

# **HANDBOOK FOR MANAGEMENT OF CTFS DATABASES**

**Written by CTFS Systems Group April 2014**

**Suzanne Lao, Shameema Esufali, Anudeep Singh,  
Richard Condit**

**Contact for further information**

**[shameemaesufali@gmail.com](mailto:shameemaesufali@gmail.com)**

**[laoz@si.com](mailto:laoz@si.com)**

# Table of Contents

CHAPTER 1. Overview of Data Management Protocols.....	5
I.Introduction.....	5
II.Role of the Data Manager .....	5
III.Common platform for software .....	5
IV.Common database structure for all sites .....	6
V.Shared higher level taxonomy.....	6
VI.Data Entry and Verification Tools and Protocols.....	6
VII.Exporting data for analysis .....	7
VIII.Backups and data storage .....	7
IX.Configuration Management Policy for CTFS Datasets.....	8
X.Managing Databases on a Server. ....	9
CHAPTER 2. Formatting input files to CTFS requirements.....	10
I.Introduction.....	10
II.Format Data Files.....	10
1.Species file.....	10
2.Tree measurement codes.....	12
3.Quadrat file.....	12
4.Role Reference file.....	13
5.Personnel file .....	13
III.Census data for uploading via ctfswb.....	13
CHAPTER 3. INSTRUCTIONS FOR SETTING UP A CTFS MySQL DATABASE .....	15
I.Introduction.....	15
II.Installing Apache, MySQL, and PHP.....	15
III.Creating the Site Database .....	15

IV.Installing the ctfsweb System.....	17
V.Uploading your Site Data into the MySQL Database.....	17
1.Loading your site's fixed data .....	17
– COUNTRY DATA – .....	17
– SITE DATA - .....	17
– CENSUS DATA – .....	18
-- MEASUREMENT CODES – .....	18
-- PERSONNEL – .....	18
-- SPECIES DATA --.....	20
-- QUADRAT DATA – .....	20
2.Loading your site's census data .....	21
CHAPTER 4. GUIDE TO CTFSWEB .....	23
I.Creating a Configuration File with ctfsweb .....	23
II.Entering Data into the Data Forms with ctfsweb .....	26
III.Double Data Entry with ctfsweb.....	27
IV.Single Data Entry .....	28
II.Inserting Location Data.....	29
V.Backing Up Files.....	29
VI.Screening Data in Temporary Tables.....	30
VII.Uploading Data into Database and Creating ViewFullTable .....	30
CHAPTER 5. Guide to Correcting Data Errors.....	32
I.Field Staff .....	32
1.What is a data error?.....	32
2.When can you correct it? .....	32
3.Who is authorized to correct a data error for your site? .....	32

4.How do you submit your corrections? .....	32
II.Data Managers.....	33
1.How to correct data errors .....	33
III.Script writing.....	34
1.What is a script .....	34
2.Why do we write scripts to correct errors .....	34
3.Text editors and language standards for CTFS data manipulation .....	35
4.Base data .....	35
5.Multiple scripts.....	35
IV.Creating a package .....	35
6.What should be in a package .....	35
V.How to test a package before submission .....	36
VI.Sample scripts – This chapter is in development .....	37
1. Insert new records.....	37
2. Update existing records.....	37
3.Delete existing records.....	37
Change tag.....	37
Change Stemtag .....	37
Change DBH.....	37
4.Delete existing records.....	37
CHAPTER 6. Digitizing and Uploading New Plant Coordinates .....	38
I.Installing ImageJ and Point Picker .....	38
II.Using ImageJ to Digitize Plot Maps.....	38
III.Converting ImageJ Text Files from Pixels to Meters.....	40
Appendix to Chapter 2. Table Descriptions .....	42

Appendix to Chapter 4. Troubleshooting long fieldsheets not saved. .... 45

# CHAPTER 1. Overview of Data Management Protocols

## I. *Introduction*

This Chapter is an overview of the processes and protocols used to manage CTFS-ForestGEO data. You need to think about who does these jobs at your site and how to make sure protocols are followed to keep your dataset to standard.

## II. *Role of the Data Manager*

The Data Manager will be managing the data collected at the site.

In general, this involves:

1. Assuring the quality of the data collected by screening it as often as needed
2. Creating forms and data entry screens that try to minimize collection and data entry errors
3. Trying to screen data as soon as it is collected, so errors can be fixed in the field if necessary
4. Keeping records of all errors found and how they were fixed
5. Scanning field forms (if they are in hard copy format), and backing up all temporary electronic work files, in a timely fashion
6. Uploading all data into the production database
7. Backing up the final database after a census
8. Making changes to database between censuses as necessary, keeping records, backing up all versions
9. Archiving final database in a repository

## III. *Common platform for software*

CTFS sites use free open source software as a standard. The database environment comprises of Apache, MySQL and PHP. Basic database management and simple queries can be completely managed in this environment with the tools provided. Once these are installed, site managers can install and use CTFS software to enter, screen and verify, correct, manage and analyse data.

Detailed instructions for installing all software are given in the Installation Guide below.

1. Install Apache, MySQL, PHP. In the case of Windows and Mac, install XAMPP, which is very user-friendly  
<http://www.apachefriends.org/en/xampp.html> or  
<http://sourceforge.net/projects/xampp/>
2. Install ctfsweb

[http://ctfs.si.edu/Public/software/ctfsweb/latest/Installation/Installation Guide.pdf](http://ctfs.si.edu/Public/software/ctfsweb/latest/Installation/Installation%20Guide.pdf)

#### IV. *Common database structure for all sites*

CTFS sites use a common database structure. In order to include your site in the network the data manager needs to:

- Install the tables and import shared taxonomic data
- Import census data or enter data using ctfsweb, a complete system for data entry and verification of field data.

#### V. *Shared higher level taxonomy*

- APG Family and Genus information is shared by all sites in the CTFS network.
- Species data is specific to each site.

Detailed instructions for setting up your site database, configuring taxonomic data and importing or entering your site data are available in:

[http://ctfs.si.edu/DatabaseGroup/final/INSTRUCTIONS FOR SETTING UP A CTFS MySQL DATABASE\\_31Oct2013.pdf](http://ctfs.si.edu/DatabaseGroup/final/INSTRUCTIONS%20FOR%20SETTING%20UP%20A%20CTFS%20MySQL%20DATABASE_31Oct2013.pdf)

#### VI. *Data Entry and Verification Tools and Protocols*

CTFS provides tools for data entry and verification of census data. With ctfsweb you can.

- Enter Census data into temporary tables using single or double entry
  - Import previously entered data from text files
  - Screen your data and correct data entry errors by consulting your field sheets.
  - Upload your data into the permanent database tables.
- 
- If there are outstanding errors that cannot be corrected at this point due to lack of information you must create scripts to correct these errors according to a scripting protocol.

A complete user guide and FAQ to ctfsweb are available at

User Guide

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/CTFSUserGuidetoDataEntryandManagement.odt](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/CTFSUserGuidetoDataEntryandManagement.odt)

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/CTFSWEBDATAENTRYFAQsandTIPS.pdf](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/CTFSWEBDATAENTRYFAQsandTIPS.pdf)

## VII. *Exporting data for analysis*

The CTFS data structure includes a consolidated (denormalized) view of all the data. This view table is called ViewFullData.

The REPORTS module of ctfswweb runs off this table. This table may be exported into a flat file for use with any statistical software.

In particular the CTFS systems group uses it to create R Analytical Tables for all sites. These tables are formatted for use with the CTFS R Package, a powerful package designed specifically to analyze plot data.

The CTFS R Package is available at

<http://ctfs.SI.edu/Public/CTFSRPackage/>

## VIII. *Backups and data storage*

1. Back up the database using mysqldump

eg. `mysqldump -u your_mysql_username -p database_name > name_saved.sql`

2. Error corrections should be accumulated and updates to the database done according to a schedule, not every time an error is found. Any updates to the database should be recorded in a script. Updates include saving the original data from the record in the Log table before making the change to the record.
3. The database should be backed up every time a set of corrections are done. The update scripts should be saved along with the database backup.
4. The official database, or the database to be used for analyses and publications, should be updated only once a year or less (once every two years for example).



5. Official databases should be archived in a digital repository. Repositories maintain your files and ensure that they remain usable in the future. Repositories also provide online access to papers and data for the research community, and can serve as a method of publishing files and data, making them more easily citable.
6. Most repositories are public access, but some may restrict access for a few years, or store some parts publically and maintain other more proprietary parts inaccessible.
7. The BCI data is archived in the Smithsonian Digital Repository in several formats: MySQL, R tables, and ASCII. It is especially important to save files in ASCII format as MySQL and R may not be maintained in the future. A detailed metadata is necessary to explain what data is collected, the quality of the data, etc.

