmlns: xlink ="http://www.w3.org/1999/xlink" xmlns:tridas="http://www.tridas.org/1.2.2">

▶ <tellervo xmlns="http://www.tellervo.org/schema/1.0" xmlns:g="http://www.opengis.net/gml" xmlns:x="http://www.w3.org/1999/xlink" xmlns:t="http://www.tridas.org/1.2.2">

18.2.2 Errors and warnings

The webservice uses two tags within the header section of the return document to inform about errors and warnings. These tags are the status and message tags. The status tag can be set to: OK; Notice; Warning; or Error.

When there is an error, the message tag includes a code attribute along with the actual error message. The list of possible error codes is provided in appendix Y. An example of a typical error response is shown below:

```
1  <?xml version ="1.0" encoding="UTF−8"?>
2  <c: tellervo xmlns:c="http ://www. tellervo .org/schema/1.0">
3    <c: header>
4      <c: securityUser id="1" username="admin" firstName="Admin" lastName="user" />
5      <c: webserviceVersion >1.1.0</c: webserviceVersion >
6      <c: clientVersion >Tellervo WSI null ( httpcore 4.1; HttpClient 4.1.1; ts null .${
           revisionnum })</c. clientVersion >
7      <c: requestDate >2012−04−02T15:01:59 −04:00 </c: requestDate >
8      <c: queryTime unit="seconds">0.07</c: queryTime>
9      <c: requestUrl >/tellervo/</c: requestUrl >
10     <c: requestType>delete </c: requestType>
11     <c: status>Error</c: status>
```

```
12        <c: message  code="907">Foreign  key  violation.   You must  delete  all  entities  associated
              with  an  object  before  deleting  the  object  itself.</c:message>
13      </c: header>
14  </c: tellervo >
```

18.3   Authentication requests

There are two methods for authenticating yourself against the Tellervo server: plain and secure. We *strongly*
recommend you use the secure method as the user name and password are sent in plain text over the internet
when using the plain method. This goes against so much of the hard work we've put in to making the system
secure. It is quite likely that we will disable the plain authentication method by default in the future. For now it
will be left in place as it makes testing new clients much easier.

For further details and discussion about the authentication design please see section 19.1, page 113.

18.3.1   Plain authentication

If you *still* want to go ahead and use plain authentication despite all the risks, then this is how you do it.

```
1  <?xml  version ="1.0"  encoding="UTF−8"?>
2  <tellervo  xmlns="http ://www. tellervo . org/schema /1.0"  xmlns: tridas="http ://www. tridas . org
       /1.3">
3     <request  type=" plainlogin">
4         <authenticate  username=" yourusername"  password="yourpassword" />
5     </request >
6  </ tellervo >
```

18.3.2   Secure authentication

Although this is much more secure, it is also somewhat more complicated because it involves a challenge and
response scheme using cryptographic nonces and hashes.

The first step is for the client to request a nonce from the server. This is done as follows:

```
1  <?xml  version ="1.0"  encoding="UTF−8"?>
2  <c: tellervo  xmlns:c="http ://www. tellervo . org/schema /1.0"  xmlns:gml="http ://www. opengis . net/
       gml"  xmlns: xlink="http ://www.w3. org/1999/ xlink"  xmlns: tridas="http ://www. tridas . org
       /1.2.2">
3     <c: request  type=" nonce" />
4  </c: tellervo >
```

The returned document will include a nonce header tag like this: <c:nonce seq="176378">97566d4d2e8b8c5696b6667fef8429f5
</c:nonce>. Armed with your nonce you can then send a request to log in securely as follows:

```
1  <?xml  version ="1.0"  encoding="UTF−8"?>
2  <c: tellervo  xmlns:c="http ://www. tellervo . org/schema /1.0"  xmlns:gml="http ://www. opengis . net/
       gml"  xmlns: xlink="http ://www.w3. org/1999/ xlink"  xmlns: tridas="http ://www. tridas . org
       /1.2.2">
3     <c: request  type=" securelogin">
4       <c: authenticate  username="admin"  cnonce="3f975c569f978731e570"  snonce="97566
           d4d2e8b8c5696b6667fef8429f5"  hash="d315ec50f7f1809492d5ef132ad4aa06"  seq="176378" />
5     </c: request >
6  </c: tellervo >
```

The authenticate attributes are filled as follows. The username is simply the username for a user with permission to access the Tellervo server. The cnonce (client nonce) is a random string of your choosing that is used in the hash. The snonce (server nonce) is the nonce you've just obtained from the server. The seq (sequence) is the value also obtained from the server. Finally the hash is an MD5 hash of "username:md5hashofpassword:snonce:cnonce". If all is well the server will respond with the details of the person that is logging in.

18.3.3 Cookies and sessions

The webservice uses a session cookie so that the user doesn't need to authenticate with each request. The cookie lasts for up to 30 minutes of inactivity, after which point the server will request the user to re-authenticate before it will serve a request. Any client attempting to access the Tellervo server will therefore need to handle cookies and be ready to respond to requests to re-authenticate.

If a session does time out, a request to the server will result in an response with the header containing an error status with code '102' (see section 18.2.2 and appendix Y) and the nonce tag ready for the client to re-authenticate.

18.3.4 Logout

The session cookie will ensure that a user is logged out after a period of inactivity, but if you want to force a logout you can simply use the following type of request:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <c:tellervo xmlns:c="http://www.tellervo.org/schema/1.0">
3    <c:request type="logout">
4  </c:tellervo>
```

18.4 Reading records

The method for reading records from the Tellervo database is largely the same for any of the types of data the server handles. The basic template for a read request is as follows:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <c:tellervo xmlns:c="http://www.tellervo.org/schema/1.0">
3    <c:request type="read" format=" ">
4      <c:entity type=" " id=" " />
5    </c:request>
6  </c:tellervo>
```

The entity type is one of: project; object; element; sample; radius; measurementSeries; derivedSeries; box; securityUser; or securityGroup. The id attribute should be the database identifier for the entity you would like to read. In Tellervo these are typically a UUID like this: 339d8ea6-7448-11e1-ad85-9b6d022add7a. The format attribute should be one of: minimal; summary; standard; or comprehensive depending on how much detail you require about the entity. Keep in mind that a comprehensive request is likely to take much longer to fulfill than a minimal request so it's best for your user, if you use the simplest request that fulfills your need.

18.5 Deleting records

The method for deleting records in Tellervo is very similar to reading records. You simply use the request type 'delete' and specify the entity type and id. The basic template for a delete request looks like this:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <c:tellervo xmlns:c="http://www.tellervo.org/schema/1.0">
3    <c:request type="delete">
4      <c:entity type=" " id=" " />
5    </c:request>
6  </c:tellervo>
```

Cascading deletes are not permitted in Tellervom therefore only entities that are not used by other entities in the database can be deleted. For example, you cannot delete an object which has elements associated, you would need to delete in the elements first. Attempts to delete an entity which still has associated records will cause the webservice to return a '907-Foreign key violation' error.

18.6 Creating records

New records are are typically created by passing a TRiDaS representation of the entity you'd like to create inside a 'create' request. The basic template of a create request looks like this:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tellervo xmlns="http://www.tellervo.org/schema/1.0" xmlns:tridas="http://www.tridas.org
      /1.2.2">
3    <request type="create" parentEntityID=" ">
4      <tridas:sample>
5        ...
6      </tridas:sample>
7    </request>
8  </tellervo>
```

The request tag should include the parentEntityID attribute with the ID of the parent entity in the database e.g. the TRiDaS element to which a sample belongs for instance. Please note that there are a number of mandatory fields for each TRiDaS entity. These must be populated otherwise the webservice will return a '902-Missing user parameter' error.

In addition to handling the standard TRiDaS project, object, element, sample, radius, measurementSeries, derivedSeries entities, the Tellervo webservice can also create some Tellervo specific records. These are securityUser, securityGroup, box and permission. How to create and alter permissions records is described in more detail in section 18.8 as they are more complicated, but securityUser, securityGroup and boxes are handled in the same way as TRiDaS entities.

18.7 Updating records

Updating existing records in Tellervo is done in much the same way as creating records. The basic template is as follows:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <c:tellervo xmlns:c="http://www.tellervo.org/schema/1.0" xmlns:tridas="http://www.tridas.org
      /1.2.2">
3    <c:request type="update">
4      <tridas:element>
5        ...
6      </tridas:element>
7    </c:request>
8  </c:tellervo>
```

The main difference is that you *must* specify the TRiDaS < identifier > tag containing the ID of the record you are trying to update.

18.8   Reading and setting permissions

```
1  <request type="create">
2    <permission>
3        <permissionToCreate>true</permissionToCreate>
4        <permissionToRead>true</permissionToRead>
5        <permissionToUpdate>true</permissionToUpdate>
6        <permissionToDelete>true</permissionToDelete>
7        <entity type="object" id="760a19e2-229c-11e1-8756-03b2aff2fe33"/>
8        <securityGroup id="3"/>
9    </permission>
10 </request>
```

```
1  <request type="read">
2    <permission>
3        <entity type="object" id="760a19e2-229c-11e1-8756-03b2aff2fe33"/>
4        <securityGroup id="3"/>
5    </permission>
6  </request>
```

```
1  <request type="update">
2    <permission>
3        <permissionToCreate>false</permissionToCreate>
4        <permissionToRead>false</permissionToRead>
5        <permissionToUpdate>false</permissionToUpdate>
6        <permissionToDelete>false</permissionToDelete>
7        <entity type="object" id="136a70a6-566b-546b-a3ae-c48cb046e4cd"/>
8        <securityGroup id="1"/>
9        <securityGroup id="3"/>
10   </permission>
11 </request>
```

```
1  <request type="delete">
2    <permission>
3        <entity type="object" id="136a70a6-566b-546b-a3ae-c48cb046e4cd"/>
4        <securityGroup id="1"/>
5    </permission>
6  </request>
```

Systems architecture

The centralised nature of the Cornell Tree-Ring Lab data required a server-client architecture of some type. In Corina this was achieved simply by having users save their data in a network folder stored on a central server. Whilst this method was adequate, it has many data storage issues that can be largely solved by moving the data storage infrastructure to a relational database management system.

Although it would be possible (and arguably simpler) to have refactored Corina to talk directly to one central database server it was decided to go a step further and implement a Web Services orientated server-client architecture for Tellervo.

A web services approach decouples the desktop client from the server so that the server can work on its own.

19.1 Authentication design

The authentication mechanism is loosely based around http digest authentication and uses a challenge and response scheme. This makes use of cryptographic hashes (a relatively short digital fingerprint of some data but which cannot be decompiled to retrieve the original data) and nonces (a pseudo-random string used just once). All hashes used in the Tellervo webservice use the MD5 algorithm. Whilst an MD5 hash of a short phrase can be compromised, the length and randomness of the original data means that using current cracking techniques would require a very substantial amount of processing power e.g. supercomputer or large botnet. Flaws in the MD5 hash are also mitigated by the time-sensitive nature of the Tellervo nonce, meaning that any attack would need to be successful within a 2 minute window. New weaknesses in security are, however, revealed on a fairly regular basis so the authentication architecture will be periodically reviewed to ensure that it still meets our needs.

The first time a client attempts to retrieve data from the webservice (or when the client's credentials are incorrect or have expired) the following events occur:

- ▶ Server returns an message requesting authentication. This message includes a nonce (a hash of the current date and time to the nearest minute) which we will call 'server nonce'.
- ▶ The client creates a second nonce (client nonce) which is a random hash of it's choosing, and a response which is a hash of "username:hashofpassword:servernonce:clientnonc". It sends this response, along with the username and client nonce back to the server but does not send the original server nonce.
- ▶ The server computes the same "username:hashofpassword:servernonce:clientnonce" hash using the information it has stored in the database. As the server nonce is constant for a minute the two response should match. If not the server recomputes the server nonce for one minute ago and tries again. This ensures that the server nonce sent to the client is valid for between 1 and 2 minutes.
- ▶ Once the server authenticates the user a session cookie is sent to the client. On subsequent requests the server recognises the session id and doesn't request authentication again.

As the user's password is hashed at all points, even if the communication is hijacked the attacker will not be able to derive the users password. The user's password is also stored in hash form within the database. This also means that system administrators do not have access to the passwords either.

The use of the server nonce within the response means that it will only be valid for a maximum of two minutes. This minimizes the possibility of a replay attack.

19.2   Database permissions design

The database has a user and group based security scheme at three TRiDaS levels: object, element and series. A user can be a member of one or more groups, and groups can be members of zero or more other groups. The current implementation allows for one nested level of groups within groups however this could be extended if required. Security is set on a group-by-group basis rather than on a single user to ensure ease of management.

There are five types of permissions granted: create, read, update, delete and no permission. Each permission is independent of each other with the exception of 'no permission' which overrides all other permissions.

A group can be assigned one or more of the permissions types to any of the sites, trees or measurements in the database. Intermediate objects such as subsites, specimens and radii inherit permissions from their parent object. For instance if a group has permission to read a site then it will have permission to read all subsites from that site.

It is envisaged that most of the time, permissions will be set on a site-by-site basis. It will not be necessary to explicitly assign permissions to trees and measurements as all permissions will be inherited. So assuming that no permissions are set on a tree for a particular group, the permissions for the tree will be derived from the site from which the tree was found. If, however, permissions are assigned to the tree, then these will override those of the site. In this way it will be possible to allow a group to read the data from one particular tree from a site in which there otherwise do not have permission to access.

Privileges are cumulative. This means that if a user is a member of multiple groups then they will gain all the privileges assigned to those groups. If one of the groups that the user is a member of has 'no privileges' set on an object it will however override all other privileges. Therefore if a user is a member of groups A and B, and group A has read privilege and group B has 'no privilege' then the user will not be able to access the record.

A special 'admin group' has been created into which only the most trusted users are placed. Members of the admin group automatically gain full privileges on all data within the database. They also have permission to perform a number of administrative tasks that standard users are insulated from.

19.3   Universally Unique Identifiers

All entities in the Tellervo database have a primary key based on the Universally Unique Identifier (UUID) concept. This is a randomly created 128-bit number which due to the astronomically large number of possibilities (3×1038) means that it is guaranteed to be unique across all installations of Tellervo. This code is typically represented by 32 hexadecimal digits and 4 hyphens like this: 550e8400-e29b-41d4-a716-446655440000.

19.4   Barcode specifications

Barcodes in Tellervo are based on the UUID primary keys of database entities. Because they are used for different entities in Tellervo (boxes, samples and series) it was also necessary to incorporate a method for determining what type of entity a barcode represents. This is done by appending a single character and a colon to the beginning of the UUID: 'B:' for box; 'S:' for sample; 'Z:' for series.

The barcodes in Tellervo use the Code 128 scheme. This symbology was chosen as it allows the encoding of alphanumeric characters in a high-density label and can be read by all popular barcode scanners. While it would have been possible to create a barcode of plain UUIDs, the 36 (or even 32) characters would result in a barcode wider than many scanners could read. Most scanners on the market have a maximum scan width of at least 80mm, so this was used as the baseline to work to.

To make the barcodes less than 80mm, the UUID (with prepended entity type character code) are Base64 encoded. For example the series with UUID 3a8f4336-d17d-11df-abde-c75e325aebae would be encoded from Z:3a8f4336-d17d-11df-abde-c75e325aebae to become: Wjo6j0M20X0R36vex14yWuuu

Tellervo Database

The database behind Tellervo is run on the popular open source relational database management system, PostgreSQL (Postgres).

20.1 Spatial extension

Tellervo uses the PostGIS extension to Postgres to store and query spatial data within the database. Rather than storing coordinate axis in separate fields, a single specialist 'geometry' field type is used.

20.2 CPGDB functions

The Tellervo Postgresql Database (CPGDB) functions are a set of functions for searching, processing, and manipulating the data in the postgresql database. All functions are in thecpgdbschema, to allow for easy development alongside the database without modifying the database or its structure.

Thus, to execute a cpgdb function, you must preface the function name with cpgdb, e.g.:

```
SELECT * FROM cpgdb.GetVMeasurementResult('xxxx');
```

GetVMeasurementResultID – This function populates the tblVMeasurementResult and tblVMeasurementReadingResult tables, returning a single varchar which contains the tblVMeasurementResult ID. You probably want to use GetVMeasurementResult instead.

GetVMeasurementResult – This function returns a table row from tblVMeasurementResult which has been populated with information from the provided VMeasurement ID.

GetVMeasurementReadingResult – This function is provided as a convenience method. It requires a VMeasurementResultID obtained from one of the above two functions. Data is returned sorted by year, ascending.

FindVMChildren – This function reverse traverses the database and gives a list of derived VMeasurements. This is most useful when given the ID of a direct VMeasurement, to find any sums, redates, or others based upon it.

FindVMParents – This function traverses the database and gives a list of parents VMeasurements. This is most useful when given the ID of a Sum, Redate, or Index, to find which VMeasurements it was based on.

FindChildrenOf – This function returns a list of all VMeasurements derived from something. Given 'tree' and '16', for instance, it will find all VMeasurements derived from Tree ID 16. e.g.:

> ✎   `select * from cpgdb.findchildrenof('specimen', 1);`

> ⚠ *Does not traverse through object relationships. Will only return children of a single particular object. See FindChildrenOfObjectAncestor()*

FindChildrenOfObjectAncestor – This function returns a list of all VMeasurements derived from a particular object and its descendants. The output is the same format as FindChildrenOf.

FindObjectTopLevelAncestor – Returns the toplevel ancestor object of a given object. Will return the given object if it has no toplevel ancestor.

FindObjectAncestors – Returns the ancestor objects of a given object, guaranteed from bottom to top. Can return an empty set.

FindObjectDescendants – Returns the descendant objects of a given object using a depth-first traversal. Can return an empty set.

FindObjectDescendantsByCode – Convenience wrapper around FindObjectDescendants which takes an object code rather than ID.

FindObjectsAndDescendantsWhere – Returns the objects and that match a given WHERE clause and their descendants. Does not return duplicates.

FindElementObjectAncestors – Returns the ancestry tree of objects, given an element id. Really just a helper function for FindObjectAncestors().

GetGroupMembership – This function returns a unique list of all the groups the specified user is a member of.

GetGroupMembershipArray – This function returns an integer array of all the securityGroupIDs the specified user is a member of.

GetUserPermissions – Returns an array of the permissions the specified user has for a particular object ID. The function backtracks *tree* ➜ *site* ➜ *default* and *site* ➜ *default* if no explicit permissions are found. If 'No permission' is returned it is the only member of the array. If a user is a member of group 1 (admin), they automatically get all permissions.

MergeObjects – This function merges two objects together. The first object is taken as the basis with all its fields maintained unchanged. Any fields that are different in the second object are noted in the comments field for checking later. If a field is null in the first object but present in the second, then this value is used. The function cascades through the entity hierarchy merge subordinate entities where required using the other merge functions.

MergeElements – As for MergeObjects but for elements.

MergeSamples – As for MergeObjects but for samples.

MergeRadii – As for MergeObjects but for radii.

20.3   Complex database functions

Beyond the standard database functions discussed in section 20.2, the Tellervo database uses PLJava perform for more complex tasks. PLJava means that we can leverage the full power of Java to perform calculations and analyses on the database.

During the standard build process a small jar called `tellervo-pljava.jar` is created. This contains classes from the packages: `org/tellervo/cpgdb/**` and `|org/tellervo/indexing/**|`. This jar is stored within the sqlj schema of the Tellervo database and the classes called as required by Postgres functions.

So unless you make changes to files within these packages, you have no need to worry about PLJava. If you do need to make changes, then you will need to add the new jar to the database. To do this you use the pljava functions within Postgres.

The main calls you need to make are:

```
    SELECT sqlj.replace_jar('file:///usr/share/tellervo-server/tellervo-pljava.jar',
'tellervo_jar', false); SELECT sqlj.set_classpath('cpgdb', 'tellervo_jar');
```

The first replaces the existing jar with the one specified. PLJava requires a name for the jar to use within the database and so we use 'tellervo_jar'. The second command sets the classpath for the specified schema (the first argument) to the contents of the specified jar (the second argument). Once you have done this you should be able to call your Java functions from within Postgres.

Please note that if you add dependencies to your classes that are not provided by the standard Java virtual machine these must also be added to the tellervo-pljava.jar. You can get rather cryptic `NoClassDefFoundError` errors that do not necessarily name the additional dependency, but the class from which it is called.

Part III

Appendices

Belfast Apple

Format name	Belfast Apple
Other name(s)	None known
Type	Text file
Extension(s)	Various (typically txt and dat)
Read/write support	Read and write
Reference implementation	No original software is known to exist so TRiCYCLE is proposed as the reference implementation
Data / metadata	Data only with comment
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	John Pilcher

A.1   Description

Belfast Apple is a simple text file format (see also Belfast Archive) originating from the Queens University Belfast lab and originally designed for use on an Apple II computer. This format is not known to be actively used but a large amount of data (especially at Belfast) is archived in this format.

- ▶ Line 1 - name of the site or object the data refers to.
- ▶ Line 2 - identifier for the sample the data refers to.
- ▶ Line 3 - number of data values in the file
- ▶ Lines 4+ - line feed delimited data values as integers in 1/100th mm
- ▶ Final line contains a comment typically starting with 'COMMENT -'

A.2   Example file

```
1   EXAMPLE SITE
2   A1805
3   106
4   188
5   165
6   184
7   112
8   103
9   111
10  239
11  226
12  132
13  143
14  146
15  140
16  100
17  176
18  139
19  124
20  115
21  78
22  80
23  156
24  75
25  110
26  80
27  130
28  83
29  157
30  99
31  115
32  102
33  110
34  108
35  87
36  135
37  107
38  96
39  70
40  128
41  119
42  86
43  101
44  106
45  129
46  88
47  101
48  151
49  106
50  97
51  110
52  97
53  91
54  93
55  100
56  124
57  99
58  134
59  125
60  105
61  96
62  COMMENT −    PB 15−NOV−99
```

Belfast Archive

Format name	Belfast Archive
Other name(s)	None known
Type	Text file
Extension(s)	Various (typically arx, txt and dat)
Read/write support	Read only
Reference implementation	No original software is known to exist so TRiCYCLE is proposed as the reference implementation
Data / metadata	Data with limited metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Martin Munro

B.1 Description

Belfast Archive is a simple text file format based on the original Belfast Apple format at the Queens University Belfast lab. It shares the same features as Belfast Apple but with the addition of a number of metadata fields at the end of the file.

- ► Line 1 - name of the site or object the data refers to.
- ► Line 2 - identifier for the sample the data refers to.
- ► Line 3 - number of data values in the file
- ► Lines 4+ - line feed delimited data values as integers in 1/100th mm
- ► The lines "[[ARCHIVE]]" and "[[END OF TEXT]]" denote the start and finish of the metadata section

The metadata section contains the following lines:

- ► Line 1 - start year as an integer.
- ► Line 2 - unknown
- ► Line 3 - Double representing the resolution of data values e.g. .1= 1/10ths mm, .01 = 1/100th mm, .001 = microns etc
- ► Line 4 - unknown
- ► Line 5 - unknown
- ► Line 6 - unknown

- ▶ Line 7 - title of the data series
- ▶ Line 8 - unknown
- ▶ Line 9 - unknown

B.2   Example file

```
1   EXAMPLE SITE
2   1
3   176
4   342
5   338
6   334
7   409
8   362
9   308
10  360
11  264
12  325
13  318
14  51
15  48
16  47
17  60
18  49
19  48
20  "[[ARCHIVE]]"
21  1277
22  9177
23  .01
24  1.035795
25  0.212144
26  BOB 25/03/95
27  EXAMPLE SITE #01
28  Pith F Sap 32
29  ""
30  "[[ END OF TEXT ]]"
```

Besançon

Format name	Besançon
Other name(s)	SYLPHE
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	Not known
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Georges Lambert

C.1   Description

The Besançon format is most commonly used in a number of French laboratories. The format allows for multiple series in the same file. Each series (or element block in Lambert's notation) is made up of a header line, optional metadata and a data block each of which are delimited by a line feed.

The header line begins with a dot character, then one or more spaces, then an element name (without spaces) followed by a space and any number of ignored characters.

The metadata fields are space or line feed delimited. Each field is recorded using a key of three letters. The format allows for the full spelling out of the field if preferred, but it is the first three letters that are read by software so LON is the same as LONGEUR. Some fields are 'unimodal' in that their presence is all that is required e.g. CAM means that cambium was observed. Other fields are 'bimodal' which means they require a value to be associated with them. In this case the field key is followed by a space and then an integer or string value e.g. POS 1950. The accepted metadata fields are as follows:

LON Number of data values

POS The temporary first ring date given relatively to a group

ORI The year for the first ring

TER The year for the last ring. Should be the same as ORI + LON

MOE Pith present

CAM Cambium present

AUB Number of the first sapwood ring

All other information in the metadata block should be ignored. This feature is often used to allow the inclusion of multi-line comments.

The data block begins with the marker line VAL (like metadata keys, subsequent characters are ignored so sometimes the rest of this line is used for comments). Subsequent lines contain integer values delimited by a space or line feed. Missing rings are marked with a comma character and the end of the data is marked with a semicolon.

C.2 Additional information

▶ There is nothing in the specification to say what precision the data values should be in. Following conversations with users it appears that Besançon files are mostly 1/100th mm but this is not always the case. Some files include a Précision field, but this is not documented or standardised.
▶ There are a number of additional fields that are commonly used but which do not appear in the format specification. These are also supported by the DendroFileIOLib
ESP Species
ECO Bark present

C.3 Example file