#### ix. *Configuration Management Policy for CTFS Datasets*

- We require that there be 2 versions of each database. The names of the databases should have the following prefixes:
  - stable\_ - this is the official version. Analyses and publications should be based on this version. Updates to this database should not happen more than once a year.
  - current\_ – updates and changes can be done to this database throughout the year. Every time an update is done, this database should be backed up. This allows the manager to go back to a previous copy if he makes a mistake updating the current database. Once a year (or once every two years) this database will be copied over to the stable database.
- By convention we name test databases working\_ – this database is used for testing scripts or programs. These databases are created explicitly for testing and dropped once the database is finalised to either a current or stable database.

We require that the backups include the date of the backup as part of the file name. DO NOT overwrite previous backups, all backups should be saved. This allows analyses from publications to be repeated on a previous version of a database if required, as well as allows us to go to a previous database version in case errors were committed during an update.

#### x. *Managing Databases on a Server.*

Managing databases on a server requires in most cases having a working knowledge of the unix operating system. The manager will have to set up usernames and passwords and restrict

access to different databases to different users. The database manager will also have to control the type of access to different users on the same database, i.e. complete privileges versus restricted privileges (such as querying the database but not allowing changes to be made).

Cron jobs can be set up to mirror databases on other servers as needed, i.e. once a year or once every two years.

Cron jobs can be set up to backup certain databases or files automatically when necessary.

The server should also have a backup system to backup up all files and databases on the server. Most institutions use tape backups and these can be programmed to back up as frequently as needed.

## **CHAPTER 2. Formatting input files to CTFS requirements**

### *I. Introduction*

Data intended for input into the CTFS Data Management systems need to be presented in a specific way. This chapter details the files you need to submit for forest plot data and also specifies their ideal format. If your data is in a different format or you have additional information that you would like to preserve in the system, please contact the CTFS Systems Group with details.

You will need to submit these data files for review prior to upload, either at a workshop or by email or dropbox.

All data files should be tab-delimited text files with a header using the column names outlined below. Avoid special characters and quotes (single or double).

A complete data set requires:

- 1.A species file
- 2.A list of all tree measurement codes used in the censuses
- 3.A quadrat file
- 4.A file with the names of the personnel involved with the census.
5. A file with staff roles defined
- 6.If you have already completed a census, one file with the tree and stem measurements for each census; that is, with 3 different censuses, 3 different files

Following is a detailed explanation of the format for each of the above files. You may include additional data that needs to be stored but each file must have the columns specified in the formats below.

### *II. Format Data Files*

For the initial census you will need permanent static data such as your country and site description including the global coordinates of your plot, its size and shape, and files containing quadrat data, species and codes lists, etc. Finally, for any census you will need a census file with dbh measurements for each stem. At recensus you must also submit any changes to the fixed data, such as new species, additional codes etc. This section specifies the files and formats for these data.

## 1. Species file

The columns spcode, genus, species, and idlevel are required, the other two are optional.

**spcode** - a code used in the field to identify the species of the tree. At CTFS, most sites use 6 letter codes where the first 4 letters are from the Genus name and the last two from the species. If two species yield the same code then the first unique part of letters is used to break the tie.

For example I have used the first character of the species names together with the first unique character to form the 6 character code. Eg's

For *Shorea macroptera subsp. baillonii* and *Shorea macrophylla*, both would be SHORMA. The species codes ended up being SHORMB and SHORMC, respectively.

You should use a similar naming convention for each morphospecies incorporating details you know. For example, using LITSBL (Litsea “Big Leaf”) or APORS1 (Aporosa sp. 1) are fine – as long as each code applies to only one morphospecies. These can be changed once identification is more complete.

Other combinations are also acceptable. Other sites use 3 letters from the genus and 3 from the species, while others use 4 letters instead of 6 (2 letters from the genus and 2 from the species).

**genus** – the taxonomic genus name (in case of unknown genus, use “Unidentified”)

**species** – the taxonomic species name

**family** – the taxonomic family name

**authority** – (optional) author of the species name

**idlevel** – to indicate to which level the species or morphospecies has been identified. Options are as follows:

- ✓ species – species known.
- ✓ subspecies – subspecies known.
- ✓ genus - genus known, species unknown but single morphospecies
- ✓ family - family known, genus and species unknown but single morphospecies
- ✓ none – no identification has been done. But single morphospecies
- ✓ multiple – not a single morphospecies, but one of a group of a certain genus or family.

## 2. Tree measurement codes

These are the codes used by field personnel to describe the condition of a tree, stem or measurement. A few are required for data processing and others are optional and may be site specific. Examples of optional codes are L – Leaning tree, BA – stem Broken Above point of measurement.

For each code you need to supply a short code and a longer description.

**code** – one or more letter codes used in the codes field in the census tables

**description** – a brief description of what the code means

Please check that this is a complete list of all the codes used in all the census datasets.

The following descriptions are required in your list (the code you decide to use is up to you):

a. **main** – to indicate the main stem in multiple-stemmed trees; that is, one of the codes should have description main

For datasets with more than one census:

b. **dead** - this code should indicate that the entire tree is dead. Try not to use this word in the description of any other code. If only one stem is “dead” and there are other live stems, use a description such as “stem lost” (or some other description).

c. **stem lost** - to indicate that a stem that was previously 1 cm dbh or above broke off and is still alive but is < 1 cm dbh

## 3. Quadrat file

This contains a complete list of the quadrat names used in the plot. The following fields are required.

**quadrat** – the name of the quadrat, eg. 0002

**startx** – the x coordinate of the lower left corner of the quadrat, eg. 0

**starty** – the y coordinate of the lower left corner of the quadrat, eg. 40

**dimx** – the x dimension of the quadrat, eg. 20

**dimy** – the y dimension of the quadrat, eg. 20

The x and y coordinates (startx and starty) refer to the distance in meters of the quadrat from the lower left corner of the entire plot.

Most of the sites have 20m x 20m quadrats and use the same naming system (0000, 0001, ....., 0024, 0100, 0101, ....4900, 4901, ....4924) for a 1000m x 500m plot.

However, some sites may use 10x10m quadrats or name your quadrats starting from 0101 (no 00 used). There is at least one site where quadrat 0000 refers to a quadrat in the center of the plot.

This table should reflect the names used in your plot, as long as the startx and starty of the quadrat indicate where this quadrat is in relation to the lower left corner or the entire plot, assuming that this lower left corner has x,y coordinates 0,0.

#### **4. Role Reference file**

This file should have a list of the job titles involved in the collection and management of CTFS plot data. It needs to contain only a single column. Roles typically include: field technician, data entry technician, supervisor, student, volunteer, as well as data manager and principal investigator, among others.

**role** – the role the person played in the census.

#### **5. Personnel file**

This file contains the names of the people who are or were involved with the plot, and the role that they played. The first and last names should be separate.

**firstname** – the first name of the person

**lastname** - the last name of the person

**role** – the role the person played in the census. This should match exactly one of the descriptions in the RoleReference table.

If a person has more than one role (for example, he was a field technician in one census, then promoted to field supervisor in a later census), then that name should be entered twice.

NOTE: this file is necessary if using ctfswb to enter census data from field sheets.

### *III. Census data for uploading via ctfswb*

Bring the tree data from each census in a separate file. Each file must have the fields listed below. The columns can be in any order. You may have extra fields in the dataset, but they will not be uploaded into the tables of the database.

**tag** – the tag of the tree (should be unique)

**stemtag** – the tag of the stem. If your site does not use stem tags, you may leave this column blank. The header however, should include this variable name.

**spcode** – the species code of the tree. All species codes should appear in the species file.

**quadrat** – the name of the quadrat the tree is located in

**lx** – the x coordinate in meters of the stem within its quadrat.

**ly** – the y coordinate in meters of the stem within its quadrat.

**dbh** – the diameter of the tree. If there is no diameter measurement (missing, dead, or resprout), please put **NULL**.

**codes** – tree or measurement codes. If there is more than one code, they have to be delimited with semicolons. This allows for codes with more than one letter. Each and every code should be accounted for in the Codes table in (2) above. The codes field may be left blank if there are no codes.

**hom** – height (in meters) where the diameter was measured, if different from 1.3 m. You may leave this field blank if the stem was measured at 1.3 m, and just fill it in when the hom is different from 1.3

**date** – date the stem was measured. This date should be in **yyyy-mm-dd** format. Example, 2011-02-24.

Note that all the multiple stems should be included in these files – you may indicate in the codes field which one is the main stem. If the tree only has one stem, you do not have to include the main stem code. The rest of the information should be repeated for each multiple stem - make sure that the information (species code, date, etc.) is the same for all multiple stems of the same tree.