```
1   .  abc22/43
2   Lon  129
3   Esp  quercus  sp    Nat  lambris
4   Precision  1/100
5   Moelle  non  presente
6   Aub  0
7   valeurs
8     149   119   156   146   170   187   197   146   191   177
9     137   108   160   108   120   177   136   174   190   109
10    189   176   170   162   114   126   133   152   146   127
11    119   131   146   133   147    82    57    77    77    82
12     96    49    97    76    88    82    72    83    81    90
13     85    87    78   104   111   132   141   105   104   120
14    111   121   115    89    94    88    90   115   111   106
15    107   120    80    92    98    84    97    82   100    86
16     99    65    85   113    90    82    57    57    99    94
17     95   105   120   110    93    96   131   133   123   122
18    113   119    95   127    88   104     ,     ,     ,     ,
19      ,     ,     ,     ,     ,     ,     ,     ,     ;
```

CATRAS

Format name	CATRAS
Other name(s)	None known
Type	Binary
Extension(s)	cat
Read/write support	Read and write
Reference implementation	CATRAS
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	Roland Aniol

D.1 Background

The CATRAS format (Aniol, 1983) is the only known binary dendro data format. As such it can't be read by a simple text editor, and can't be imported by spreadsheet or database programs. The format was designed by Roland Aniol for use in his program of the same name. The binary nature of the format means the files are typically much smaller than text files containing similar data. The closed nature of the format originally meant that users were tied to the application. The fact that users can't manually edit the file means that the validity of files is not a problem like it is with most other dendro formats.

The format was originally decoded in the early 1990's and permission was granted by Aniol for a converter to be included in Henri Grissino-Mayer's CONVERT5 application. Subsequently others have independently released application and code that can read CATRAS files to a greater or lesser extent.

Following its original release in 1983, CATRAS was updated several times, the most recent version (v4.42) was released in 2010. The code in DendroFileIOLib is based in part on Matlab, Fortran and C code of Ronald Visser, Henri Grissino-Mayer and Ian Tyers.

D.2   Reading byte code

Reading byte code is more complicated than reading text files. Each byte is 8-bits and therefore can represent up to 256 values. Depending on the type of information each byte contains, the bytes are interpreted in one of four ways:

D.2.1   Strings

Some of the bytes in CATRAS files contain character information. In this case each byte represents a letter. In java an array of bytes can be directly decoded into a string.

D.2.2   Integers

As a byte can only represent 256 values, whenever an integer is required it is stored as a byte pair. Each byte pair consists of a least significant byte (LSB) and a most significant byte (MSB). The order that they appear in files typically varies between platforms and is known as 'endianness'. As CATRAS solely runs on Microsoft (x86) processors we can safely assume that all CATRAS files will be using little-endian (i.e. LSB MSB). The counting in a byte pair therefore works as follows:

Value	LSB	MSB
1	1	0
2	2	0
3	3	0
. . .	. . .	. . .
256	0	1
257	1	1
258	2	1
. . .	. . .	. . .

A byte pair can therefore store 256x256=65536 values (more than enough for most number fields).

D.2.3   Real numbers

Statistical values–such as arithmetic mean, standard deviation, first-order autocorrelation, and mean sensitivity–are given for all the ring widths and optionally for the ring widths in a restricted part of the series. The real numbers are given in standard format defined by the IEEE 754 Standard for Floating-Point Arithmetic.

D.2.4   Categories

Categories are typically recorded as single bytes as most categories have just a few possible values. They can therefore be conceptualized as being integers where 0=first option, 1=second option etc. The exception to this is for species because there are more than 256 species. In this case, a byte pair is used in exactly the same way as described for integers above. The only problem for species is that the codes are unique to each laboratory and refer to values enumerated in a separate '.wnm' file. Without this dictionary the species code is of little use.

D.2.5   Dates

The date of the creation of the series and the date of the last amendment to the series are stored as three single bytes each, one for day, one for month, and one for year. The year is stored with an offset of 1900. Therefore numbers from 1 to 100 belong to the 20th century (calendar year 1901 to 2000) and numbers from 101 to 200 belong to the 21th century (calendar year 2001 to 2100).

D.3   Metadata

The first 128 bytes contain the file header information and the remainder of the file contains the ring-width data and sample depth data (if series is a chronology). If a series is only partly suitable for further analysis then this indicated in bytes 49–52. The quality code at position 58 is an overall rating for the series. This helps to exclude poor series from analyses other than dating.

D.4   Data

The remaining bytes in the file contain the actual data values stored as integer byte pairs. All data are stored in multiples of 128 bytes. If the number of data bytes given in the header at position 45–46 is not a multiple of 128 the file is padded with extra bytes accordingly. Padded bytes should be ignored.

D.4.1   Ring widths

Ring widths are stored in hundredths of a millimetre in the same order as the tree had been grown. When working with archaeological or geological wood it might occur that a particular ring is damaged and therefore its width cannot be determined precisely. To indicate that fact and to exclude this particular ring from further calculations its measured width is stored negative. In the CATRAS program a negative ring width will be taken into account neither in the calculation of tree curves and chronologies nor in the statistics or in comparisons with other series.

D.4.2   Chronologies

Chronology files are indicated at position 84 in the file header and contain additional data in respect to raw data files. After the block of ring width data three additional data blocks follow. Firstly the number of ring widths averaged at a particular position follows (the sample depth). Then the number of series with the same trend between subsequent ring widths at a particular position follows. Then the number of series with the opposite trend between subsequent ring widths at a particular position follows. All data blocks are stored in multiples of 128 bytes. If the number of data bytes given in the header at position 45-46 is not a multiple of 128 each block is padded with extra bytes accordingly. Padded bytes should be ignored.

Bytes	Data type	Field	Description
1–32	C	Series name	
33–40	C	Series code.	Must be upper case and match file name.
41–44	C	File extension	
45–46	I	Series length	
47–48	I	Sapwood length	
49–50	I	First valid ring	Used if a portion of the series is unreliable
51–52	I	Last valid ring	Used if a portion of the series is unreliable
53	B	Scope	1=pith; 2=waldkante; 3=pith to waldkante; 4=bark; 5=pith to bark
54	B	State of last ring	0=last ring complete; 1=last ring only early wood
55–56	I	First ring	Calendar year of first ring: 0=not dated; ¡0=B.C.; ¿0=A.D.
57	B		Number of valid characters in series name
58	B	Quality code	0=not known; 1=very good . . . 5=uncertain
59–60	I	Species code	Requires an associated catras.wnm file
61–63	D	Creation date	DMY, Y offset 1900
64–66	D	Last updated	DMY, Y offset 1900
67	B	Real number format	normally 1=IEEE
68	B	Type of series	0=ring widths; 1=early wood widths; 2=late wood widths
69–81			Reserved
82	C	Special sources	A=averaged; D=digitized; E=extern; H=manual input
83	B	Protection	0=no protection; 1=not to be deleted; 2=not to be amended
84	B	File type	0=raw; 1=tree curve; 2=chronology
85–88	C	Creator	Initials of creator
Statistics for total series			
89–92	R		Arithmetic mean
93–95	R		Standard deviation
96–100	R		First-order autocorrelation
101–104	R		Mean sensitivity
105–106	I		Number of rings for mean
107–108	I		Number of rings for autocorrelation
Statistics for restricted part of series			
109–112	R		Arithmetic mean
113–116	R		Standard deviation
117–120	R		First-order autocorrelation
121–124	R		Mean sensitivity
125–126	I		Number of rings for mean
127–128	I		Number of rings for autocorrelation

Table D.1: Summary of the metadata portion of CATRAS files. Data types are: strings (C); integers (I); real numbers (R); binary categories (B); and dates (D). Bytes 89–128 contain descriptive statistics for the file. Bytes 89–108 concern the entire series, and bytes 109–128 a subset of the series where some poor quality data (defined in bytes 49–52) have been excluded.

Cracow Binary Format

Format name	Cracow Binary Format
Other name(s)	Cracow
Type	Binary file
Extension(s)	AVR and AVS
Read/write support	Read
Reference implementation	Unknown
Data / metadata	Data only
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	Unknown

E.1   Description

This is a binary format used primarily in Polish labs (e.g. AGH-UST in Cracow, Universities in Szczecin, Torun and Katowice). It is a simple data-only format with no support for calendars or dating. There are two types of file (AVR and AVS) but the format for them is identical, the only difference being that the AVR extension indicates the file is a measurement series file and AVS extension indicates a chronology file.

The file begins with a six character header:

1. Zero
2. Ring number where sapwood begins
3. Zero
4. Ring number where sapwood ends (also length of sample)
5. Zero
6. Zero

If there is no sapwood recorded, then bytes 2 and 4 will both be zero. The zero bytes at positions 1, 3, 5 and 6 can be used as a signature to check the file is in Cracow format.

Following the six header bytes, there comes the ring width values. Each ring value is stored in two bytes, the second containing the values from 0 to 99 and the first contains the values ¿100. For instance the ring value 345 would be stored in two values, 3 and 45.

Comma Separated Values

Format name	Comma Separated Values
Other name(s)	CSV
Type	Text file
Extension(s)	Various (typically txt or csv)
Read/write support	Read and write
Reference implementation	n/a
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	No
Original designer	n/a

F.1   Description

Comma separated values format is a simple text format for representing tabular data. It is not specific to dendrochronology data and is supported by most spreadsheet and database applications. Data is delimited into columns using a comma character to indicate cell boundaries.

Support for CSV files in TRiCYCLE is limited to a particular layout of data. The expected layout is the same as for Excel and ODF spreadsheet files:

- ▶ Row 1 - Header names for each column
- ▶ Column A - Year values
- ▶ Column B+ - One column for each series containing data values. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

F.2   Example file

```
 1   Year , MySample1 , MySample2
 2   500 ,0.33 ,
 3   501 ,0.26 ,0.26
 4   502 ,0.2 ,0.2
 5   503 ,0.14 ,0.14
 6   504 ,0.08 ,0.08
 7   505 ,0.02 ,0.02
 8   506 ,0.2 ,0.2
 9   507 ,0.14 ,0.14
10   508 ,0.08 ,0.08
11   509 ,0.2 ,
12   510 ,0.33 ,
13   511 ,0.08 ,
14   512 ,0.33 ,
15   513 ,0.22 ,
```

Corina Legacy

Format name	Corina Legacy
Other name(s)	Corina
Type	Text file
Extension(s)	Various including raw, rec, ind, cln, sum)
Read/write support	Read and write
Reference implementation	Corina
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	Yes
Multi series support	No
Original designer	Robert 'Mecki' Pohl

G.1   Description

The Corina Legacy format is the file format used by the Corina software prior to version 2, when it transferred to using TRiDaS. The format was originally designed for use with the MS-DOS version of Corina but was also used as the native file format in the later Java versions (up to and including v1.1).

A Corina file contains yearly data (ring-width and number of samples for that year), some fixed metadata, and optionally weiserjahre data and a listing of element samples (for summed samples).

The title comes first, on a line by itself, followed by a blank line. The title is repeated later, so this is only to make it easier for people or external programs to read the title.

The *metadata section* comes next. The syntax is `;TAG` value. Tags are all uppercase. Their order is fixed. Some values are terminated by a newline, others by the next semicolon. Valid tags, and their internal names are:

- ► ID - 8 character ID used when exporting to Tucson format
- ► NAME - Name of the series
- ► DATING - Either R (relative) or A (absolute)
- ► UNMEAS_PRE - Number of unmeasured rings towards the pith
- ► UNMEAS_POST - Number of unmeasured rings towards the bark
- ► FILENAME
- ► COMMENTS, COMMENTS2 etc - Free text comments
- ► TYPE - either C (core), H (charcoal) or S (section)

- ▶ SPECIES
- ▶ SAPWOOD - Count of sapwood rings
- ▶ PITH - either P (present), * (present but undateable), or N (absent)
- ▶ TERMINAL - either B (bark), W (waney edge), v (near edge), vv (unknown)
- ▶ CONTINUOUS - referring to the outer ring, either C (continuous), R (partially continuous) or N (not continuous)
- ▶ QUALITY - either + (one unmeasured ring), ++ (more than one unmeasured ring)
- ▶ FORMAT - either R (raw) or I (indexed)
- ▶ INDEX_TYPE - type of index used
- ▶ RECONCILED - Y or N indicating whether the series has been reconciled against another series

The *data section* comes next and this always starts with the line ;DATA and for reasons lost in time there are nine spaces afterwards.

Data lines come in pairs, the first line containing the year and data values, the second containing the sample depth/count for each value. For reasons unknown, the first and last data line pair have a slightly different syntax to the others.

- ▶ First data line begins with a space and an integer for the first year in the line. There then follows 9 spaces followed by the integer data value for the first ring. The remaining data values (often less than a full decades worth) on that line follow as integers left padded by spaces to take up 6 characters.
- ▶ The sample depth line that pairs with this follows next starting with 16 spaces, followed by the sample depth value enclosed in square brackets. The remaining sample depth values follow in square brackets left padding with spaces to take up 6 characters.
- ▶ Next comes the first normal data line. This begins with a space, followed by an integer year value. The data values follow as integers left padded by spaces to take up 6 characters. A data line has a decades worth of data values.
- ▶ Next comes the normal sample depth line. It begins with 7 spaces followed by each of the sample depth values enclosed in square brackets and left padded with spaces up to 6 characters.
- ▶ Data lines continue in pairs until the last line is reached. This is the same as a normal data line except it includes an extra data value 9990 as a stop marker. This data line may have less than a full decade of values.
- ▶ The final sample depth line is the same as normal except it is shifted left by 4 characters. A sample depth value is also included for the dummy 9990 stop marker year.

Following the data block there is a blank line and two option blocks of data that are only included if the file is a chronology file.

The next block of information in a chronology file is denoted by a line ;ELEMENTS. The following lines contain the file names of the data files that have contributed to the creation of the chronology.

Following this is an optional block denoted by the line ;weiserjahre followed by the weiserjahre data. Each weiserjahre data line begins with a space followed by a integer year value for the first year in the line. The weiserjahre value is left padded with spaces to fill 6 characters and the value itself is written as X/Y where X is the number of samples that show an upward trend in width; and Y is the number of samples that show a downward trend in width. The weiserjahre value is forward facing so the value for ring 1001 shows the trend between ring 1001 and 1002. There is therefore one less weiserjahre value in the final row than there are ring-widths.

The final line of Corina data files contains the author's name preceded by a tilde.

G.2 Example file

```
1   Trebenna, Byzantine Fortress, NW tower 1AB
2
3   ;ID 907010;NAME Trebenna, Byzantine Fortress, NW tower 1AB;DATING R;UNMEAS_PRE 1;UNMEAS_POST
        1
4   ;FILENAME G:\DATA\TRB\TRB1AB.SUM
5
6
7   ;TYPE S;SPECIES Juniperus sp.;FORMAT R;PITH +
8   ;TERMINAL vv;CONTINUOUS N;QUALITY +
9   ;RECONCILED Y
10  ;DATA
11   1001          125   219   207   139    62   107    29    91    65
12                 [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
13   1010    71   132    74   150    75   156   122    81    46    57
14          [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
15   1020   147    78    89   126    73   121    67    71    64   129
16          [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]
17   1030   149   155   122   126    53   136    90    65   100    67
18          [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [1]   [2]
19   1040    67   101   132   102    40    67    42    36    62    29
20          [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
21   1050    30    44    46    40    34    61    55    29    44    63
22          [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
23   1060    62    38    22    26    26    28    37    21    21    27
24          [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]
25   1070    17    18    50    21    33    12    17    16    27    20
26          [2]   [2]   [2]   [2]   [2]   [2]   [2]   [2]   [1]   [1]
27   1080    18    11     9     8  9990
28       [1]   [1]   [1]   [1]   [1]
29
30  ;ELEMENTS
31  G:\DATA\TRB\TRB1A.REC
32  G:\DATA\TRB\TRB1B.REC
33  ;weiserjahre
34   1001   1/0      0/1     0/1     0/1     1/0     0/1     1/0     0/1     1/0
35   1010   1/0      0/1     1/0     0/1     1/0     0/1     0/1     0/1     1/0     1/0
36   1020   0/1      1/0     1/0     0/1     1/0     0/1     1/0     0/1     1/0     1/0
37   1030   1/0      0/1     1/0     0/1     1/0     0/1     0/1     1/0     0/1     1/1
38   1040   2/0      2/0     0/2     0/2     2/0     0/2     0/2     2/0     0/2     2/0
39   1050   2/0      1/1     0/2     0/2     2/0     0/2     0/2     2/0     2/0     1/1
40   1060   0/2      0/2     2/0     1/1     2/0     2/0     0/2     1/1     2/0     0/2
41   1070   1/1      2/0     0/2     2/0     0/2     2/0     1/1     1/0     0/1     0/1
42   1080   0/1      0/1     0/1
43  ~ Unknown User
```

DendroDB

Format name	DendroDB
Other name(s)	None known
Type	Text file
Extension(s)	dat
Read/write support	Read only
Reference implementation	DendroDB website
Data / metadata	Data and some structured metadata
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Simon Brewer

H.1 Description

The DendroDB format is an export file format produced by the DendroDB website/database. There is no known software that can natively read DendroDB files so a 'writer' for this format has not been developed.

The format is self-explanatory, beginning with a copyright line, followed by 7 metadata lines, then the data itself. There are eight possible data variables: Total width; Earlywood width; Latewood width; Min. Density; Max. Density; Earlywood density; Latewood density; Average density. Ring width data is provided in microns but the units for density measurements are not document.

As of Feb 2011, the DendroDB database does not contain data prior to 441AD so handling of BC/AD transition has not been tested. The DendroDB web interface suggests that BC dates should be entered as negative integers, but it also allows request for data from year 0. This suggests the database uses an Astronomical calendar and this is how the DendroIOLib treats it.

H.2 Example file