The dataset for the first census should only contain trees and stems tagged and measured from that census. The dataset for subsequent censuses should contain stems from the previous census. Dead or lost stems should have the appropriate codes.

## CHAPTER 3. INSTRUCTIONS FOR SETTING UP A CTFS MySQL DATABASE

### I. Introduction

This chapter outlines the process of setting up a new system. It begins with the installation of the programs needed on your computer to use the CTFSWeb Data Management Systems and goes on to a step by step guide to creating a plot database.

### II. Installing Apache, MySQL, and PHP

You need to install Apache (a web server software program), MySQL (an open-source relational database management system), and PHP (a programming language designed for web development) in your computer. There are several packages available, but we suggest you use XAMPP, which is relatively user-friendly. To install XAMPP, please follow the instructions in the document *Installation Guide ctfsweb.pdf* up to step 3. <http://ctfs.si.edu/Public/software/ctfsweb/latest/Installation/Installation%20Guide.pdf>

### III. Creating the Site Database

To create the site database (step 4 of the Installation Guide), do the following.

Open a MySQL console in your computer with the following steps:

On your Desktop, click on Start, then Run

Type: `cmd`, then click on OK

In the Command Prompt Window, change to the mysql folder by typing:

```
cd\xampp\mysql\bin
```

Go into mysql by typing: `mysql -u root`

If it asks for a password, just press ENTER (no password needed).

---

#### NOTE:

If the xampp directory is not in your DOS path and you want to be able call up mysql anywhere, do the following carefully:

Go to the Control Panel in your System Settings

Click on System

Click on Advanced

Click on "Environment Variables" in the bottom

In System Variables in the second window, scroll to Path

Add the following to the end of the existing path:



;C:\xampp\mysql\bin

IMPORTANT: You can do the above only if you are an administrator of your computer.

---

Once you have opened a mysql terminal, create a database by typing the following.

NOTE: Please replace the name of the database *current\_yoursitename* with the name of your database, eg. current\_bci. The names of the mysql commands are in caps, but they can be in small letters. The names of the databases and tables, however, are case-sensitive.

```
mysql> CREATE DATABASE current_yoursitename;  
mysql> USE current_yoursitename;
```

**If you already have a database that you dumped** (i.e.backed up), just source the resulting file:

```
mysql> SOURCE /path-to-file/yoursite.sql;  
eg. SOURCE C:/ctfs/databases/bci.sql;
```

NOTE: The path cannot contain spaces.

---

NOTE: For the following steps, the scripts can be downloaded from:  
[http://ctfs.si.edu/Public/DatabaseSetup/ctfsweb/database\\_instructions/table\\_scripts/](http://ctfs.si.edu/Public/DatabaseSetup/ctfsweb/database_instructions/table_scripts/)

---

**In the case that you have not entered any data yet and are not migrating from the old database structure**, insert the empty tables into the database:

```
mysql> SOURCE New_Database_Structure.sql;
```

Enter the taxonomic family and genus names from the APG system into the Family and Genus tables:

```
mysql> SOURCE Family.sql;  
mysql> SOURCE Genus.sql;
```

#### *IV. Installing the ctfsw eb system*

Once the database is created, install the ctfsw eb system by continuing on to Steps 5 through 7 in the *Installation Guide ctfsw eb.pdf*.

#### *V. Uploading your Site Data into the MySQL Database*

##### 1. Loading your site's fixed data

This step is required for all users creating an initial census dataset, whether you are using the ctfsw eb data entry module or uploading previously entered census data. Prepare your data for uploading into your database. Your data files should be in the format as described in the chapter *Format Data Files*.

Create the temporary tables needed for ctfsw eb by running the following script:

```
mysql> SOURCE createTempTables.sql;
```

**NOTE:** For a brief explanation of the variables in these tables, refer to the data dictionary in

[http://ctfs.si.edu/Public/DataDict/data\\_dict.php](http://ctfs.si.edu/Public/DataDict/data_dict.php)

A. Enter the COUNTRY information:

```
mysql> INSERT INTO Country (CountryID, CountryName) VALUES (1, 'Country');
```

B. Enter the SITE information:

Here you must substitute your own site information. This information can be displayed on reports etc.

Site description usually specifies forest type eg. "Upper montane tropical forest".

Shape can contain the length of each side for a rectangular plot or give particulars of the shape for an irregular plot.

Area specifies the number of hectares or square meters of your plot.

QdimX and QdimY refer to the dimensions of each quadrat.  
GUOM, GZUOM – units of measure for the global coordinates  
PUOM – units of measure for the plot coordinates  
QUOM – units of measure for the quadrat coordinates  
GCoorCollected, PCoorCollected, QcoorCollected – whether the global, plot and quadrat coordinates were collected, respectively  
IsStandardSize – whether the plot is standard shape, i.e. rectangular

```
mysql> INSERT INTO Site (PlotID, PlotName, LocationName, CountryID, ShapeOfSite,
DescriptionOfSite, Area, QDimX, QDimY, GUOM, GZUOM, PUOM, QUOM,
GCoorCollected, PCoorCollected, QCoorCollected, IsStandardSize)
VALUES (1, 'yoursitename', 'Your Site Location eg. South Western Province', 1,
'1000x500', 'Any site description you want to insert here', 500000, 20, 20, "", "", 'm',
'm', 'N', 'Y', 'Y', 'Y');
```

#### C. Insert the CENSUS information

You need to insert one census record for each census you plan to upload.

```
mysql> INSERT INTO Census (CensusID, PlotID, PlotCensusNumber, StartDate,
EndDate, Description)
VALUES (1, 1, 1, '2010-01-02', '2010-10-31', 'Any census description you want to
insert here');
```

#### D. Insert the MEASUREMENT CODES:

The statement below directly loads your codes file into the codes table (TSMAttributes).

```
mysql> LOAD DATA LOCAL INFILE '/path-to-filecodes.txt' INTO TABLE
TSMAttributes fields TERMINATED BY '\t' ENCLOSED BY " IGNORE 1 LINES
(TSMCode, description);
```

#### E. Insert the list of names into the PERSONNEL table:

There are three tables that hold Personnel data, The Personnel table stores names, the RoleReference table holds job descriptions or roles, and the PersonnelRole table links people with their roles.

First insert roles into the **RoleReference** table if empty:

If you have a role text file with your list of roles then you may use the LOAD DATA LOCAL INFILE command to insert your data.

```
LOAD DATA LOCAL INFILE '/path-to-file/rolel.txt' INTO TABLE RoleReference fields
TERMINATED BY '\t' ENCLOSED BY '"' IGNORE 1 LINES (Description);
```

If you do not have a role.txt you can create some records with the insert statement below.

NOTE: The three roles: field technician, field supervisor, and data entry technician are necessary, the others are optional.

```
mysql> INSERT INTO RoleReference (RoleID, Description) VALUES
(1, 'scholarship recipient'),
(2, 'data administrator'),
(3, 'data entry technician'),
(4, 'data manager'),
(5, 'field supervisor'),
(6, 'field technician'),
(7, 'plot supervisor'),
(8, 'predoc student'),
(9, 'principal investigator'),
(10, 'volunteer');
```

Then add a list of people involved in the census and link to their roles. Your personnel.txt file should have the three columns firstname, lastname, and role as specified in the Format Data Files section.

```
mysql> DROP TABLE IF EXISTS stagePersonnel;
mysql> CREATE TABLE stagePersonnel(
  PersonnelRoleID INT UNSIGNED AUTO_INCREMENT,
  FirstName CHAR(30),
  LastName CHAR(32),
  Role CHAR(40),
  RoleID INT,
  PersonnelID INT,
PRIMARY KEY (PersonnelRoleID)
);
```

```
mysql> LOAD DATA LOCAL INFILE '/path-to-file/personnel.txt' INTO TABLE
stagePersonnel fields TERMINATED BY '\t' ENCLOSED BY " IGNORE 1 LINES
(FirstName, LastName, Role);
mysql> UPDATE stagePersonnel A, RoleReference B SET A.RoleID=B.RoleID WHERE
A.Role=B.Description;
mysql> INSERT into Personnel (firstname, lastname) SELECT DISTINCT
firstname,lastname FROM stagePersonnel;
mysql> UPDATE stagePersonnel A, Personnel B SET A.PersonnelID=B.PersonnelID
WHERE A.FirstName=B.FirstName and A.LastName=B.LastName;
mysql> INSERT INTO PersonnelRole (PersonnelRoleID,PersonnelID,RoleID) SELECT
PersonnelRoleID,PersonnelID,RoleID FROM stagePersonnel;
```

#### F. Insert SPECIES data:

This is a key table in the system so take the time to review the file for spelling errors, etc. Make sure the IDLevels are filled in. There should be at least one species code for unidentified species. Check the instructions in the Format Data Files section.

```
mysql> LOAD DATA LOCAL INFILE '/path-to-file/species.txt' INTO TABLE
TempSpecies fields TERMINATED BY '\t' ENCLOSED BY " IGNORE 1 LINES
(Mnemonic, Genus, SpeciesName, FieldFamily, Authority, IDLevel);
```

#### G. Insert QUADRAT data:

There are three tables that hold location information: Quadrat, CensusQuadrat and Coordinates. The code below is written for census=1. If you are loading a recensus, change the censusid accordingly.

```
-- LOAD Quadrat AND CensusQuadrat INFORMATION --
```

```
mysql> LOAD DATA LOCAL INFILE 'path-to-file/quadrat.txt' INTO TABLE
TempQuadrat FIELDS TERMINATED BY '\t' IGNORE 1 LINES (QuadratName, StartX,
StartY, DimX, DimY);
```

```
mysql> UPDATE TempQuadrat SET PlotID=1,CensusID=1;
```

```
mysql> INSERT INTO Quadrat (PlotID, QuadratName, Area, IsStandardShape,
QuadratID) SELECT PlotID,QuadratName,DimX*DimY, 'Y', QuadratID FROM
TempQuadrat;
```

```
mysql> INSERT INTO CensusQuadrat (CensusID,QuadratID)
SELECT CensusID,QuadratID FROM TempQuadrat;
```

-- QX and QY are 0,0 if StartX and StartY refer to the lower lefthand corner of the  
quadrat --

```
mysql> INSERT INTO Coordinates (PlotID, QuadratID, PX, PY, QX, QY) SELECT
PlotID, QuadratID, StartX, StartY, 0, 0 FROM TempQuadrat;
```

## 2. Loading your site's census data

If you are using ctfswb data entry to enter data skip this section. Your data has to be uploaded into the corresponding Temp (temporary) tables in the database so that ctfswb can then upload into the production (permanent) tables.

Because your data is probably not in the correct format for uploading into the Temp tables, upload them into staging tables first. Create staging tables that exactly reflect the structure of your text files, their columns and the type of data in them.

Below is only a guide: change the structure of the staging tables to conform to the structure of your text files.

```
-- LOAD CENSUS DATA --
```

```
-- The last 4 variables are necessary, include them in your structure.
```

```
mysql> DROP TABLE IF EXISTS stageCensus;
mysql> CREATE TABLE stageCensus(
  Tag          CHAR(10),
  StemTag      CHAR(32),
  Mnemonic     CHAR(10),
  QuadratName  CHAR(12),
  x            FLOAT(8) DEFAULT NULL,
  y            FLOAT(8) DEFAULT NULL,
  DBH          FLOAT(8) DEFAULT NULL,
  Codes        VARCHAR(50) DEFAULT NULL,
  HOM          FLOAT(8) DEFAULT NULL,
  ExactDate    DATE,
  PlotID       INT UNSIGNED,
  PlotCensusNumber  INT UNSIGNED,
  CensusID     INT UNSIGNED,
  tempID       INT UNSIGNED AUTO_INCREMENT,
  PRIMARY KEY (tempID)
)ENGINE=MyISAM;
```

```
-- Change "path-to-file" to the path where your tab-delimited data files are --
```

```
mysql> LOAD DATA LOCAL INFILE '/path-to-file/census.txt' INTO TABLE stageCensus
fields TERMINATED BY '\t' ENCLOSED BY " IGNORE 1 LINES (Tag, StemTag,
Mnemonic, QuadratName, x, y, DBH, codes, HOM, Exactdate);
```

```
-- Replace all PlotID, PlotCensusNumber, and CensusID with the correct numbers, i.e.
if first census, then use 1 --
```

```
mysql> UPDATE stageCensus SET PlotID=1,PlotCensusNumber=1,CensusID=1;
```

```
-- Add an index to the table --
```

```
mysql> ALTER TABLE stageCensus ADD INDEX(Tag,PlotID);
```

```
-- If site uses stemtags, create the following index instead:
```

```
mysql> ALTER TABLE stageCensus ADD INDEX(Tag,StemTag,PlotID);
```

```
-- Now insert the relevant columns into TempNewPlants and TempLocations--
```

```
mysql> INSERT INTO TempNewPlants (QuadratName, Tag, StemTag, Mnemonic,
DBH, codes, HOM, ExactDate, x, y, PlotCensusNumber, PlotID, CensusID) SELECT
QuadratName, Tag, StemTag, Mnemonic, DBH, codes, HOM, ExactDate, x, y,
PlotCensusNumber, PlotID, CensusID FROM stageCensus;
```

```
mysql> INSERT INTO TempLocations (QuadratName, Tag, StemTag, X, Y,
PlotCensusNumber, PlotID, CensusID) SELECT DISTINCT QuadratName, Tag,
StemTag, x, y, PlotCensusNumber, PlotID, CensusID FROM stageCensus;
```



## CHAPTER 4. GUIDE TO CTFSEWEB

### *1. Creating a Configuration File with ctfswweb*

Go to the ctfswweb system and call up the database using the username and password that you set up in Step 7d of the *Installation Guide ctfswweb.pdf*.

Click on Data Entry Configuration in the top menu and select your plot. Select New Plants Form as the form type. Then set up a configuration file with columns that are in the same order as your field sheet. You can start from scratch and add new columns or alternatively, you can use the default configuration and change it to reflect your field sheet.

The names that you enter in the Column Name parameter have to be written exactly the same as the names in Column Name in the List of Column Names (click on this list in the left panel), including whether they are in capital letters or not.

The options for column types are:

- Simple textbox with text, integer or decimal values without limits  
This allows you to set up a column where you can enter any type of data but where no screening is done
- Simple textbox with numeric values with limits  
This allows you to set up a column to enter numeric values, specifying their minimum and maximum values, preventing the user from entering values outside this range.
- Species dropdown  
This creates a column where the user can only choose valid codes from the Species table
- Subquadrat dropdown  
This allows the user to include as one of the fields the subquadrat where the current tree is located. In most sites, this refers to the 5x5 m subquadrat within the 20x20 m quadrat.  
The "Starts with" option allows you to enter a "0" or "1", referring to which digit the site uses to start naming the subquadrats, i.e. 00, 01, 02, 03, .....33 or 11, 12, 13, 14, ...44.
- User dropdown

Allows you to specify specific options for the user to select. Separate the options with semicolons (;). Examples:

yes; no

L; B; M; D; B

- Simple textbox with a prefix value

This allows you to specify a field where the first few characters are set, while the rest changes from one record to the next. For example, some sites use tag numbers where the first 4 characters refer to the quadrat number, i.e. 1212-0001, 1212-0002, 1212-0003, and so on.

- Tree tag increment with stem tag

Several sites use stem tags that increase throughout the plot, instead of starting at 1 again for each new tree. This option allows the user to enter incremental numbers for the stem, whereas the tree gets the tag that is the smallest stem tag of that tree.

In all the options, except for the dropdowns, you have to configure the following 3 parameters for each column. These parameters permit the data entry person to enter data faster by not having to key in the tag numbers for each tree and stem, allowing the tag numbers to increment automatically from one record to the next.

a. Is Value Changed with Stem:

This lets you configure data entry such that the number entered into the current record is incremented by 1 in the next stem record (incremented) or optionally, the number in the current record is repeated in the next stem record (carried over).

b. Is Value Changed with Tree:

This lets you configure the data entry such that the number that is typed into the current record is incremented by 1 in the next tree record (incremented) or optionally, the number in the current record is repeated in the next tree record (carried over).

c. Add Next Row Stoke:

If you choose "Yes", this allows the cursor to go to the next record when you press the [ENTER] or [PLUS] in this field.

You should not choose YES in the first fields of the form, not until all the necessary fields have data entered. This prevents blank variables from being erroneously recorded. We suggest that you choose the YES option only after the DBH variable.

Following are some examples.

1. If you want your tree tag to increase by 1 from one tree to the next and your stem tags to start from 1 for each new tree, you would choose the following in the configuration file.