```
1   Data downloaded from DendroDB. Please acknowledge authors
2   Site: Example site
3   Contact: A N Other
4   Species: Larix sibirica
5   Parameter: Latewood width
6   Latitude: 53.25
7   Longitude: 57.35
8   Elevation: 1670
9   Tree Core Year Latewood width
10  1 1 1648 16
11  1 1 1649 21
12  1 1 1650 8
13  1 1 1651 10
14  1 1 1652 6
15  1 1 1653 8
16  1 1 1654 11
17  1 1 1655 13
18  1 1 1656 9
19  1 1 1657 10
20  1 1 1658 10
21  1 1 1659 4
22  1 1 1660 5
23  1 1 1661 7
24  1 1 1662 4
25  1 1 1663 8
26  ...
```

Heidelberg

Format name	Heidelberg
Other name(s)	TSAP, FH
Type	Text file
Extension(s)	fh
Read/write support	Read and write
Reference implementation	TSAP-Win
Data / metadata	Data and extensible metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	Yes
Original designer	Frank Rinn

I.1 Description

The Heidelberg format (Rinn, 2008) is the native file format for Rinntech's TSAP-Win software. It supports metadata in the form of keyword-value pairs. There are more than 140 standard keywords specified in the documentation, but users can extend these with their own. This makes the format extremely flexible, but the absence of any checking of data types (strings, numbers categories etc) and no method of validation means that there can be problems interpreting metadata entries.

Heidelberg files can store one or more series in a single file. Each series is represented by a header and a data block.

The header block begins with a line HEADER:. This is followed by lines of metadata, with one field on each line, in the format keywords=value much like a standard Windows INI file. As mentioned previously there are a number of predefined keywords, all of which are outlined here:

- AcceptDate
- Age
- AutoCorrelation
- Bark
- BHD
- Bibliography
- Bibliography[n]
- BibliographyCount
- Bundle
- CardinalPoint
- ChronologyType
- ChronoMemberCount

- ▶ ChronoMemberKeycodes
- ▶ Circumference
- ▶ Client
- ▶ ClientNo
- ▶ Collector
- ▶ Comment
- ▶ Comment[n]
- ▶ CommentCount
- ▶ Continent
- ▶ CoreNo
- ▶ Country
- ▶ CreationDate
- ▶ DataFormat
- ▶ DataType
- ▶ DateBegin
- ▶ Dated
- ▶ DateEnd
- ▶ DateEndRel
- ▶ DateOfSampling
- ▶ DateRelBegin[n]
- ▶ DateRelEnd[n]
- ▶ DateRelReferenceKey[n]
- ▶ DateRelCount
- ▶ DeltaMissingRingsAfter
- ▶ DeltaMissingRingsBefore
- ▶ DeltaRingsFromSeedToPith
- ▶ Disk
- ▶ District
- ▶ EdgeInformation
- ▶ EffectiveAutoCorrelation
- ▶ EffectiveMean
- ▶ EffectiveMeanSensitivity
- ▶ EffectiveNORFAC
- ▶ Key
- ▶ EffectiveNORFM
- ▶ EffectiveStandardDeviation
- ▶ Eigenvalue
- ▶ Elevation
- ▶ EstimatedTimePeriod
- ▶ Exposition
- ▶ FieldNo
- ▶ FilmNo
- ▶ FirstMeasurementDate
- ▶ FirstMeasurementPersID
- ▶ FromSeedToDateBegin
- ▶ GlobalMathComment[n]
- ▶ GlobalMathCommentCount
- ▶ GraphParam
- ▶ Group
- ▶ HouseName
- ▶ HouseNo
- ▶ ImageCellRow

- ▶ ImageComment[n]
- ▶ ImageFile[n]
- ▶ ImageCount
- ▶ ImageFile
- ▶ Interpretation
- ▶ InvalidRingsAfter
- ▶ InvalidRingsBefore
- ▶ JuvenileWood
- ▶ KeyCode
- ▶ KeyNo
- ▶ LabotaryCode
- ▶ LastRevisionDate
- ▶ LastRevisionPersID
- ▶ Latitude
- ▶ LeaveLoss
- ▶ Length
- ▶ Location
- ▶ LocationCharacteristics
- ▶ Longitude
- ▶ MajorDimension
- ▶ MathComment
- ▶ MathComment[n]
- ▶ MathCommentCount
- ▶ MeanSensitivity
- ▶ MinorDimension
- ▶ MissingRingsAfter
- ▶ MissingRingsBefore
- ▶ NumberOfSamplesInChrono
- ▶ NumberOfTreesInChrono
- ▶ PersId
- ▶ Pith
- ▶ Project
- ▶ ProtectionCode
- ▶ Province
- ▶ QualityCode
- ▶ Radius
- ▶ RadiusNo
- ▶ RelGroundWaterLevel
- ▶ RingsFromSeedToPith
- ▶ SampleType
- ▶ SamplingHeight
- ▶ SamplingPoint
- ▶ SapWoodRings
- ▶ Sequence
- ▶ SeriesEnd
- ▶ SeriesStart
- ▶ SeriesType
- ▶ ShapeOfSample
- ▶ Site
- ▶ SiteCode
- ▶ SocialStand
- ▶ SoilType

- ▶ Species
- ▶ SpeciesName
- ▶ StandardDeviation
- ▶ State
- ▶ StemDiskNo
- ▶ Street
- ▶ Timber
- ▶ TimberHeight
- ▶ TimberType
- ▶ TimberWidth
- ▶ TotalAutoCorrelation
- ▶ TotalMean
- ▶ TotalMeanSensitivity
- ▶ TotalNORFAC

- ▶ TotalNORFM
- ▶ TotalStandardDeviation
- ▶ Town
- ▶ TownZipCode
- ▶ Tree
- ▶ TreeHeight
- ▶ TreeNo
- ▶ Unit
- ▶ UnmeasuredInnerRings
- ▶ UnmeasuredOuterRings
- ▶ WaldKante
- ▶ WoodMaterialType
- ▶ WorkTraces

The meaning of many of these keywords is fairly self-explanatory but others are a little more obscure. As there is no data typing or validation the format of the contents of these fields cannot be predicted. This is particularly a problem when trying to compare fields such as Latitude, Longitude and FirstMeasurementDate, but is especially a problem when comparing files produced in different labs.

The header section is followed by a data section denoted by a line containing the keyword DATA: followed by the type of data present which can be one of Tree; HalfChrono; Chrono; Single; Double; Quad. Tree, HalfChrono and Chrono are the original keywords supported by early versions of TSAP but these are now deprecated in preferences of the more generic Single, Double and Quad terms. The terms Single, Double and Quad are largely interchangeable with Tree, HalfChrono and Chrono respectively, but not completely. Double can refer to both Tree and HalfChrono format data. When the newer terms are used, the header keyword DataFormat is used to record whether the data is equivalent to Tree, HalfChrono or Chrono.

Single format - data is typically used for storing raw measurement series. Each data line contains 10 data values each being a left space padded integer taking up 6 characters. Any spare data values in the final data line are filled with zeros. Alternatively it appears that TSAP-Win also accepts this data section as single integer values one per line.

Double format - data is for storing data with sample depth information - typically chronologies. Like the single format section, data is stored as 10 integer values, each taking up 6 characters and left padded with spaces. The values are in pairs of ring-widths and sample depths, therefore five rings are stored per line.

Quad format - data is for storing chronologies with sample depth as well as data on how many of the constituent series increase and decrease. This format therefore requires four numbers for each data point: ring-width; sample depth; increasing series; decreasing series. Numbers are stored as integers, left space padded as before, but this time only using 5 characters not 6. Four data points are included on each line, therefore this means there are 16 numbers per row and each row is 80 characters long.

I.2   Example file - raw series

```
1   HEADER:
2   DateEnd=−66
3   KeyNo=27
4   Project=Growth studies
5   Length=103
6   Location=Example site
7   Species=PISY
8   SapWoodRings=14
9   WaldKante=WKF
10  State=Colorado
11  PersId=FR
12  KeyCode=271017
13  Country=USA
14  DateOfSampling=19950506
15  TreeNo=5
16  CoreNo=1
17  Exposition=North−West
18  CreationDate=19970526
19  SoilType=Sand
20  DATA: Tree
21     125    130     99    120    115    145    151    130    135    151
22     200    190    151    170    170    174    170    200    210    130
23     180    197    210    160    180    155    180    199    140    150
24     146    140    145    150    155    110    115    113    120    130
25     110    120    150    120    120    110    115    160    160    145
26     135    145    125    115    145    149    120    150    160     99
27     110     75     70     82     96     90    120    151    155    130
28     132    133    149    110    130    120    128    118    125    115
29      95     90    110     98     80     85     97     88     70    100
30      90     70     80     90     85     78     95     84     70     90
31      80     75     70      0      0      0      0      0      0      0
```

I.3   Example file - chronology

```
1   HEADER:
2   KeyCode=ABCK0530
3   DataFormat=HalfChrono
4   SeriesType=Mean curve
5   Length=60
6   DateBegin=987
7   DateEnd=1046
8   Dated=Dated
9   Location=Example site
10  Species=QUSP
11  GlobalMathCommentCount=0
12  CommentCount=0
13  DATA: Double
14     125      1    125      2    264      2    206      2    115      2
15     111      2    188      2    308      2    197      2    419      2
16     238      2    227      2    279      2    293      2    271      2
17     309      2    170      2    204      2    163      2    175      2
18     164      2    211      2    134      2    141      2    107      2
19      72      2     74      2     91      2    110      2     47      2
20      87      2     87      2     35      2     47      2     80      2
21      66      2     38      2     82      2     78      2     65      2
22      63      2     76      2     67      2     91      2     73      3
23      39      3     41      3     78      3     57      3     54      3
24      41      3     39      3     52      3     53      3     43      3
25      48      3     32      3     32      3     48      3     59      3
```

KINSYS-KS

Format name	KINSYS-KS
Other name(s)	KINSYS, KINSYS/KS-Sauvala, MIT
Type	Text file
Extension(s)	.MIT
Read/write support	Read
Reference implementation	KINSYS-KS
Data / metadata	Data and structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Unknown

J.1   Description

The KINSYS-KS format was designed for use with the application of the same name at the Finnish Forest Research Institute – Metla. The format is for raw measurement series and contains both data and structured metadata. The primary focus of the metadata fields included are for forestry sampling.

The format allows for multiple data series within a single file, with each series beginning with a 14 line header, followed by multiple data lines, and finishing with the line `*** DATA END ***`.

The first line of the header contains the date and time the file was created. All subsequent header lines are numbered #0, #1, #2 etc and contain one or more metadata fields delimited with a backslash as shown below:

- ▶ Measurement timestamp
- ▶ #0 - Project code / Project name / Responsible person
- ▶ #1 - Sampling date
- ▶ #2 - Ycoord / Xcoord / Elevation
- ▶ #3 - Experiment / Period of measurement / Location
- ▶ #4 - Plot / Subplot
- ▶ #5 - Measurement series code / ID Code
- ▶ #6 - Tree species code
- ▶ #7 - Last measurement year / Subsample code
- ▶ #8 - Incomplete growth / estimated age increase / Last ring type

- ▶ #9 - Sampling azimuth / measurement direction / Sample height / Sample height code
- ▶ #10 - User defined parameters
- ▶ #11 - Number of tree rings in sample / Data type / Columns / Number of decimals
- ▶ #12 - Number of tree rings measured

The meanings and formats of these fields are as follows:

Measurement timestamp - This is the date and time that the file was created. It is in the format dd-mm-yy hh:mm:ss

Project code - Administrative code for the project

Project name - Administrative name for the project

Responsible person - Person responsible for the project

Sampling date - Date the sample was taken formated dd.mm.yy

Ycoord - Y coordinate of the location of the site in Finnish KKJ Uniform Grid coordinates

Xcoord - X coordinate of the location of the site in Finnish KKJ Uniform Grid coordinates

Elevation - Elevation of the site in metres

Experiment -

Period of measurement -

Location - Name of site

Plot - Code number for the plot being sampled

Subplot - Code number of the sub-plot being sampled

Measurement series code - Code for the measurement series

ID Code -

Tree species code - Species code using either the ITRDB or VMI Finnish National Forest Inventory schemes

Last measurement year - Calendar year for the last measurement

Subsample code - Code for if the sample is broken into pieces - marked a,b,c etc. 'x' marks a problematic samples where there are difficulties in analysis

Incomplete growth - If the last incomplete growth ring is measured that this is set to '1', otherwise it is '0'

Estimated age increase - Estimate for the number of tree-rings between the birth of the tree and the pith at the sample height. Taken from VMI correction tables produced on a site and regional basis

Last ring type - If the last incomplete ring is earlywood then this is marked as '1', otherwise it is marked as '2'

Sampling azimuth - Either a letter code for the radii measured, or an angle in degrees of the azimuth along the measurement radius

Measurement direction - Pith-to-bark = 0, Bark-to-pith = 1

Sample height - Height at which sample was taken

Height code -

User defined parameters - Free-text information

Number of tree rings in sample - Number of rings in sample including those not measured. If all rings were measured then this is indicated with a full-stop

Data type - Coded as: 90=ring-width, 91=height shoots, 92=volume growths, 93=earlywood, 94=latewood. Defaults to 90=ring-width.

Column with - Number of digits the data is stored in. Default=3.

Number of decimals - Number of decimal places the values are stored in. Default=2;

Number of tree rings measured - Number of rings that have been measured.

J.2   Example file

```
1   30−05−09 11:09:05
2   # 0 Project Index/Timonen
3   # 1 30.05.2009
4   # 2 4905692/7596282/365
5   # 3 100//Kalmankaltio
6   # 4 110/1
7   # 5 19/6787
8   # 6 1
9   # 7 2008
10  # 8 0
11  # 9 a
12  # 10 Fire scars
13  # 11 .
14  # 12 25
15  57
16  57
17  74
18  76
19  57
20  51
21  59
22  90
23  82
24  95
25  67
26  78
27  99
28  79
29  87
30  112
31  87
32  88
33  81
34  77
35  94
36  110
37  81
38  108
39  129
40  *** DATA END ***
```

Microsoft Excel 97/2000/XP

Format name	Microsoft Excel 97/2000/XP
Other name(s)	Binary Interchange File Format, BIFF
Type	Binary file
Extension(s)	xls
Read/write support	Read and write
Reference implementation	Microsoft Excel
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Microsoft

K.1   Description

The Excel file format is a widely used format for storing spreadsheet data. It is a proprietary binary format created by Microsoft but suppported by many spreadsheet and statistical applications. It is not to be confused with the Office Open XML format which was introduced by Microsoft with MS Office 2007 and typically has the file extension xlsx.

Although Excel files can contain multiple sheets in a workbook, only the first sheet is considered. Like the CSV and ODF Spreadsheet formats, support for Excel files is limited to a particular layout or style of spreadsheet. The layout of the data sheet should be as follows:

- ► Row 1 - Header names for each column
- ► Column A - Year values
- ► Column B+ - One column for each series containing data values. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

Microsoft Excel 2007

Format name	Microsoft Excel 2007
Other name(s)	Office Open XML Spreadsheet, OOXML, OpenXML
Type	XML file
Extension(s)	xlsx
Read/write support	Read and write
Reference implementation	ISO 29500
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Microsoft

L.1   Description

This is the new XML file format introduced by Microsoft with Excel 2007. Unlike the binary format used by the previous version of Excel, this format is an open standard. However, it should not be confused with the OpenDocument Format standard that was developed by the OASIS consortium.

The layout of the data sheet should be just as for the Excel 97/2000/XP format:

▶ Row 1 - Header names for each column
▶ Column A - Year values
▶ Column B+ - One column for each series containing data values. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

See the screenshot in the Microsoft Excel 97/2000/XP format to see how an example of how the spreadsheet should look.

Nottingham

Format name	Nottingham
Other name(s)	Nottingham Laboratory format
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	Unknown
Data / metadata	Data only
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Cliff Litton

M.1 Description

The Nottingham format was designed by Cliff Litton. It is a simple text format with no support for metadata.

Line 1 contains a series name and an integer indicating how many data values there are in the file. Subsequent lines contain the data represented as 1/100th mm integers in twenty columns seemingly in either 4 characters or 3 characters + 1 space.

There is no known reference implementation for this format and few known examples of data so little is known about how it should handle unusual situations such as negative values, values > 999 etc.

M.2 Example file