For the tree tag:

- a. Is Value Changed with Stem – Carried Over
- b. Is Value Changed with Tree – Incremented
- c. Null Value allowed - No
- d. Add Next Row Stoke – No

For the stem tag:

- a. Is Value Changed with Stem – Incremented
- b. Is Value Changed with Tree – No
- c. Null Value allowed - No
- d. Add Next Row Stoke – No

2. If the stem tags are not repeated throughout the plot, in other words, all stems are tagged and the tags are unique across the plot, such as:

<u>Tag</u>	<u>Stemtag</u>
123412	123412
123413	123413
123414	123414
123414	123415
123414	123416
123414	123417
123414	123418
123419	123419
123419	123420
123421	123421
123422	123422

Choose the following in your configuration file:

Add a new column for the tree tag, select “Tree Tag Increment with Stem Tag”, and choose the following options:

- a. Is Value Changed with Stem – Carried Over
- b. Null Value allowed - No
- c. Add Next Row Stoke – No

For the stem tag, select “Simple text box with text, integer or decimal values without limits” and the following options:

- a. Is Value Changed with Stem – Incremented
- b. Is Value Changed with Tree – Incremented
- c. Null Value allowed - No
- d. Add Next Row Stoke – No

Once you are finished configuring your data entry sheet, you can preview the form and try it out to make sure that you have set it up correctly, especially with respect to the tags and stem tag columns.

If you need to make changes, you can delete any of the variables, add a new column, then redo the configuration. This will insert your column in the bottom of the fields. To move this column to its correct position in the form, click on “Rearrange Columns” and drag the field to its correct place.

Once you are satisfied with the data entry form, save your changes by clicking on “Create Config File.”

**NOTE:** A common error is forgetting to create the configuration file and going directly into Data Entry.

## *II. Entering Data into the Data Forms with ctfswb*

If you already have data in the Temp tables, you can skip this section and the next (section III) and go directly to section IV.

Once you have created the configuration files, you are ready to use the data entry form to enter data from your census by going into Data Entry in the top menu and choosing the data entry form to enter data. Read the instructions on the screen. Some keys have been programmed to make data entry easier and faster.

- The asterisk [\*] allows the user to go from one field to the next.
- The slash [/] allows the user to go to a previous field.
- The plus [+] key from the numeric keypad adds a new stem record and carries over specified data into this new record in the fields where you specified “yes” in the “Add

Next Row Stoke” parameter.

- The [Enter] key adds a new tree record and carries over specified data in the fields where you specified “yes” in the “Add Next Row Stoke” parameter.

After you have finished entering a quadrat and pressed the submit button, a warning comes up asking whether you want to save the file. This prevents you from accidentally pressing submit and taking you out of the form before you have entered all the data. The warning also asks that you check that the quadrat is correct. A common error in data entry is forgetting to change the quadrat number when starting a new quadrat, resulting in the wrong quadrat being entered for all the records of that quadrat.

Once saved, we strongly suggest that you click on the name of the resulting text file and quickly check that all the records have been saved. Some computers have run into memory problems when there are many stems in one quadrat, and not all records are saved. For a solution if you run into this problem, see Appendix I.

The text files are placed in their respective folders under .../ctfswb/files/datafiles. Notice that the names include the name of the form, the name of the plot, the census number, the quadrat name, the date of data entry, and the name of the person entering the data.

Notice that there is an option for you to add more data into a quadrat that you had already started entering previously. This allows you to enter data from trees that were missed in a first pass through the quadrat by the field technician. It also allows you to divide a quadrat with a lot of trees into two or more entry sessions. Be careful to pick the correct file to add data to. The data from the later entry sessions will be inserted to the bottom of the file.

### *III. Double Data Entry with ctfswb*

We strongly suggest that census data be entered twice by two different data entry technicians if possible. Double data entry catches most of the errors caused by data entry. Once census data is entered twice, compare both files, look at the errors and pick the correct record to save.

Follow the steps below to compare two files in double data entry in ctfswb.

- Click on Data Screening in the left panel.
- Click on Double Data Entry
  
- Click on Copy Text Files into Temporary Tables
  
- Select the two text files that you want to compare. This inserts the two files into

MySQL temporary tables. Repeat this as many times as the number of quadrats that you have entered.

NOTE: Make sure you don't pick the same two files – this will lead to no errors found when comparing them.

- Start comparing the data in the temporary tables – choose the quadrat that you want to compare.

NOTE: Make sure you don't pick the same two files – this will lead to no errors found when comparing them.

- All records that are not exactly the same will show up in order of tag. Choose the correct record to keep by clicking on the circle to the left (the default is the first record). You can make changes to any of the fields of the record that you want to keep. You can also delete the record if you don't want to keep any of the two by pressing on “Delete Record” in the upper right hand corner. You can also skip this record and go back to it later if you are not sure which is correct and need to check it later.
- Be careful selecting the records to keep in the case where one of the files doesn't have a corresponding record in the other file. Because comparisons are made based on the tree tag, if the tree tag is entered incorrectly in one of the files, it will appear in one of the files in one instance while the other file has a missing record and vice versa in a later instance.

Double data entry screening should be done regularly throughout a census, not only at the end. One reason for this is that if there are any doubts or questions, the field technician can still remember what happened if the data was collected recently, or if necessary, go back to the field to check an entry.

As data entry progresses, there will be many temporary tables to scroll through when selecting the quadrats to compare. You can delete those quadrats that you have already compared and are empty by clicking on “Delete all Screened Temporary Tables with no data”.

NOTE: There may be times when you want to compare two text files again that you had already compared before. Once the text files are selected, they are moved from their folders to their respective upload folder in `.../ctfsweb/files/datafiles/uploaded/`. You can move them back to their original folder and they will again appear in the dropdown list for you to select. You may have to delete the records of this quadrat from the corresponding Temp table also if they had already been uploaded, i.e. TempNewPlants, TempOldTrees, or TempMultiStems. Be careful when deleting – you will not be able to recover the records once deleted.

#### *IV. Single Data Entry*

If census data was entered once only, then you can press the option “Single Data Entry”

under Data Screening to insert the text files into the corresponding Temp tables: TempOldTrees, TempNewPlants, and TempMultiStems.

## *II. Inserting Location Data*

If you use field maps to map the locations of the stems in the plot, you might want to scan them and digitize the stem locations. If you don't already have a digitizing software, we suggest that you use the open source image processing package ImageJ and the plugin pointpicker (<http://rsbweb.nih.gov/ij/>) to digitize the maps. The resulting coordinates are in pixels and you can convert them to meters by using the R function `fullplot.imageJ()` from the CTFSRPackage (<http://ctfs.si.edu/Public/CTFSRPackage> -> Topics -> topography imageJ.r).

To upload the resulting coordinates into the TempLocations table, use a LOAD command like the following.

```
mysql> LOAD DATA LOCAL INFILE '/path-to-file-with-coordinates/plotcoordinates.txt'  
INTO TABLE TempLocations FIELDS TERMINATED BY '\t' ENCLOSED BY '"' IGNORE  
1 LINES (Tag,x,y,QuadratName);
```

## *V. Backing Up Files*

Back up all your text files daily if possible. Back up everything from the `.../ctfswb/files/datafiles/` folder where the text files are. If you have already started double data screening, then you should back up the MySQL database also. To back up your database, go to the DOS command prompt and type:

```
CD\XAMPP\mysql\bin\  
mysqldump -u root --add-drop-table --extended-insert --result-file="/path-to-dump-  
file/yoursitename.sql" current_yoursitename
```

or alternatively, if you want to back up the Temp tables only:

```
mysqldump -u root --add-drop-table --extended-insert --result-file="/path-to-dump-  
file/yoursitename_Temptables.sql" current_yoursitename TempNewPlants  
TempOldTrees TempMultiStems
```

where

`yoursitename.sql/yoursitename_Temptables.sql` is the name that you want to call the "dumped" file

`current_yoursitename` is the name of the database you want to back up

If yoursitename.sql is a large file, you may want to compress it:

```
zip -r /path-to-zip-file/yoursitename_22August2013.zip /path-to-dump-  
file/yoursitename.sql
```

We strongly suggest that you include the date in the name of the compressed file. Do not overwrite backups, instead save them all. This way you will always be able to go back to a previous version if necessary.

## *VI. Screening Data in Temporary Tables*

Once the data is inserted into the Temp tables, you can screen them and look at the “errors”. Screening may take a little while, so please be patient. Once screening is finished, you can look at a summary of the errors by clicking on View Error Summary. Some of the errors can be ignored (i.e. main stem errors), while others should be fixed (i.e. duplicate tag/stemtags, codes not found, >1 Quadrat or >1 Species, etc.).

The errors should be fixed using MySQL commands and saved in a text file so that a record is kept of all the changes that were made to the Temp tables. If you save all your sql commands in a script, you can run them all again if necessary by sourcing this file. To write these commands in MySQL, some basic knowledge of SQL is needed.

Rescreen the Temp tables as many times as necessary until the errors that come out are errors that you can ignore. For those errors that cannot be fixed in the office, save them in a file to give to the field workers to check in the field.

## *VII. Uploading Data into Database and Creating ViewFullTable*

Once screening displays only errors that are basically warnings and not fatal errors (fatal errors include duplicate tags/stemtags, codes not listed in the TSMAttributes table, species not found in the Species table), then you can start uploading the data from the Temp tables to the permanent (production) tables. Load them in order, i.e. the data from the Old Trees form before the new recruits, for the case where this is a recensus (i.e. not the first census). Uploading may take a while especially in the case of plots with many trees.

After the data has been uploaded, create the report tables. This step creates view tables (ViewFullTable and ViewTaxonomy) which takes all the records from the relevant tables and merges them again into flat files. This script also creates a variable called status in ViewFullTable that defines whether the tree is dead, alive, or missing, and whether the stem is lost.



ViewFullTable makes querying the database easier and faster since the user does not have to join records from the various tables and instead can just query this table. ViewFullTable is what the Plot Data Report queries to get requests out. This is also the table that is used to make the R analytical tables.

NOTE: Every time a change is made to the database, it is necessary to update the ViewFullTable and ViewTaxonomy by running this step.

We suggest that you run a check on the data that is uploaded into the permanent tables and review the results to make sure there are no glaring problems. This is done by running Post Screening.

Some of the checks that are done in Post Screening are:

- Counting the number of records by quadrat
- Counting the number of dead trees and missing stems
- Checking whether there are any trees located outside the limits of the plot or any trees with missing coordinates
- Checking whether there are any duplicate tags
- Displaying the range of the dates of the censuses
- Displaying the largest dbh and height of measure from each species
- Checking for any trees that are both alive and dead in one census
- Checking whether all trees from one census are accounted for in the next census
- Checking whether any trees that were dead in one census were alive in the next census
- Checking whether there are any trees with extreme growth rates.

Once the database has been reviewed and considered to be “stable” enough (until the next set of edits), save the database by using mysqldump (see Section IX above). The name of the resulting file should include the date the dump was made.

You can now erase all the Temp and staging tables so that your database contains only the production tables.

Once you are finished, log out of ctfswb.

## **CHAPTER 5. Guide to Correcting Data Errors**

### *I. Field Staff*

#### 1. What is a data error?

Field Errors – A mistake made in the field. Some example of these errors may be duplication of existing tag numbers, incorrect measurements of dbh, incorrect identification of species, incorrect assumption of death in an individual. Some errors may be systematic. Eg. a quadrat number may be off by one for an entire line.

Data Transcription errors- These errors are made after field data is brought in. They may be typing errors, format errors such as the incorrect use of delimiters for codes, omission errors where a data sheet or an entire column has not been entered. Errors may also occur in the digitizing system when entering new plants. Incorrect reading of tag numbers when digitizing locations, incorrect calibration when using ImageJ to convert pixels to plot coordinates, etc.

#### 2. When can you correct it?

While errors are best corrected as quickly as possible, it is more efficient to gather errors into coherent batches for submission. Typically a lot of error correction will occur immediately after a census. When all the known errors have been fixed, the database is uploaded and a version of the database is frozen. Subsequent errors should be collected into batches and submitted every three months or so. Each batch of corrections will result in a minor freeze which is a new version of the database. At the beginning of each calendar year a special effort must be made to clean up all known database errors so that they can be corrected and uploaded by March of each year for inclusion into a major freeze which is a version of the database that can be used by researchers.

#### 3. Who is authorized to correct a data error for your site?

Each site has a data manager who is the person authorized to apply corrections to the official database for your site. The data manager may be local to your site or you may request assistance from a CTFS systems group data manager (i.e. Suzanne Lao and Shameema Esufali).

#### 4. How do you submit your corrections?

Corrections should be submitted in the form of spreadsheets (excel) or comma delimited textfiles. Each column in the spreadsheet should contain only one data item. At the minimum, the spreadsheet should have the following columns:

Key - necessary column or columns needed to completely identify the item to be changed. Eg. tag and stem number to identify a stem to be changed, species code or complete name for a species change etc.

Old value – value to be changed, eg. old dbh, old tag

New value – Corrected dbh, corrected tag

If there are different kinds of changes, for instance changes to individual trees and changes to species records, then report them in separate spreadsheets or csv files.

## *II. Data Managers*

### 1. How to correct data errors

Data errors are corrected by using a script to apply corrections to a known version of the database, typically the export of the current database.

This is best done in the following way.

1. Login to MySQL console window and create a blank test database in MySQL on your local computer.

```
mysql> DROP DATABASE IF EXISTS test_sitename;
```

```
mysql> CREATE DATABASE test_sitename;
```

```
mysql> USE test_sitename;
```

2. Install a known version (usually the last export) of your dataset. This is your starting point or base dataset.

```
mysql> source /path-to-file/sitename.sql;
```

Every site should have an export file of their database in its current state. These can be downloaded from the CTFS website for the site using the login given to the PI for each site.

<http://ctfs.si.edu/sitename/current/archive/>

Download the most current database by looking at the date in the file names.

3. Source your correction script against the imported database

```
mysql> source /path-to-file/correction_script.sql;
```

Check your results, debug the script if needed and repeat from step 1.

When you are satisfied that the script is working correctly, proceed to the next step.

4. Rebuild ViewFullTable and ViewTaxonomy.

Use the Create Reports Tables in the Finishing Up option in the ctfsweb application to recreate ViewFullTable and ViewTaxonomy.

5. Export the corrected database. This is your new current version.

6. Send the following files to your CTFS Systems Data Manager (Suzanne or Shameema):

- Correction\_script.sql

- csv/excel files used by the correction script

- A README.txt file listing the base version of the database that you used at the beginning of this process.

- A check sum file. A standard checksum script is available at:

[http://ctfs.si.edu/Public/DatabaseSetup/ctfsweb/database\\_instructions/sample\\_scripts/Checksum.sql](http://ctfs.si.edu/Public/DatabaseSetup/ctfsweb/database_instructions/sample_scripts/Checksum.sql)

Run this and include the results file in your package.

### *III. Script writing*

#### 1. What is a script

A script is a program that can be run against the database to install corrections. Typically it reads one or more data correction files, the comma delimited files described above, and updates the erroneous value to a new value. Sometimes it needs to delete records, for example if there are duplicates, and sometimes missed records need to be inserted. Scripts are written in MySQL or php. A data manager should have a reasonable knowledge of MySQL. Scripts may be submitted to Suzanne or Shameema for review and either of us is always available for advice and examples of code.

#### 2. Why do we write scripts to correct errors

This is an important standard for making corrections for several reasons.

If you start with a known database which has an exported dump file and run a script against it to make corrections and if you find that you have made a mistake, then you simply correct the script, reimport your clean database and rerun the corrected script.

The same process can be duplicated on any server that holds your database to produce an exact duplicate of your current database.

A permanent record of your changes is available for inspection in case of later problems.

### 3. Text editors and language standards for CTFS data manipulation

CTFS recommends the use of Geany, Notepad++ or gedit for script writing. These text editors are much more powerful than the standard Wordpad and Notepad: they can open large files significantly faster, they support smart syntax and code highlighting for many programming languages, they have line numbering, etc. Scripts must be written in MySQL. Advanced users may use php.

### 4. Base data

Corrections are usually applied to the last exported dataset. These datasets are available to authorized users at <http://ctfs.si.edu/sitename/current/>. The PI for each site has the user name and password for the site and you must be authorized by the PI to retrieve and use site data.

### 5. Multiple scripts

It may be necessary to use more than one script. If scripts are to be run sequentially, then the README file should list the commands needed to run the scripts in the correct order and provide any other information required to complete the upgrade.

## *IV. Creating a package*

### 6. What should be in a package

The following files should be in an upgrade package. The package should be compressed using winzip or tar before emailing.

- Script or scripts (.sql or .php files)
- Data (excel or csv files)
- README.txt containing:

1. Name of programmer, site, date and a brief description of the corrections being made.

2. The version number of the base dataset prior to the application of the script

3. Source command or commands to execute script or scripts.

4. Checksum.sql (Checksum.sql is available at

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/sample\\_scripts/Cchecksum.sql](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/sample_scripts/Cchecksum.sql))

5. Run the Checksum.sql on your database before you make any changes and save the results to a text file called ResultofChecksumbeforechanges.txt.  
Repeat the checksum after you run the changes and copy the results to a text file called ResultofChecksumafterchanges.txt.  
Send these as the final component of your package.