```
ABCD01       176
342 338 334 409 362 308 360 264 325 318 134 151 219 268 290 222 278 258 173 198
294 202 170 176 172 121  87 130 114 108 170 135 131 126  87 100  86 104 103 127
112  94  96 120 168 149 119 124  79  67  88  90  93  77  49  42  53  38  57  43
 50  41  56  66  62  55  55  45  47  63  58  60  44  45  49  50  62  61  43  54
 91  60  56  43  52  51  65  68  55  44  41  75  94  78  63  69  58  75  55  47
 58  46  62  45  52  50  77  50  63  75  77  64  66  57  80  57  78  65  68  75
 65  98  85  82 119  89  85  87  83 108 129 123 160 117 129 121  88  69  97  77
 96 106  71  89  50  65 133  89  88  50  60  95  95  91 102 158  83  55  98  70
 45  46  40  36  64  58  52  58  56  94  51  48  47  60  49  48
```

ODF Spreadsheet

Format name	ODF Spreadsheet
Other name(s)	ODF, ODS, OpenDocument Spreadsheet, OpenOffice.org Spreadsheet,
Type	XML file
Extension(s)	ods
Read/write support	Read and write
Reference implementation	ISO/IEC 26300:2006
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	OASIS consortium

N.1   Description

The OpenDocument Format (ODF) spreadsheet format is an XML-based specification developed by the Organization for the Advancement of Structured Information Standards (OASIS) consortium. It should not be confused with the similarly named Office Open XML format developed by Microsoft. The ODF spreadsheet format is an open standard which can be read by most modern spreadsheet applications including MS Excel, OpenOffice.org and Google Docs.

Support for ODF spreadsheets in TRiCYCLE is necessarily limited to a particular layout of spreadsheet:

▶ Row 1 - Header names for each column
▶ Column A - Year values
▶ Column B+ - One column for each series containing data values. Cells are left empty if no data is available for a series because it does not extend to a particular year. Data must be continuous for each series, so missing/unmeasured rings should be included as zero.

Oxford

Format name	Oxford
Other name(s)	Dan Miles Format, English Heritage Format
Type	Text file
Extension(s)	Various including dan, ddf but often none
Read/write support	Read and write
Reference implementation	Various English Heritage applications
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	No
Original designer	Ancient Monuments Laboratory of English Heritage

O.1   Description

The Oxford format seems to be only currently used in the Oxford Dendrochronology Laboratory. It was designed in the 1980s for use with a number of DOS based applications for the English Heritage Ancient Monuments Laboratory. It is still actively used by the Oxford Lab with these programs and a number of newer Windows applications.

The file is a text file format containing two header lines following by a block of data values and an optional block of count/sample depth values. Some files also contain a number of comment lines at the end of the file.

Line 1 contains the following fields:

- ▶ Char 1 - Apostrophe
- ▶ Chars 2-8 - Series name
- ▶ Char 9-10 - spaces
- ▶ Char 11 - $<$
- ▶ Chars 12-15 - First year in sequence (when series is securely dated). Year should be left padded with spaces if less than 4 characters.
- ▶ Char 16 - hyphen
- ▶ Chars 17-20 - Last year in sequence (when series is securely dated). Year should be left padded with spaces if less than 4 characters.
- ▶ Char 21 - space

- ▶ Char 22+ - Description - typically name of site/building etc
- ▶ Final char - optional apostrophe

Line 2 contains:

- ▶ Integer number of years
- ▶ Comma
- ▶ Integer start year

The start year on line 2 and the first year on line 1 will be the same for securely dated series. When the series is tentatively or relatively dated the first year (and/or) the last year on line 1 will be left blank. For undated series the start year is set to 1001.

The data lines follow the two header lines. These typically contain 10 data values per line, but there can be more (if rings have been added) or less e.g. last line. The values are in 1/100th mm integers and can only contain three digits (e.g. max 999 1/100th mm). Data values are space delimited. Some example files contain values that are left padded with zeros if the value is on 1 or 2 characters wide (e.g. '025' rather than ' 25').

Following the data values there should be an empty line followed by an optional sample count/depth block. The count block is formatted in largely the same way as the data values block. The values are stored in columns 2 characters (rather than 3 characters) wide. Like the data values, the count values are space delimited integers, typically (but not always) 10 per line.

The file is terminated with 0, 1 or 2 free-text comment lines. A number of Oxford data files have been seen that terminate with the ASCII control character referred to variably as 'SUB', 'SUBSTITUTE' or 'CTRL+Z' (represented in Unicode as character dec 26 - hex 1A). It is not clear whether this is necessary for any particular programs to function.

O.2 Limitations

- ▶ Only holds whole ring-width data
- ▶ Does not cope with data values > 999 1/100th mm
- ▶ Does not cope with chronologies of > 99 samples
- ▶ Does not allow dates before 1AD

O.3 Example file

```
1  'ABCD        <1850−1925> A Fictious site − abcd1 abcd2'
2  75,1850
3  422 582 355 266 225 271 361 235 387 395
4  794 611 446 248 277 359 111 226 189 711
5  464 172 190 239 128 153 234 828 207 157
6  768 180 178 168 204 163 160 255 166 136
7  182 201 142 188 223 186 150 135 134 666
8  191 122 223 555 123 126 108 133 137 134
9  161 222  93 100 132 104  86 277 101 141
10 185 151 261 110 145
11
12   1  2  2  2  2  2  2  2  2  2
13   2  2  2  2  2  2  2  2  2  2
14   2  2  2  2  2  2  2  2  2  2
15   2  2  2  2  2  2  2  2  2  2
16   2  2  2  2  2  2  2  2  2  2
17   2  2  2  2  2  2  2  2  2  2
18   2  2  2  2  2  2  2  2  2  2
19   2  2  2  2  1
```

PAST4

Format name	PAST4
Other name(s)	P4P PAST4 Project File
Type	Text file
Extension(s)	p4p
Read/write support	Read and write
Reference implementation	PAST4
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Bernhard Knibbe

The PAST4 format (Knibbe, 2008) is the native file format for SCIEM's PAST4 software. It is a hybrid XML file, containing most metadata in structured XML but some metadata and all data as plain text. It is unique amongst dendro data formats in that it contains not only data and metadata but also settings information for the PAST4 software such as details on what colours to use in graphs, which series should be displayed on screen etc. The general structure of a P4P file is as follows:

- ▶ Project header (required)
- ▶ Settings (optional)
- ▶ Groups (required, repeatable)
- ▶ Records (required, repeatable)

The root XML tag for the file is `<PAST_4_PROJECT_FILE>`. Inside this is the `<PROJECT>` tag which contains the following attributes:

- ▶ ActiveGroup - Zero based index specifying which group is active
- ▶ EditDate - Date the file was last edited
- ▶ Groups - Number of groups within this project
- ▶ Locked - Either TRUE or FALSE indicating whether a password is required to open the file
- ▶ Name - Name of the project
- ▶ Password - Password used to lock the project
- ▶ PersID - Abbreviation of the authors name
- ▶ Records - Number of records in the project
- ▶ Reference - Zero based index indicated which is the reference series (-1 if none selected)
- ▶ Sample - Zero based index indicating which is the selected sample (-1 if none selected)

▶ Version - Version number for this PAST4 format. At the time of writing only one version exists (400).

Of these fields only Name, Groups and Records are mandatory. The project tag can also contain a `<![CDATA[` tag which allows the storing of a project description in plain text.

Next comes the `<SETTINGS>` tag. This is one very large XML tag with many attributes controlling the what PAST4 should display the data. The contents of this tag are optional and are therefore irrelevant for the transfer of dendro data.

Next comes one or more `<GROUPS>` tags. A group is an arbitrary collection of series, perhaps representing a number of measurements of a single object, or perhaps an administrative collection of series. Groups can be nested in a hierarchy, but rather than use the hierarchical nature of XML files, the format instead lists all groups side-by-side and maintains the relationships through the use of an 'owner' attribute containing the index of the parent group. This arrangement means than any changes to the hierarchy, or the deletion of a group requires all indices to be carefully updated to avoid corrupting the file. The group tag has the following attributes:

▶ Name - Name of the group
▶ Visible - Either TRUE or FALSE indicating whether the group should be shown in graphs
▶ Fixed - Either TRUE or FALSE indicating whether the group can be moved
▶ Locked - Either TRUE or FALSE. If locked the group can be used in the calculation of further mean values.
▶ Changed - Internal TRUE or FALSE value for keeping track of changes
▶ Expanded - TRUE or FALSE value indicating whether the group should be expanding in the project navigator window
▶ UseColor - TRUE or FALSE value for is content should be displayed in color
▶ HasMeanValue - TRUE or FALSE indicating if the group has a dynamic mean value
▶ IsChrono - TRUE or FALSE indicating if the group mean is calculated with sample depth information
▶ Checked - TRUE or FALSE indicating if the group is locked and checked
▶ Selected - TRUE or FALSE indicated in the group is selected in the project navigation window
▶ Color - 24bit integer indicating the RGB volor value for the group using Borland format
▶ Quality - Integer value describing the quality of the group mean
▶ MVKeycode - String code for the group. If empty the Name field is used
▶ Owner - Integer pointing containing the index of the parent group if this group is in a hierarchy. If its a top level group it should be -1.

As with the project tag, the group tag can also contain a `<![CDATA[` section for storing a plain text description of the group.

The final tag type in the file is the `<RECORDS>` tag. These contain the actual data series and most of the metadata. Like group tags, records tags are placed side-by-side in the file and are placed into the group hierarchy by the use of the 'owner' attribute. In addition, the tag also has the following attributes:

▶ Keycode - Name of the series
▶ Length - Integer for the number of rings
▶ Owner - Integer index to the group to which this record belongs
▶ Chrono - TRUE or FALSE indicating whether this record has density information
▶ Locked - TRUE or FALSE indicating in the record can be moved
▶ Filter - TRUE or FALSE indicating if an indexing function is appled to the data
▶ FilterIndex - Integer index for the filter used
▶ FilterS1 - Parameter 1 for the filter
▶ FilterS2 - Parameter 2 for the filter
▶ FilterB1 - Additional filter parameter
▶ FilterWeight - Additional filter parameter
▶ Offset - Position of the first ring
▶ Color - 24bit RGB color for record in Borland format
▶ Checked - TRUE or FALSE indicating is the record is selected for use in the dynamic group mean

- ▶ !VShift - Temporary integer value added to data value to shift vertically in graphs
- ▶ IsMeanValue - TRUE or FALSE indicating if this is a dynamic mean value
- ▶ Pith - TRUE or FALSE
- ▶ SapWood - Integer storing the number of sapwood rings
- ▶ Location - String location information
- ▶ Waldkante - String description of presence of waney edge
- ▶ FirstValidRing - Integer indicating which ring is the first valid ring. If ¿0 then some rings are discarded
- ▶ LastValidRing - Integer indicating which ring is the last valid ring. If ¿0 then some rings are discarded
- ▶ UseValidRingsOnly - TRUE or FALSE - internal use only
- ▶ Quality - Integer indicating the quality of the record

The record tag then contains a `<HEADER>` tag with a `<![CDATA[` section which includes additional free-text header information. There are no requirements as to how information should be laid out in this field however many users seem to adopt the Heidelberg style of keyword=value.

Next comes the `<DATA>` tag which is empty except another `<![CDATA[` section. This is where the actual ring-width data is stored. Each data value is recorded on a separate line (using CR LR line breaks). Each line contains the following six tab delimited fields:

- ▶ Ring width as a floating point number
- ▶ Sample depth
- ▶ Number of sample increasing
- ▶ Latewood percentage as a floating point value 0-1 (0 if not known)
- ▶ Duplicate/backup ring-width value to store the original ring-width value. If an index is applied the ring-width value in column 1 is altered.
- ▶ Comment string about this particular ring

P.1   Dating

PAST4 contains an option for enabling/disabling the year 0 but it does not record within the data file whether the option was set when the file was created. By default the year 0 is disabled therefore the library treats PAST4 files as if they use the Gregorian calendar but it is possible that files were in fact created with the Astronomical calendar in mind.

P.2 Example file