### *V. How to test a package before submission*

After you test the upgrade for accuracy on your own computer, one more step is required before dispatching the package. Programs that refer to files typically have a directory path that is only valid on your own computer. These paths can be standardized so that they work on any computer. The package root directory should have the following structure.

```
yoursiteupgrade2013OCT (root folder for the upgrade -change to reflect your site and upgrade)
  scripts - directory
  data - directory
  README.txt -file
  Checksum.sql -file
  ResultofChecksumbeforechanges.txt -file
  ResultofChecksumafterchanges.txt -file
```

Now go into the scripts directory and change the path of any referenced file so that it is relative to the root directory for the package.

Eg. SOURCE scripts/correct\_errors.sql  
LOAD DATA LOCAL INFILE data/tag\_corrections.csv ...

Now start your mysql console from the root and run your upgrade according to the instructions in your readme file. If this runs without error your package is ready to compress and send.

## *VI. Sample scripts – This chapter is in development*

### 1. Insert new records

A sample script is available at

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/sample\\_scripts/sampleinserttree.sql](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/sample_scripts/sampleinserttree.sql)

### 2. Update existing records

### 3. Delete existing records

A sample script is available at

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/sample\\_scripts/sampledeletetree.sql](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/sample_scripts/sampledeletetree.sql)

You will also need to download the utilities required to run this script

[http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database\\_instructions/sample\\_scripts/utilities.sql](http://ctfs.si.edu/Public/DatabaseSetup/ctfswweb/database_instructions/sample_scripts/utilities.sql)

## CHAPTER 6. Digitizing and Uploading New Plant Coordinates

### *I. Installing ImageJ and Point Picker*

If your site uses paper maps, we suggest using the image processing and analysis software called *ImageJ* to digitize the points. This software is open source and available for Windows, Linux, and Mac OS platforms. You can download the program, as well as the instructions, from:

<http://rsb.info.nih.gov/ij>

You also need to install the plugin Point Picker. We have modified the original Point Picker program to allow labeling of the points. Install the modified Point Picker by extracting the files from the zipped file “**pointpicker\_modified3.zip**” into /Program Files/ImageJ/plugins (or wherever ImageJ was installed). The description and user manual for Point Picker can be downloaded from:  
<http://bigwww.epfl.ch/thevenaz/pointpicker/>.

### *II. Using ImageJ to Digitize Plot Maps*

- 1) Scan all your paper maps and save them as JPEGs on your computer. Use the quadrat name (and subquadrat, if applicable) of the map as part of the file name. Be consistent when naming the map images. Examples of possible naming conventions for JPEG map files are:

e.g. “**Q0312.jpg**” or “**Map\_0312\_2.jpg**”

“0312” denotes the quadrat number and in the second case, “2” specifies the subquadrat.


- 2) Open the ImageJ application and then open the specific JPEG map file that will be digitized and ultimately recorded as a text file.

NOTE: It is strongly recommended that the original paper copy of the hand-drawn map be examined alongside the corresponding datasheet to make sure all stems are actually mapped for the specific quadrat/subquadrat, and that all tag numbers correspond.


- 3) Once the JPEG image is open in ImageJ, select the “pointpicker” menu option as follows:

**Plugins** → **pointpicker** → **PointPicker**



- 4) Select the “Add crosses” button  (the very first option with the plus sign) to add points to the four corners of the map. The four corners of the subquadrat map should be digitized first for calibration, in the following order, and labeled as follows:


**p1 = lower left**  
**p2 = upper left**  
**p3 = upper right**  
**p4 = lower right**


- 5) Now select the “Add crosses” button  to digitize the trees by putting the cursor over the center of all the points. Label the points with the complete tag number.


One of the most common errors when digitizing maps is to leave out points, another is to label the points incorrectly. There are several ways of preventing this.


One suggestion is to digitize the points in sequential tag order, with out-of-sequence tags done at the very end.


Another approach would be to finish one subquadrat at a time, making sure to digitize all the stems from that subquadrat.



- 6) Check the resulting list after all crosses have been added to the map image. The Results list is accessed by clicking the “Export/Import list of points” button  and then clicking on the “Show” option.

- 7) The “Magnifying glass” feature  may be used to zoom in to areas of the map where stem dots or tag numbers appear small at the original magnification, or where many dots are clustered close together. Use the left-click button on the mouse to zoom in and the right-click button to zoom out.

Tag numbers may be edited using the “Edit labels” button  (the second icon with a plus sign).

Crosses may be deleted using the “Remove crosses” button .


If any crosses added appear off center, the crosses may be adjusted using the “Move crosses” button .

- 8) In sites where the multiple stems from one tree may have different locations, each multiple stem should be digitized. If there is a case where only one dot is drawn in for a group of multiple stems, it is suggested that crosses be added to the main stem tag (the drawn dot) *and* the non-represented stems around the dot to account for the multiple stem tags.
- 9) Once all crosses have been added, check over the map image to ensure that every drawn-in tagged stem has an added cross to it. Next, the Results list should be reviewed to ensure that all four corners and all the tagged stems are displayed in the list and labeled correctly.
- 10) When everything is verified as being correct for the now digitized quadrat or subquadrat map, save the list of points and corresponding x/y coordinates to a text file. The text file may be saved by clicking the “Export/Import list of points” button  and then clicking the “Save as” option.
- 11) Be consistent when naming the text files. The text files should be saved following the naming convention for the scanned maps.  
e.g. **“Q0312.txt”** or **“Map\_0312\_2.txt”**  
We suggest that the resulting text files be saved in directories denoting the column:  
e.g. **“C:\Maps\Column03”**
- 12) Following completion of the digitized quadrat/subquadrat map, the image must be closed before opening the next quadrat or subquadrat map image. This can be done in one of two ways:
  - a) Click on the “Exit PointPicker” button , then click the “Done” option to exit the “PointPicker” feature which will remove all added crosses from the map image and return to the original ImageJ configuration. Then close the map image. Now the next

map image may be opened.

Or

b) Simply close the ImageJ application altogether and then reopen the application to completely reset it for the next map image.

**13)** If a completed digitized map needs to be reopened to review the placement of crosses on the map image or the Results list, simply open the specific quadrat/subquadrat map image in ImageJ, select the “PointPicker” menu option, click on the “Export/Import list of points” button , and then click the “Open” option to open the text file that corresponds with the matching quadrat/subquadrat map image. All added crosses should appear in correct placement over the map image and the Results list can then be accessed following the same procedure outlined in step 7 above.

### ***III. Converting ImageJ Text Files from Pixels to Meters***

To convert imageJ text files from pixels to x,y coordinates in meters do the following in R:

1) Open R

2) Install or load the the CTFSRPackage: `attach('/path/CTFSRPackage.rdata')`

3) Use the `fullplot.imageJ` function. A tutorial is available in:  
<http://ctfs.si.edu/Public/CTFSRPackage/index.php/web/tutorials>  
Select ImageJ.

Choose the correct parameters for `fullplot.imageJ` according to the path where the text files are located, the size of your quadrats or subquadrats, and how you named your map files.

For an explanation of the parameters, go to:

<http://ctfs.si.edu/Public/CTFSRPackage/index.php/web/topics/topography~slash~imageJ.r/fullplot.imageJ>

e.g. `fullplot.imageJ(path='/maps/Column02', gridsize=c(10,10),`

```
outfile='ConvertedCoord1.txt', prefix='Map_', colrange=c(0,49),  
rowrange=c(0,24), corners=c('p1','p2','p3','p4'),  
subquadsuffix=c('_1','_2','_3','_4'))
```

- a) This will put all the converted coordinates with their tag numbers and corresponding quadrat names (from the name of the text files) into a file called “ConvertedCoord1.txt”
- b) This file will be saved in the path you specify: /maps/Column02/
- c) All files in this path with “.txt” extension, “Map\_” prefix and subquadrat suffix specified above will be converted (example: Map\_0201\_1.txt, Map\_0222\_2.txt, etc.)
- d) The subquadrat suffix should be named in a clockwise direction starting from the lower left corner
- e) The calibrated corners should also be named in a clockwise direction from the lower left corner

In the case that there are no subquadrats, only 20x20m quadrats:

```
e.g. fullplot.imageJ(path='/maps/Column02', gridsize=c(20,20), outfile='Converted.txt',  
prefix='Q', colrange=c(0,49), rowrange=c(0,24), corners=c('p1','p2','p3','p4'))
```

All files in this path with “.txt” extension and “Q” prefix will be converted (example: Q0301.txt, Q0324.txt, etc.) and saved into a text file called “Converted.txt”.