```
 1  <?xml version="1.0"?>
 2  <PAST_4_PROJECT_FILE>
 3      <PROJECT Name="title0" Version="400" Locked="FALSE" Password=""
 4          CreationDate="04/05/2006 2:13:51 PM" EditDate="09/01/2010 13:02" ActiveGroup="0"
 5          Reference="−1" Sample="−1" PersID="investigator0" Groups="2" Records="3">
 6  <![CDATA[description0
 7  ]]></PROJECT>
 8      <SETTINGS/>
 9      <GROUP Name="title1" Visible="TRUE" Fixed="FALSE" Locked="FALSE" Changed="FALSE"
10          Expanded="TRUE" UseColor="TRUE" HasMeanValue="FALSE" IsChrono="FALSE"
11          Checked="FALSE" Selected="FALSE" Color="0" MVKeycode="" Owner="−1">
12          <![CDATA[]]></GROUP>
13      <GROUP Name="Unnamed Group" Visible="TRUE" Fixed="FALSE" Locked="FALSE" Changed="FALSE"
14          Expanded="TRUE" UseColor="TRUE" HasMeanValue="FALSE" IsChrono="FALSE" Checked="FALSE
              "
15          Selected="FALSE" Color="0" MVKeycode="" Owner="−1"><![CDATA[]]></GROUP>
16      <RECORD Keycode="title6" Length="4" Owner="0" Chrono="FALSE" Locked="FALSE" Filter="
          FALSE"
17          FilterIndex="−1" FilterS1="100" FilterS2="100" FilterB1="FALSE" FilterWeight=""
              Offset="0"
18          Color="0" Checked="FALSE" VShift="0" IsMeanValue="0" Pith="FALSE" SapWood="0"
19          Location="locationComment1" Species="Quercus" Waldkante="" FirstValidRing="0"
20          LastValidRing="0" UseValidRingsOnly="FALSE">
21          <HEADER><![CDATA[Unit=1/100th millimetres
22  ]]></HEADER>
23          <DATA><![CDATA[123         1         1         0         123
24  123       1         1         0         123
25  123       1         1         0         123
26  125       1         1         0         125
27  ]]></DATA>
28      </RECORD>
29      <RECORD Keycode="title6" Length="4" Owner="0" Chrono="FALSE" Locked="FALSE" Filter="
          FALSE"
30          FilterIndex="−1" FilterS1="100" FilterS2="100" FilterB1="FALSE" FilterWeight=""
              Offset="0"
31          Color="0" Checked="FALSE" VShift="0" IsMeanValue="0" Pith="FALSE" SapWood="0"
32          Location="locationComment1" Species="QUSP" Waldkante="" FirstValidRing="0"
33          LastValidRing="0" UseValidRingsOnly="FALSE">
34          <HEADER><![CDATA[Unit=1/100th millimetres
35  ]]></HEADER>
36          <DATA><![CDATA[123         1         1         0         123
37  123       1         1         0         123
38  123       1         1         0         123
39  125       1         1         0         125
40  ]]></DATA>
41      </RECORD>
42      <RECORD Keycode="Unnamed series" Length="2" Owner="1" Chrono="FALSE" Locked="FALSE"
43          Filter="FALSE" FilterIndex="−1" FilterS1="100" FilterS2="100" FilterB1="FALSE"
44          FilterWeight="" Offset="0" Color="0" Checked="FALSE" VShift="0" IsMeanValue="0"
45          Pith="FALSE" SapWood="0" Location="" Species="" Waldkante="" FirstValidRing="0"
46          LastValidRing="0" UseValidRingsOnly="FALSE">
47          <HEADER><![CDATA[Unit=Wierd units
48  ]]></HEADER>
49          <DATA><![CDATA[96          1         1         0         96        fire_damage; fire_damage;
50  34        1         1         0         34        fire_damage; fire_damage;
51  ]]></DATA>
52      </RECORD>
53  </PAST_4_PROJECT_FILE>
```

Sheffield

Format name	Sheffield
Other name(s)	D Format
Type	Text file
Extension(s)	d
Read/write support	Read and write
Reference implementation	Dendro for Windows
Data / metadata	Data and some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	Yes
Multi series support	No
Original designer	Ian Tyers

Q.1   Description

Sheffield format (Tyers, 1999) is a dendro specific text file designed by Ian Tyers for his Dendro for Windows application. It is probably most widely used in the UK but is also used in continental Europe as well as New Zealand.

The format contains both data and some structured metadata with each field/value stored one per line. The order of fields is fixed so missing data must be indicated by the use of a question mark. The data present on each line is as follows:

1. Site name/sample number - Free form text not including `,"()` up to 64 characters
2. Number of rings - Whole positive number
3. Date type - Single character; A = absolute date, R = relative date
4. Start date - Whole number (can be negative). If absolute year then add 10000 to value so 1AD = 10001
5. Raw data type *or* Mean data type
   - ▶ Single character; R = annual raw ring-width data (NB earlier versions used some other codes here for species e.g. ABEFPSU these are all interpreted as equivalent to R)
   - ▶ Single character; W=timber mean with signatures, X=chron mean with signatures, T = timber mean, C = chron mean, M = un-weighted master sequence
6. Raw sapwood number *or* mean number of timbers/chronologies
   - ▶ Whole positive number or 0
   - ▶ Whole positive number

7. Raw edges inf. *or* Mean chronology type
   - ▶ Single character; Y = has bark, ! = has ?bark, W = terminal ring probably complete (i.e. possibly Winter Felled), S = terminal ring probably incomplete (i.e. possibly Summer Felled), B = has h/s boundary, ? = has ?h/s boundary, N = has no specific edge, (NB but may have sap), U = sap/bark unknown, C = charred outer edge, P = possibly charred outer edge
   - ▶ Single character; R = raw unfiltered data, 5 = 5 year running mean, I = indexed data, U = unknown mean type
8. Author and comment - Free form text not including ,"() up to 64 characters
9. UK National grid reference - 2 characters +even no of digits up to 14 characters in all, ? = not known e.g. TQ67848675
10. Latitude and longitude - Either decimal format e.g. 53.382457;-1.513623 or previously N51^30 W1^20
11. Pith - single character; C = centre of tree, V = within 5 years of centre, F = 5-10 years of centre, G = greater than 10, ? = unknown
12. Cross-section code - Two character code; first character, A = whole roundwood, B = half round, C quartered, D radial/split plank, E tangential/sawn plank. second character, 1 untrimmed, 2 trimmed, X irregularly trimmed. or, X = core /unclassifiable, ? unknown/unrecorded
13. Major dimension - whole number in mm, 0 if unrecorded or mean
14. Minor dimension - whole number in mm, 0 if unrecorded or mean
15. Unmeasured inner rings - single character+whole number; use pith codes + number of rings or, H = heartwood, N = none
16. Unmeasured outer rings - single character+whole number; use edges code + number of rings except that S = sapwood with no edge and V is the spring felling equivalent other codes are, H = heartwood with no edge, N = none
17. Group/Phase - free form text not including , " ( ) up to 14 characters
18. Short title - free form text not including , " ( ) up to 8 characters
19. Period - single character; C = modern, P = post medieval, M = medieval, S = Saxon, R = Roman, A = pre Roman, 2 = duplicate e.g. repeat measure, B = multiperiod e.g. long master, ? = unknown
20. ITRDB species code - 4 character code - refer to ITRDB species codes
21. Interpretation and anatomical notes - ? =no interpretation/notes. The interpretation and the anatomical notes can be in any order but each must consist of three parts, a single character A or I for anatomy or interpretation, a separator , for interpretations the date of the start, for anatomy the ringno, a separator , for anatomy the anatomical code for interpretations P for plus, 0 for felled and a number for the length of the range, where more than one record is present these are separated by , there must not be a terminal separator and each record must consist of the tree parts. The anatomical codings can be anything of a single character but supported usage is based on Hans-Hubert Leuschners anatomical codes; D = Density Band, R = Reaction Wood, L = Light Latewood, H = Dense Latewood, F = Frost Ring, K = Small Earlywood Vessels - oak, G = Great Latewood Vessels - oak, T = Wound Tissue, N = Narrow Latewood, A = Light Latewood End, P = Narrow and Light Latewood, Q = Narrow and Dense Latewood
22. Data type - single character; D = ring widths, E = early-wood widths only, L = late-wood widths only, R = late+early wood widths (i.e. reverse of normal rings), I = minimum density, A = maximum density, S = early, late; (i.e. sequentially and separately), M = mixed (?means of others)

The remaining lines contain the data:

- ▶ For each width (equivalent to the value of length) the individual increments etc. if a C X T or W type mean. No negatives or zeros
- ▶ Check field - Single character H
- ▶ For each width the individual weightings of the mean sequences. If an X or W type mean. No negatives or zeros.
- ▶ Check field - Single character R
- ▶ For each width the number of individual series with rising values. No negatives or zeros.
- ▶ Check field - Single character F
- ▶ For each width the number of individual series with falling values. No negatives.

Q.2   Dating

The format copes with the problem of the non-existent year 0AD/BC by adding 10000 to all year values. Therefore:

Year	Value in file
1AD	10001
1BC	10000
9999BC	2
10000BC	1

Q.3   Example file

```
 1  Ship wreck 4 timber mean
 2  170
 3  A
 4  10784
 5  W
 6  4
 7  R
 8  made PB 22/6/2004
 9  ?
10  ?
11  ?
12  ?
13  0
14  0
15  N
16  N
17  A
18  Example
19  M
20  QUSP
21  ?
22  D
23  391
24  454
25  309
26  314
27  270
28  273
29  229
30  319
31  267
32  276
33  128
34  163
35  221
36  269
37  214
38  201
39  218
40  199
41  198
42  209
43  156
44  177
45  ...
```

Topham

Format name	Topham
Other name(s)	Instrument format
Type	Text file
Extension(s)	txt
Read/write support	Read and write
Reference implementation	Not known
Data / metadata	Data only
Calendar type	n/a
Absolute dating support	No
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	John Topham

R.1 Description

The Topham format is probably the most simplistic of formats consisting of just a column of decimal data values and no metadata whatsoever. Each data value is a decimal ring width in millimetres.

R.2 Example file

```
1   3.42
2   3.38
3   3.34
4   4.09
5   3.62
6   3.08
7   3.60
8   2.64
9   3.25
10  3.18
11  3.42
12  3.38
13  ...
```

TRiDaS

Format name	TRiDaS
Other name(s)	Tree-Ring Data Standard, TRiDaS XML
Type	Text file
Extension(s)	xml
Read/write support	Read and write
Reference implementation	TRiCYCLE
Data / metadata	Data and structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	Yes
Multi series support	Yes
Original designer	Esther Jansma, Peter Brewer and Ivo Zandhuis

S.1 Description

TRiDaS (Tree-Ring Data Standard see http://www.tridas.org) is a data format designed by over 80 dendrochronologists, computer scientists and users of dendrochronological data from a variety of associated fields as part of the DCCD project and the Dendro Data Standard forum. It is designed to accurately represent any dendro data and metadata and it is hoped over time the dendro community will accept TRiDaS as the de facto standard for all dendro data.

The format uses extensible markup language (XML) which means the standard can be extended and evolve as future needs change. The format is structured around the eight data entities described below:

A project is defined by a laboratory and encompasses dendrochronological research of a particular object or group of objects. Examples include: the dating of a building; the research of forest dynamics in a stand of living trees; the dating of all Rembrandt paintings in a museum. What is considered a "project" is up to the laboratory performing the research. It could be the dating of a group of objects, but the laboratory can also decide to define a separate project for each object. Therefore, a project can have one or more objects associated with it.

An object is the item to be investigated. Examples include: violin; excavation site; painting on a wooden panel; water well; church; carving; ship; forest. An object could also be more specific, for example: mast of a ship; roof of a church. Depending on the object type various descriptions are made possible. An object can have one or more elements and can also refer to another (sub) object. For instance a single file may contain three objects: an archaeological site object, within which there is a building object, within which there is a beam

object. The list of possible object types is extensible and is thus flexible enough to incorporate the diversity of data required by the dendro community. Only information that is essential for dendrochronological research is recorded here. Other related data may be provided in the form of a link to an external database such as a museum catalogue.

An element is a piece of wood originating from a single tree. Examples include: one plank of a water well; a single wooden panel in a painting; the left-hand back plate of a violin; one beam in a roof; a tree trunk preserved in the soil; a living tree. The element is a specific part of exactly one object or sub object. An object will often consist of more than one element, e.g., when dealing with the staves (elements) of a barrel (object). One or more samples can be taken from an element and an element may be dated using one or more derivedSeries.

A sample is a physical specimen or non-physical representation of an element. Examples include: core from a living tree; core from a rafter in a church roof; piece of charcoal from an archaeological trench; slice from a pile used in a pile foundation; wax imprint of the outer end of a plank; photo of a back plate of a string instrument. Note that a sample always exists and that it can either be physical (e.g. a core) or representative (e.g. a picture). A sample is taken from exactly one element and can be represented by one or more radii.

A radius is a line from pith to bark along which the measurements are taken. A radius is derived from exactly one sample. It can be measured more than once resulting in multiple measurementSeries.

A measurementSeries is a series of direct, raw measurements along a radius. A single measurementSeries can be standardised or a collection of measurementSeries can be combined into a derived- Series. The measurements themselves are stored separately as values.

A derivedSeries is a calculated series of values and is a minor modification of the "v-series" concept proposed by Brewer et al. (2010). Examples include: index; average of a collection of measurementSeries such as a chronology. A derivedSeries is derived from one or more measurementSeries and has multiple values associated with it.

A value is the result of a single ring measurement. Examples include: total ring width; earlywood width; latewood width. The values are related to a measurementSeries or a derivedSeries. In case of a measurementSeries the variable and its measurement unit (e.g. microns, 1/100th mm etc) are recorded as well.

For a full description of the standard see Jansma et al. (2010).

S.2   Example file

```xml
1  <?xml version="1.0" encoding="UTF−8" ?>
2  <tridas xmlns:xsi="http://www.w3.org/2001/XMLSchema−instance"
3      xsi:schemaLocation="http://www.tridas.org/1.2.1 ../dev/sourceforge/tridas/XMLSchema
          /1.2.1/tridas−1.2.1.xsd"
4      xmlns="http://www.tridas.org/1.2.1" xmlns:xlink="http://www.w3.org/1999/xlink">
5      <project>
6          <title>Aegean Dendrochronology Project</title>
7          <identifier domain="dendro.cornell.edu">C</identifier>
8          <createdTimestamp certainty="exact">1997−02−01T14:13:51.0Z</createdTimestamp>
9          <lastModifiedTimestamp certainty="exact">1997−02−01T14:13:51.0Z</
              lastModifiedTimestamp>
10         <type>Dating</type>
11         <description>Our key long−range goal is to build long multi−millennial scale tree−
              ring
12             chronologies in the Aegean and Near East that will extend from the present to
                  the
13             early Holocene to cover, broadly speaking, the last 10,000 years of human and
14             environmental history. Our raison d'etre is to provide a dating method for the
                  study
15             of history and prehistory in the Aegean that is accurate to the year. This kind
                  of
16             precision has, up to now, been lacking in ancient studies of this area. Indeed,
                  few
```

```
17          archaeological problems stimulate as much rancor as chronology, especially that
                of
18          the Eastern Mediterranean. The work of the Aegean and Near Eastern
                Dendrochronology
19          Project aims to help to bring some kind of rational and neutral order to Aegean
                and
20          Near Eastern chronology from the Neolithic to the present. </description>
21      <laboratory>
22          <name>Malcolm and Carolyn Weiner Laboratory for Aegean and Near Eastern
                Dendrochronology</name>
23          <address>
24              <addressLine1>B48 Goldwin Smith Hall</addressLine1>
25              <addressLine2>Cornell University</addressLine2>
26              <cityOrTown>Ithaca</cityOrTown>
27              <stateProvinceRegion>NY</stateProvinceRegion>
28              <postalCode>14853</postalCode>
29              <country>USA</country>
30          </address>
31      </laboratory>
32      <category>Archaeology</category>
33      <investigator>Peter I Kuniholm</investigator>
34      <period>1976-present</period>
35      <reference>reference1</reference>
36      <object>
37          <title>White Tower, Thessaloniki</title>
38          <identifier domain="dendro.cornell.edu"
39              >28acb483-f337-412f-a063-59d911c37594</identifier>
40          <createdTimestamp certainty="exact">1997-02-01T14:13:51.0Z</createdTimestamp>
41          <lastModifiedTimestamp certainty="exact">1997-02-01T14:13:51.0Z</
                lastModifiedTimestamp>
42          <type normalStd="Corina Dictionary" normalId="4" normal="Building">Building</
                type>
43          <description>The White Tower of Thessaloniki was originally constructed by the
                Ottomans
44              to fortify the city's harbour.</description>
45          <coverage>
46              <coverageTemporal>Ottoman</coverageTemporal>
47              <coverageTemporalFoundation>Stylistic</coverageTemporalFoundation>
48          </coverage>
49          <location>
50              <locationGeometry xmlns:gml="http://www.opengis.net/gml">
51                  <gml:Point srsName="urn:ogc:def:crs:EPSG:6.6:4326">
52                      <gml:pos>40.6263 22.9485</gml:pos>
53                  </gml:Point>
54              </locationGeometry>
55              <locationPrecision>20</locationPrecision>
56              <locationComment>Thessaloniki, Greece</locationComment>
57          </location>
58          <object>
59              <title>Fourth floor</title>
60              <type>Floor</type>
61              <element>
62                  <title>C-TWT-65</title>
63                  <identifier domain="dendro.cornell.edu"
64                      >89dbd409-03a3-42a0-9391-62c6be7009ad</identifier>
65                  <createdTimestamp certainty="exact">1997-02-01T14:13:51.0Z</
                        createdTimestamp>
66                  <lastModifiedTimestamp certainty="exact"
67                      >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
68                  <type normalStd="Corina Dictionary" normalId="3" normal="Rafter">Rafter<
                        /type>
69                  <description>15th Rafter from the south</description>
70                  <taxon normalStd="Catalogue of Life Annual Checklist 2008" normal="
                        Quercus"
71                      normalId="49139">Quercus sp.</taxon>
```

```
72                            <dimensions>
73                                <unit normalTridas=" metres" />
74                                <height>1</height>
75                                <width>1</width>
76                                <depth>1</depth>
77                            </dimensions>
78                            <authenticity>Original</authenticity>
79                            <sample>
80                                <title>C-TWT-65-A</title>
81                                <identifier domain=" dendro . cornell . edu"
82                                    >ff688357-b2d4-4394-a21a-90696cd4558c</identifier>
83                                <createdTimestamp certainty=" exact"
84                                    >1997-02-01T14:13:51.0Z</createdTimestamp>
85                                <lastModifiedTimestamp certainty=" exact"
86                                    >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
87                                <type normal=" Corina Dictionary" normalId="1" normalStd=" Section"
88                                    >Section</type>
89                                <samplingDate certainty=" exact">1981-07-25</samplingDate>
90                                <state>Dry</state>
91                                <radius>
92                                    <title>C-TWT-65-A-B</title>
93                                    <identifier domain=" dendro . cornell . edu"
94                                        >5b7baa8b-cd4e-4b3b-88fa-82939420e544</identifier>
95                                    <createdTimestamp certainty=" exact"
96                                        >2006-05-04T18:13:51.0Z</createdTimestamp>
97                                    <lastModifiedTimestamp certainty=" exact"
98                                        >2006-05-04T18:13:51.0Z</lastModifiedTimestamp>
99                                    <woodCompleteness>
100                                       <pith presence=" absent" />
101                                       <heartwood presence=" incomplete" />
102                                       <sapwood presence=" complete" />
103                                       <bark presence=" present" />
104                                   </woodCompleteness>
105                                   <measurementSeries>
106                                       <title>C-TWT-65-A-B-A</title>
107                                       <identifier domain=" dendro . cornell . edu"
108                                           >8c50234e-8eda-41bb-b578-01cc881d1ea1</identifier>
109                                       <createdTimestamp certainty=" exact"
110                                           >1997-02-01T14:13:51.0Z</createdTimestamp>
111                                       <lastModifiedTimestamp certainty=" exact"
112                                           >1997-02-01T14:13:51.0Z</lastModifiedTimestamp>
113                                       <analyst>Laura Steele</analyst>
114                                       <dendrochronologist>Peter I Kuniholm</dendrochronologist>
115                                       <measuringMethod normalStd=" Corina Dictionary" normalId="1"
116                                           >Measuring platform</measuringMethod>
117                                       <interpretation>
118                                           <firstYear suffix="AD">1254</firstYear>
119                                           <statFoundation>
120                                               <statValue>8.3</statValue>
121                                               <type>t-score</type>
122                                               <usedSoftware>Corina 2.10</usedSoftware>
123                                           </statFoundation>
124                                           <deathYear suffix="AD">1535</deathYear>
125                                           <provenance>Possibly from the region of Serres</
                                                   provenance>
126                                       </interpretation>
127                                       <values>
128                                           <variable normalTridas=" Ring width" />
129                                           <unit normalTridas="1/100th millimetres" />
130                                           <value value="54" />
131                                           <value value="111" />
132                                           <value value="71" />
133                                           <value value="40" />
134                                           <value value="56" />
135                                       </values>
```

```
136                              </measurementSeries>
137                          </radius>
138                      </sample>
139                  </element>
140              </object>
141          </object>
142      </project>
143  </tridas>
```

TRIMS

Format name	TRIMS
Other name(s)	None known
Type	Text file
Extension(s)	rw
Read/write support	Read and write
Reference implementation	
Data / metadata	Data only
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	No
Original designer	Unknown

This is a simple data only text file format. These files were originally produced using the Henson rotary micrometer measuring stages but have largely been phased out.

▶ Line 1 - Initials of user that created the series
▶ Line 2 - Date the file was created in dd/MM/YY format
▶ Line 3 - Year of first data value (0 treated as undated series)
▶ Line 4+ - Space character followed by an integer data value in 1/100th mm
▶ Final line - Space character + 999 denoting end of series.

T.1   Example file

```
1   pb
2   05/10/94
3     1816
4     169
5     96
6     165
7     85
8     139
9     87
10    112
11    ...
12    999
```

Tucson

Format name	Tucson
Other name(s)	Decadal, RWL, CRN, ITRDB, Time series format, TSF
Type	Text file
Extension(s)	Various including tuc, rwl, dec, crn
Read/write support	Read and write
Reference implementation	COFECHA
Data / metadata	Data with some structured metadata, however, standardisation of metadata is very poor resulting in metadata often being little more than free text comments
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Richard Holmes (based on early punch card format)

U.1   Description

The Tucson format is perhaps the most widely used dendro data format. Unfortunately it seems there was never definitive documentation. Support for the format has been incorporated into a number of dendro applications but without format documentation there are variations in these implementations resulting in quite a lot of subtle differences in files. The often tight association between the Dendro Program Library (DPL) and the ITRDB means that perhaps the most definitive documentation for the format is the ITRDB website.

The Tucson format is best considered as covering two different sub-formats which are often referred to by their file extensions (RWL and CRN). RWL files are used for storing ring-width data, whereas CRN files are used for storing chronologies.

The ITRDB website includes detailed information on how to include structured metadata in Tucson format files. Unfortunately there are no tools for creating and/or validating Tucson files so the vast majority of files circulating in the community today (including those in the ITRDB) do not adhere to these standards.

U.2 RWL files

Tucson RWL files begin with three lines of metadata. Strictly these lines should contain structured metadata, but with no software to assist in this, users either only partially stick to these rules, or reject them entirely instead using the three lines as free-text comment lines. The metadata should be set out as follows:

- ▶ Line 1 - Chars 1-6 Site ID
- ▶ Line 1 - Chars 10-61 Site Name
- ▶ Line 1 - Chars 62-65 Species Code followed by optional ID number
- ▶ Line 2 - Chars 1-6 Site ID
- ▶ Line 2 - Chars 10-22 State/Country
- ▶ Line 2 - Chars 23-30 Species
- ▶ Line 2 - Chars 41-45 Elevation
- ▶ Line 2 - Chars 48-57 Lat-Long in degrees and minutes, ddmm or dddmm
- ▶ Line 2 - Chars 68-76 1st and last Year
- ▶ Line 3 - Chars 1-6 Site ID
- ▶ Line 3 - Chars 10-72 Lead Investigator
- ▶ Line 3 - Chars 73-80 comp. date

Then follows the data lines which are set out as follows:

- ▶ Chars 1-8 - Series ID - the series ID should be unique in the file so that it is clear where one series ends and another begins when multiple series are present in the same file.
- ▶ Next 4 chars - Year of first value in this row.
- ▶ Ten data values consisting of a space character and 5 integers. The file and last data line for a series may have less than 10 data values so that the majority of lines begin at the start of a decade.

The final data value should be followed by a stop marker which is either 999 or -9999. When a stop marker of 999 is used this indicates that the integer values in the file are measured in 0.01mm (1/100th mm) units, whereas if a -9999 stop marker is used the units are 0.001mm (microns). The stop marker is therefore used to indicate the end of the data series and the units the data are stored in.

There appears to be no official specification as to how missing rings should be encoded, but the standard notation seems to be to use -999 or 0.

U.3 CRN files

Tucson CRN files are used to store chronology data. In addition to each data values they also have space for a sample depth or count value to record how many values were combined to give each data value. CRN files should strictly begin with the same 3 header lines that are described above for RWL. Like RWL files the specification is often partially adhered to and at times ignored completely.

The data lines for CRN files are quite different to RWL:

- ▶ Chars 1-6 - Series ID
- ▶ Next 4 chars - Year of first value in this row.
- ▶ Ten data value blocks consisting of four integer characters for the data value, then a space, then two integer characters for sample depth.

The stop marker in a CRN file should be 9990.

U.4 Workarounds and quirks

▶ No information was given as to how to handle the non-existent year 0AD/BC. For data files with years all in the AD period, this is not a problem. Most dendro software seem to treat year numbers in Tucson files as using the 'Astronomical Calendar' whereby 1 = 1AD, 0=1BC, -1=2BC etc. This goes against what most dendrochronologists assume (and do) when using Tucson files. For instance most people that work entirely in the BC period use negative integers to represent BC years e.g. -5 as 5BC. With no clear specification and different people interpreting the format in different ways, there is no way of being certain what data negative year numbers in Tucson files mean.

▶ Tucson format places a restriction of just four characters to the year values. This means that strictly the earliest value a Tucson file can represent is -999. Some users work around this by steeling the last character of the series ID to give them five characters for the year. For example: ABCDEFG-9999. This conversely limits the series ID to 7 characters. To add to the confusion, other users have been known to add an arbitrary number (e.g. 5000) to all year numbers to overcome this problem.

▶ The fact that 999 is used as the stop marker for series in 1/100th mm means that Tucson files cannot store a ring value of 9.99mm. In the unlikely event that a sample should have this large a ring, it should be rounded up or down to 998 or 1000.

▶ Some programs appears to add padding values after the stop marker to fill the rest of the 10 data values in the row.

▶ Some data files seem to use 9990 as a stop marker

▶ Some files appears to use a full-stop character to indicate empty data values after the stop marker.

▶ Data values in RWL files are space delimited, however some programs use tabs instead.

▶ When reading Tucson files, COFECHA and ARSTAN ignore all lines that do not match the standard data line format. As such, some users have used this to enable them to include multiple comment lines in their files.

▶ The ITRDB documentation says they should be recorded as DDMM or DDDMM, but this along with sign (N,S,E,W,+ or -) would require 11 characters, when the Tucson specification only allows for 10. Perhaps this was due to an assumption that all places would be in the northern hemisphere? This has resulted in a large amount of variation in the way that coordinates are recorded making it extremely difficult to parse them without error. Here are some examples (including some that use 11 chars not 10):

- 4652N01101E
- +4652-01101
- N4652E01101

- 4652-01101
- 465201101
- 4652 01101

U.5 Example file – raw series

```
107       1 OBERGURGL
107       2 AUSTRIA       NORWAY SPRUCE      6726     4652N01101E       1911 1959
107       3 GIERTZ                                                       08   76
107011   1911      78    93    43   100    93   110   135   115   102
107011   1920      92   125   110   135    98    80    75   125   102   110
107011   1930     105   105    95   120   135   140   110   120   130   135
107011   1940     120   130   130   165   135   145   155   160    88   135
107011   1950     140   150   140   130   115   130   130   110   110   135
107011   1960     125   120   135   160    15   102   105   135   105   140
107011   1970     120   115   100   110   110   999
107012   1862     450   580   550   480   620   420   390   420
107012   1870     360   370   300   360   470   460   410   430   510   500
107012   1880     500   510   500   410   380   430   340   380   350   400
107012   1890     290   260   270   320   340   370   330   310   240   170
107012   1900     280   300   300   310   350   400   300   280   280   180
107012   1910     190   290   270   210   230   300   220   360   240   260
107012   1920     200   270   250   230   270   210   160   210   220   200
107012   1930     170   250   200   130   140   210   210   180   190   180
107012   1940     170   180   190   190   190   200   190   180   110   180
107012   1950     220   230   180   220   200   240   220   210   240   999
```

Tucson Compact

Format name	Tucson Compact
Other name(s)	Compact
Type	Text file
Extension(s)	rwm
Read/write support	Read and write
Reference implementation	Various DPL programs including FMT
Data / metadata	Data only
Calendar type	Astronomical
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Richard Holmes

V.1   Description

The Tucson Compact format was design by Richard Holmes for use with a number of the applications in the Dendro Program Library (DPL). Holmes designed it as a space saving alternative to the standard Tucson format at a time when disk space was expensive. The format never really caught on, perhaps due to the complexity and variability of the format.

The key feature of Tucson Compact format is the inclusion of a code that describes the layout of the data within the series. This code means that only the required amount of space is allocated to each data value in the text file with little wastage. No space is provided for metadata.

Tucson Compact files can contain one or more series of data so the description of a data series below can be repeated multiple times in a single file. All lines should be 80 characters long and the first line of a series is denoted by a tilde ( ) in the final column. This meta line contains four fields:

- ▶ Chars 0-9 = number of data values terminated with =N
- ▶ Chars 11-19 = start year terminated with =I
- ▶ Chars 21-68 = series title
- ▶ Chars 69-79 = fortran format descriptor
- ▶ Char 80 = Tilde marker

The Fortran format descriptor in the example below is -2(26F3.0). The constituent parts are as follows:

▶ -2 = this is the scaling factor for the data values. In this case -2 = 10-2 = 0.01. Please note that in the Convert5 program this scaling factor is only read once in the first header line so files with multiple series each with different scaling factors will read incorrectly.

▶ 26F = means there are 26 values in each line

▶ 3.0 = means that each data value should be read as 3 integer values

The example below therefore means there are 26 data values per line each consisting of 3 digits which should be interpreted by multiplying by 0.01 (i.e. values are in 1/10ths mm).

V.2   Example file