## Appendix to Chapter 2. Table Descriptions

The tables below need to be submitted to CTFS for inclusion into the program. The key transaction table is the last one listed 'ViewFullTable'. The balance tables are key reference tables that contain static data about the plot. Leave blank any foreign key id's or any other inapplicable data.

Table Name	Column Name	Description	Is Nullable	Column Type	Key
<b>Country</b>					
CountryID		Primary key, an integer automatically generated to uniquely identify a country.	NO	int(10) unsigned	PRI
CountryName		Name of Country	YES	varchar(64)	
<b>Quadrat</b>					
PlotID		Foreign Key to Site table.	NO	int(10) unsigned	MUL
QuadratName		The character name for the quadrat, usually the name used in the field; may be the row and column. eg. 0322.	YES	char(8)	MUL
Area		Area of quadrat in square meters.	YES	float unsigned	
IsStandardShape		Y if quadrat is a square, otherwise N.	NO	enum('Y','N')	
QuadratID		Primary key, an integer automatically generated to uniquely identify a quadrat.	NO	int(10) unsigned	PRI
<b>TSMAttributes</b>					
TSMID		Primary key, an integer automatically generated to uniquely identify a Tree, Stem or Measurement Code.	NO	int(10) unsigned	PRI
TSMCode		Code describing or explaining tree, stem or measurements. Eg Dead, Lost, Leaning etc.	NO	char(10)	
Description		Free text description of the code above..	NO	varchar(128)	

Table Name	Column Name	Description	Is Nullable	Column Type	Key
<b>Site</b>					
	PlotID	Primary key, an integer automatically generated to uniquely identify a plot site.	NO	int(10) unsigned	PRI
	PlotName	Name of the plot eg. BCI, Sinharaja.	YES	char(64)	
	LocationName	Geographical location. Eg 'Barro Colorado Island', 'Central Province'.	YES	varchar(128)	
	CountryID	Foreign Key to Country table.	NO	smallint(5) unsigned	MUL
	ShapeOfSite	A character description of the plot's shape (user-defined; might be dimensions, e.g. 1000x500m or 500x500m).	YES	char(32)	
	DescriptionOfSite	A free text description of the site.	YES	varchar(128)	
	Area	Area of the plot in square meters.	NO	float unsigned	
	QDimX	Length of quadrat in meters along the X axis.	NO	float unsigned	
	QDimY	Length of quadrat in meters along the Y axis.	NO	float unsigned	
	GUOM	Unit of Measure for global coordinates.	NO	varchar(32)	
	GZUOM	Unit of Measure for global elevation coordinates.	NO	varchar(32)	
	PUOM	Unit of Measure for plot coordinates.	NO	varchar(32)	
	QUOM	Unit of Measure for quadrat coordinates.	NO	varchar(32)	
	GCoorCollected	Were global coordinates collected: Y or N?	YES	varchar(32)	
	PCoorCollected	Were plot coordinates collected: Y or N?	YES	varchar(32)	
	QCoorCollected	Were quadrat coordinates collected: Y or N?	YES	varchar(32)	
	IsStandardSize	Y if plot is rectangular, N if circular or irregularly shaped.	YES	enum('Y','N')	

## View Taxonomy

SpeciesID	Foreign Key to Species table	NO	int(11)	PRI
SubspeciesID	Foreign Key to SubSpecies table, indicating the subspecies if there is one	NO	int(11)	PRI
Family	Taxonomic family name (from the Angiosperm Phylogeny Group - APG - system).	YES	char(32)	
Genus	Genus the species belongs to, according to the APG system.	YES	char(32)	
Mnemonic	Code used in the field for designating the species, usually 6 letters (4 for the genus and 2 for the species).	YES	char(10)	
SpeciesName	Species part of Latin name; (or may be a morphospecies name).	YES	char(64)	
SubspeciesName	Subspecies portion of the Latin name, may be a subspecies or variety. The deepest taxonomic level for which full identification is known. Limited to values species, genus, family, none, or multiple. None is used when family is not known. Multiple is used when the name may include a mixture of more than one species.	YES	char(64)	
IDLevel		YES	char(8)	
Authority	Taxonomic authority for the classification of the species.	YES	char(124)	
ListOfOldNames	List of old names or synonyms used previously, separated by commas	YES	varchar(255)	
NumberOfHerbarium	Number of herbaria where specimens of this species is found	YES	int(11)	
ListOfHerbarium	List of the names of herbaria where specimens of this species are found	YES	varchar(255)	
Description	A free text description of the species, as relevant for the plot (especially, who identified and how).	YES	varchar(128)	

Table Name	Column Name	Description	Is Nullable	Column Type	Key
------------	-------------	-------------	----------------	-------------	-----

## ViewFullTable

DBHID	Foreign Key to DBH table. This may be left blank	NO	int(11)	PRI
PlotID	Foreign Key to Site table.	NO	int(11)	MUL
Plot	Descriptive name of the site also referred to as the plot.	YES	varchar(35)	
Family	Taxonomic family name (from the Angiosperm Phylogeny Group - APG - system).	YES	char(32)	
GenusSpecies	Scientific Latin name that includes the genus and species.	YES	char(64)	
Genus	Genus of the plant, according to the APG system.	YES	char(32)	MUL
SpeciesName	Species part of Latin name, may be a morphospecies name.	YES	char(64)	MUL
SubSpeciesName	Subspecies portion of the Latin name, may be a subspecies or variety.	YES	char(64)	
SpeciesID	Foreign Key to Species table.	YES	int(10) unsigned	MUL
Mnemonic	Code used in the field for designating the species, usually 6 letters (4 for the genus and 2 for the species).	YES	char(10)	
QuadratID	Foreign Key to Quadrat table.	NO	int(11)	
QuadratName	Descriptive name for the quadrat used in the field. The first two characters (digits) usually refer to the column and the last two to the row.	YES	varchar(12)	MUL
QX	Distance from the lower left corner of the quadrat in meters on the X axis.	YES	float	
QY	Distance from the lower left corner of the quadrat in meters on the Y axis.	YES	float	
PX	Distance from the plot origin (lower left corner) in meters on the X axis.	YES	float	
PY	Distance from the plot origin (lower left corner) in meters on the Y axis.	YES	float	
TreeID	Foreign Key to Tree table.	NO	int(11)	MUL
Tag	Tag number on the tree in the field, should be unique within each plot.	YES	char(10)	MUL
StemID	Foreign Key to Stem table.	NO	int(11)	
StemNumber	Column used to carry stemid from previous database version.	NO	int(11)	
StemTag	The stem tag used in the field to identify the different stems of a tree in the case of multiple-stemmed trees.	YES	varchar(32)	
PrimaryStem	A character description of the stem, whether it is the primary or a secondary stem, or a branch, etc.	YES	char(20)	MUL
CensusID	Foreign Key to Census table.	NO	int(11)	PRI
PlotCensusNumber	Census number, an integer, 1=first census, etc.	YES	int(11)	MUL
DBH	Stem diameter, usually at breast height. Units are user defined, but assumed to be consistent within the database. It is recommended that the dbh be rounded down to the nearest 5 mm for trees <=5 cm dbh.	YES	float	MUL
HOM	Height (in meters) on the stem at which the diameter was measured, usually at 1.3 meters.	YES	float	
ExactDate	Date on which the measurement was taken (format is yyyy-mm-dd).	YES	date	
ListOfTSM	Codes indicating the attributes or condition of the tree, stem, or measurement. Codes are separated by a comma in the case of more than one. An explanation of the codes is found in the TSMAttributes.txt file.	YES	varchar(256)	MUL
Status	lost_stem (stem is dead, not found, or broken, etc. but other stems of the tree are still alive), or missing (tree or stem was not found, so measurement is unknown).	YES	varchar(15)	MUL



### ***Appendix to Chapter 4. Troubleshooting long fieldsheets not saved.***

Some computers have run into memory problems when entering data in ctfswb, where long field sheets from a quadrat are not saved. When this happens, you may want to change the configuration php file that comes with XAMPP.

Do the following carefully.

1. Go to the `/xampp/php/` folder and make a copy of the original `php.ini` file (i.e. copy it to another file called `php_orig.ini`).
2. Open `php.ini` in a text editor, look for the following parameters, and make the corresponding changes. Leave a space in front of and after the equal sign.

```
max_execution_time = 2400
```

```
max_input_time = 600
```

```
memory_limit = 768M
```

```
post_max_size = 48M
```

3. Add the following line, maybe after line 460:

```
max_input_vars = 20000
```

4. Save `php.ini`.
5. Stop Xampp and restart Apache and MySQL.

If you are still having problems, change the parameters again by increasing the numbers.