```
1       176=N      1277=I ABCD01                                                    −2(26F3.0)˜
2 142338334409362308360264325318134151219268290222782581731982942021701761722121
3  871301141081701351311268 87100 86104103127112 94 96120168149119124 79 67 88 90
4  93 77 49 42 53 38 57 43 50 41 56 66 62 55 55 45 47 63 58 60 44 45 49 50 62 61
5  43 54 91 60 56 43 52 51 65 68 55 44 41 75 94 78 63 69 58 75 55 47 58 46 62 45
6  52 50 77 50 63 75 77 64 66 57 80 57 78 65 68 75 65 98 85 82119 89 85 87 83108
7 129123160117129121 88 69 97 77 96106 71 89 50 65133 89 88 50 60 95 95 91102158
8  83 55 98 70 45 46 40 36 64 58 52 58 56 94 51 48 47 60 49 48
```

VFormat

Format name	VFormat
Other name(s)	OJ Format
Type	Text file
Extension(s)	Various depending on data type but commonly !oj
Read/write support	Read and write
Reference implementation	VFormat
Data / metadata	Data with some structure metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	No
Relative dating support	No
Multi series support	Yes
Original designer	Thomas Reimer and Hans-Hubert Leuschner

W.1   Description

A relatively extensive format which includes highly encoded header lines for metadata. VFormat files have an array of file extensions depending on the type of data the files contain.

VFormat files can contain mutliple data series. Each series contains 2-4 header lines followed by a number of data lines. The metadata fields are encoded into the header lines in specific character positions. In line 1 the character positions are as follows:

- ▶ 1-12 = Series identifier. The series identifier also determines the filename. If there is just one series in the file then the series identifier will be the same as the filename. For files with multiple series, the filename will use characters 1-7 of the series identifiers that are the same throughout the file with the remaining (different) characters replaced by an underscore. The 8th character of the filename would contain a running number for files that would otherwise be named the same. The series identifier is made up of the following characters:
  - – 1 = Code representing the project or country
  - – 2 = Code representing the region of ecological area
  - – 3-4 = Code number for sample site (optionally encoded using hexadecimal or hexatresimal to enable values greater than 99).
  - – 5-6 = Series/tree number (optionally encoded using hexadecimal or hexatresimal to enable values greater than 99).
  - – 7 = Height code encoded as follows: 1 = 1m, 2=2m, 9=9m, A=10m, B=11m, S = Lumber height 30cm, T = breast height =130cm.

- 8 = Running number if several series have the same values in columns 1-7.
- 9 = Fixed as a dot character
- 10 = Either ! (single), % (partial), # (mean curves or chronologies)
- 11 = Code for statistical treatment. One of F (frequency filtered series); I (index); M (mean); O (original); P (pointer-year stat); Q (cluster-pointer-year stat); R (residual); S (moving deviation or variance); T (trend, fitted curve, model); W (wuchswert); X (series with standardized running mean and variance); Z (central moment, deviation or variance between several series).
- 12 = Code for the measured parameter. One of D (mean density); F (earlywood width); G (maximum density); J (ring width); K (minimum density); P (percentage latewood); S (width of latewood).
▶ 13-15 Measurement units
▶ 16-20 Length of series
▶ 21-24 Species either encoded using ITRDB taxon codes or by using the first two letters of the genus and species.
▶ 25-30 Year of the last ring
▶ 31-50 Description
▶ 51-58 Measurement date (ddMMyy or ddMMyyyy)
▶ 59-60 Initials of author
▶ 61-68 Last modified date (ddMMyy or ddMMyyyy)
▶ 69-70 VFormat version identifier (00,01 etc)
▶ 71-73 Estimated number of missing rings as the start of the series
▶ 74-75 Standard error of this estimate (. if unknown)
▶ 76-78 Estimated number of missing rings at the end of the series
▶ 79-80 Standard error of this estimate (. if unknown)

The second data line is a free text comment up to 80 characters.

VFormat files from version 10 onwards then contain a third header line. This contains 8 floating point numbers of 10 digits each. These represent:

▶ Longitude
▶ Latitude
▶ Altitude
▶ Height of the tree's measurement
▶ Four other user definable numbers

VFormat files from version 20 onwards contain a forth header line. This is of the same format as line 3 but each of the values is user definable.

Following the 2-4 header lines come the data lines. These lines are made up of 10 data fields each containing 8 characters. Each data field is made up as follows:

▶ Two character code for validity and importance:
  - space = full validity
  - ! = not yet used
  - " = not yet used
  - # = not yet used
  - $ = no validity for long-term evaluations
  - % = no validity for single-value evaluations
  - & = no validity except for cumulative stats
  - ' = no validity at all, unknown value

The second character is a pseudo-binary character used to define a weighting factor. For full details of the complex method for calculating this weighting factor see the VFormat documentation.

▶ One character user definable code for recording information about the data value
▶ Five digit floating point data value which is divided by 100 for interpretation

W.2   Example file

```
G1101020 .!OJmm      81Qusp  1510FLA−02  32 /572              HL01.04.9002   810 10 .
G1101020 .!OJ/S20102_0 .!OJ/_
     281      221      225      169      178      197      126      103      112      130
     132      207      176      175      126      150       99      131      187      204
     218      172      202      115      135      130      196      135      142      129
     144      116       92       71      109      120      137       98       86      117
      64       79       72       61       62       82       75       81       83       69
      83       66       84       95       85       94       87       99       92      109
     150      108       70      113      119      120      122      107      111      114
     123      145      112      145      164      158      122      177      155      182
     153
G1101050 .!OJmm     121Qusp  1516FLA−05  13 /586              HL01.04.9002  1510 13 .
G1101050 .!OJ/S20105_0 .!OJ/_
     448      286      341      213      346      371      745      719      580      466
     487      353      279      323      422      436      351      238      135      172
     179      210      277      145      165      261      263      190      194      183
     127      110      144      189      135      154      217      110      115       99
     106      101      106      198      191      185      185      160      112      152
      93       95       83      176      165      193      139      101       93      113
      85      145      174      157      132      130       74       52      114      138
     174      132      144      125       83      124      118      127      150      189
     152      133      117       91      104       96       56       90      130      126
     103      163       92      103      174       99      117       85      123      116
     147      127      145      133      155      144      114      115      121      111
     174      113      112       89       99      130      111      104      164      110
     139
```

WinDENDRO

Format name	WinDENDRO
Other name(s)	None known
Type	Text file
Extension(s)	txt
Read/write support	Read only
Reference implementation	WinDENDRO
Data / metadata	Data with some structured metadata
Calendar type	Gregorian
Absolute dating support	Yes
Undated series support	Yes
Relative dating support	No
Multi series support	Yes
Original designer	Regent Instruments

X.1 Description

WinDENDRO format is a dendro text file format designed by Regent Instruments for their WinDENDRO software. Regent Instruments claims the format is proprietary. Although it is unclear whether such a claim is legally binding for a plain text file, the authors of DendroFileIOLib have decided to comply by not implementing a WinDENDRO format writer. However, in the interests of the dendro community and to ensure users can gain access to their data, DendroFileIOLib does include support for reading WinDENDRO format files.

WinDENDRO files differ from most other formats in that they contain a great deal of information specific to the image used to measure the sample. The WinDENDRO software allows users to measure ring widths from scans or photographs of samples rather than by using a traditional measuring platform.

WinDENDRO files are really just tab-delimited text files with data in columns in a specific order with a few additional header lines.

Line 1 should contain 8 tab-delimited fields

- ▶ Field 1 = WINDENDRO
- ▶ Field 2 = WinDENDRO file format version number, either 3 or 4
- ▶ Field 3 = Orientation of the data: R = in rows; C = in columns. All WinDENDRO files are in rows
- ▶ Field 4 = The column number where the data values begin. For version 3 files this is 13 and version 4 files this is 36
- ▶ Field 5 = The direction the data is recording in: P = pith to bark; B = bark to pith

- ▶ Field 6 = Whether the data is recorded incrementally (I) or cumulatively (C). WinDENDRO files are always incremental.
- ▶ Field 7 = Whether the bark width has been measured (Y or N). If yes, then there will be one more data value than there are rings
- ▶ Field 8 = RING

Line 2 contains the field names. For version 3 files these are:

- ▶ TreeName - The name of the tree being measured
- ▶ Path identification - ID of the path along which the series is measured
- ▶ Site identification - Name of the site from which the tree was taken
- ▶ YearLastRing - Year of the last ring in the series
- ▶ Sapwood - Distance (in mm) from the start of the sample to the start of the sapwood.
- ▶ Tree height - Height of tree in metres
- ▶ Tree age - Age of the tree. If unknown this should be 0, then it is assumed to be equal to the number of rings
- ▶ SectionHeight - Height up the tree in metres at which the sample was taken
- ▶ User variable - User defined variable - must be numerical
- ▶ RingCount - Number of rings the series contains
- ▶ DataType - Keyword indicating the type of data measured. This can be: RINGWIDTH; EARLYWIDTH; LATEWIDTH; EARLYWIDTH%; LATEWIDTH%; DENSITY; EARLYDENSITY; LATEDENSITY; MAXDENSITY; MINDENSITY; RINGANGLE.
- ▶ OffsetToNext - The number of lines to skip to go to the next data line of the same type. For instance a file can contain earlywood and latewood data for multiple samples. If this is the case then each sample will have two rows, one for each variable, and the OffsetToNext field will be 1.

In additional to these fields, version 4 files also include the following:

- ▶ ImageName - The filename for the image used to do this analysis. If the image was taken directly from the scanner or camera then this field will be SCANNER
- ▶ Analysis Date Time - Date and time the measurements were initially saved to disk in format dd/mm/YYYY HH:mm
- ▶ Acquisition Date Time - Date and time the image file was acquired in format dd/mm/YYYY HH:mm
- ▶ Modified Date Time - Date and time the file was last modified in format dd/mm/YYYY HH:mm
- ▶ ImageSize H V NBits Channel - The image size in pixels followed by bits per pixel per channel (8 or 16), channel used for analysis (Grey, RGB, R G or B)
- ▶ CalibMethod XCal YCal EditedDendro - Method of calibration: Intr (Intrinsic); Obj (ObjKnownDiam). This is followed by the size of a pixel and Y or N indicating if the image has been edited in WinDENDRO
- ▶ ScannerCamera Make Model Software - Details about the imaging hardware
- ▶ LensFocLength [35mm] - The 35mm equivalent focal length of the imaging lens
- ▶ PathBegX BegY EndX EndY Width - The coordinates for the start of the path/radius followed by the path width
- ▶ RingBoundary AutoMan Meth Precise - Details about the path taken. Ring boundary - Tg (tangent to ring) or Perp (perpendicular to path); Detection method - A (automatic) or M (manual); Ring detection method - Int (intensity differences) or T&S (teach and show); whether the 'more precise detection' method is active (Y) or not (N)
- ▶ EarlywoodDef - Earlywood-latewood transition criteria
- ▶ DensActive Media Calib - Density Analysis active (Y or N); Density Media setting (F - negative file or photo, W wood direct xray, positive film or photo); Light calibration setting (Acq - after image acquisition, Man - manual; No - none)
- ▶ DensNSteps MatDens Interpol - Number of steps and the density of the step wedge used for calibration followed by the interpolation method used between steps: Lin (Linear) Spl (Spline)
- ▶ DensStepsThick - The thickness of each step of the wedge used for density calibration

- ▶ DensStepsLightInt - The light intensity of each step of the wedge determined during the light intensity calibration
- ▶ DensStepsWoodDens - Equivalent wood density of each step of the wedge determined during light intensity calibration
- ▶ DiskArea - Area of the sample
- ▶ DiskPerim - Perimeter of the sample
- ▶ DiskAvgDiam - Average diameter of the sample
- ▶ DiskFormCoef - Sample area form coefficient
- ▶ CompWoodArea - Total area occupied by the compression areas
- ▶ VoidArea - Total area occupied by the void areas
- ▶ PathLength - Length of radius measured

Lines 3+ contain the actual data and metadata, one line for each series. Following the 13 or 36 columns of metadata (depending on file version) there are x number of columns containing ring values. The values are recorded as floating point data. The units for these data values are: mm for widths; % for percentages; g/cm^3 for densities; radians for angles.

XML Error Codes

Table Y.1: The Tellervo webservice provides error feedback by means of an error code and description.

Section	Code	Description
General	001	Error connecting to database
	002	Generic SQL error
Authentication	101	Authentication failed
	102	Login required
	103	Permission denied
	104	Unsupported request
	105	Invalid server nonce
	106	User unknown
	107	Unsupported client
	108	Unsupported client version
	109	Password needs to be updated
	110	Password upgrade required
Miscellaneous	666	Unknown Error
	667	Program bug
Internal	701	Internal SQL error
	702	Feature not yet implemented
	703	Invalid XML being returned by webservice
	704	Configuration error
User	901	Invalid user parameter(s)
	902	Missing user parameter(s)
	903	No records match
	904	Parameters too short
	905	Invalid XML request

Table Y.1: The Tellervo webservice provides error feedback by means of an error code and description.

Section	Code	Description
	906	Record already exists
	907	Foreign key violation
	908	Unique constraint violation
	909	Check constraint violation
	910	Invalid data type
	911	Series with this version number already exists
	912	There must be at least one administrator

GNU General Public License

The Tellervo server and desktop client are released under the GNU General Public License (GPL) version 3.

Copyright © 2016 Peter Brewer

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Z.1   Preamble

Copyright © 2007 Free Software Foundation, Inc. `http://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Z.2 Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in

accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim

or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHER-WISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

References

Aniol, R., 1983. Tree-ring analysis using CATRAS. Dendrochronologia 1, 45–53.

Baillie, M., Pilcher, J., 1973. A simple crossdating program for tree-ring research. Tree-Ring Bulletin 33, 7–14.

Brewer, P., Murphy, D., Jansma, E., 2011. TRiCYCLE: a universal conversion tool for digital tree-ring data. Tree-Ring Research 67, 135–144.

Brewer, P., Sturgeon, K., Madar, L., Manning, S., 2010. A new approach to dendrochronological data management. Dendrochronologia 28, 131–134.

Brewer, P.W., Guiterman, C.H., 2016. A new digital field data collection system for dendrochronology. Dendrochronologia 38, 131 – 135.

Cook, E., Peters, K., 1981. The smoothing spline: A new approach to standardizing forest interior tree-ring width series for dendroclimatic studies. Tree-Ring Bulletin 41, 45–53.

Cormen, T., Leiserson, C., RL, R., C, S., 2001. Introduction to Algorithms. The MIT Press.

Huber, B., Fletcher, J., 1970. Dendrochronology in Europe.

Jansma, E., Brewer, P., Zandhuis, I., 2010. TRiDaS 1.1: The tree-ring data standard. Dendrochronologia 28, 99–130.

Knibbe, B., 2008. PAST4 - Personal Analysis System for Treering Research software version 4.5. SCIEM.

Reinsch, C., 1967. Smoothing by spline functions. Numerische Mathematik 10, 177–183.

Rinn, F., 2008. TSAP-Win software. Rinntech, Heidelberg.

Tyers, I., 1999. Dendro For Windows program guide. ARCUS Report 500.

Van Loan, C., 1999. Introduction to Scientific Computing: A Matrix-Vector Approach Using MATLAB. Prentice Hall.

Wigley, T., Jones, P., Briffa, K., 1987. Cross-dating methods in dendrochronology. Journal of Archaeological Science 14, 51–64.

Index