RCLogbook

Version 5.1

Import & Export Guide

clevertangerine software

Table of Contents

About this Guide

This guide describes the format of some files that RCLogbook uses to move data between its database and the user. RCLogbook can export and import a subset of its database using this format.

- Using this format for *export*, RCLogbook provides the user with a human-readable version of a subset of the database contents.
- Using this format for *import*, the user can provide RCLogbook with descriptions of edits to make to the database including both updating fields as well as adding new records.

For detailed discussion on how you can use the export or import functionality in RCLogbook, please consult the *RCLogbook User's Guide*.

☝ *The user's guide is linked from the RCLogbook "About RCLogbook" screen accessible from the Database tab, or from the web at http://www.clevertangerine.com/RCLogbook*

The RCLogbook Import/Export format, RIX, is text-based, human-readable, and allows you update some, but not all, of the content in the RCLogbook database from a text file that you can edit on your Mac or PC.

Import Functionality Assumptions

It is important to understand the assumptions that RCLogbook makes around *how* you use the import function with an RIX file. The import functionality **was intended** to support:

- Making batch additions to your database. For example, import allows you to incorporate logs from other sources (such as a different app or a paper logbook) into RCLogbook without having to go through the user interface.
- Making many simple edits to a subset of the database.

The import function **was not intended** to support:

- Making backups of the database. Because not all records or database fields can be accessed through this file (for example, checklist records or the image of a model), exporting, then importing, will not generally recreate the exported database.
- Removing records or information from the database.

The import functionality is **not** intended to be a substitute for basic editing that can be done through the user interface.

☝ *With Great Power comes Great Responsibility. You can make a mess of your database if you are not careful. RCLogbook does its best to keep you from doing so, but please exercise extreme caution when importing.*

*It is **strongly** advised that before importing, you make a full backup of your database so that you can get back to a known good state if something gets messed up.*

RCLogbook tries to be very picky about the formatting and contents of the text file for import in order to try to protect you from some common mistakes.

Conventions

As seen above, this guide uses the following notation to indicate a point of interest that explains a point further or brings attention to differences with a prior release.

RCLogbook 5.1 can now spin the black circle.

This guide typesets RIX text file contents like this:

```
1  [LOCATION]
2  Location UID➔Location Name➔Latitude➔Longitude➔Location Notes
3  354➔Apple Computer➔37.331607➔-122.030382➔{Notes Go Here}
```

The symbol "➔" represents a tab character. The numbers in italics at the left of each line help identify the line number within the file; these numbers help orient the discussion and **would not** appear in the file. Within the body of the file, we use italic text in curly brackets to identify placeholders. In this example, the text "*{Notes Go Here}*" on line 3 does not literally appear in the file, but instead it calls out that the notes would be located at this point in the file.

Contacting Us

We welcome your feedback and suggestions. Many of the new features that have been added since 1.0 have come from suggestions by users. You can reach us on the web at http://www.clevertangerine.com or over email at support@clevertangerine.com. Links can be found on the "About RCLogbook" page found in the Database tab of the application.

File Format & Structure

The RIX text files for import and export both use an identical "tab-separated" format where tab characters separate field values on a line of text. Import and export differ primarily in how the fields are filled in. A RIX text file consists of a number of *lines* each terminated by one or more newline characters[1]. Each *line* is further divided into one or more tab-delimited *values*. By definition, *values* cannot contain tab characters. As discussed later, groups of consecutive, non-blank, lines in the file correspond to record(s) from the database. RCLogbook ignores blank and comment lines (any line beginning with the characters "//").

Here is an example of this basic structure to illustrate these concepts,

```
1  iPod➔➔iPhone
2
3  // This is a comment
4  iPad➔iMac
```

Line 1 has three values: "iPod", "" (that is, a blank value; note that back-to-back tabs imply a blank value), and "iPhone". Line 2 is blank and has no values. Line 3 is a comment and

[1] Specifically, the characters are a CR and LF pair, though RCLogbook is flexible about the specifics.

similarly has no values. Line 4 has two values: "`iPad`" and "`iMac`". RCLogbook would ignore lines 2 and 3 were it to process this file.

A RIX text file for import or export begins with a *header line* that establishes the file format and version of RCLogbook that can process the file. The remainder of the file is made up of groups of lines, called *sections*, where the *values* describe *fields* from database records.

File Header Line

This header is always the first non-blank line in the RIX text file. The header consists of one line that marks the start of the file and identifies the version of RCLogbook that generated or can process the file. Specifically,

```
RCLogbook 5.1.0 Db v.8 (CD) / Text Format
```

To import, RCLogbook requires that the version in the header match both the application and database versions. RCLogbook 5.1.0, for example, can only process a file marked with the correct application version (5.1.0) and the correct database version (v.8). A given release of RCLogbook will not support the RIX files formatted for any prior release.

By generating an export or template file from the application (see the *RCLogbook User's Guide* for more information), you can determine the proper header to use for an import file. And, generally, it's valuable to work from the template file when creating an import file.

Sections

Sections in RIX carry groups of database records of the same type. Sections always contain at least two lines. The first line is a *section marker* that marks the start of the section and identifies the type of information that follows. The second line is the *field definition header* that identifies the fields that are present on the remaining lines of the section. Finally, any lines following the *field definition header* of the section describe the content of records from the database. The section continues until the next *section marker* or the end of the file. For example,

```
1  [CARS]
2  Wheels→Name
3  2→Katana
4  4→Tesla
5  [HIGHWAYS]
6  Name→Length→Start→End
7  I-5→1381→California→Washington
```

In this example, there are two sections: `CARS` and `HIGHWAYS`. Note that section names are always enclosed in square brackets. The `CARS` section includes lines 1 through 4 and the `HIGHWAYS` section includes lines 5 through 7. Lines 2 and 6 provide the *field definition headers* for each of the two sections. In the `CARS` section, for example, each line has a "`Wheels`" field and a "`Name`" field.

Mapping Lines in the RIX File to Database Records

Within a section in a RIX file, one or more lines correspond to a set of one or more related database records. The value(s) on those lines correspond to values within database records. Note that the RIX format may only allow some of the fields in a database record to be edited via RIX.

Each line in a section in RIX always typically starts with an "Action" field immediately followed by an "ID" field. These fields define the action to take, the database record to operate on, and how many lines correspond to the record. For the action,

- "`ADD`" – Adds a new record to the database. With this action, the subsequent ID field is always empty.
- "`UPD`" – Updates an existing database record. With this action, the subsequent ID field identifies the record to edit.

The ID field is used internally by RCLogbook and ***<u>should not</u>*** *be edited by the user. Editing these identifiers at best generates an error during processing and at worst can corrupt your database.*

When exporting, RCLogbook will always leave the action field empty and populate the ID field with the proper database identifier. Also, as described later, it is also possible to have additional action and ID fields appear later on the line when there is a relationship between different types of database records.

When the fields from a database record require more than one line to express in the RIX file (for example, a multi-line text field from a record), the database record will correspond to N contiguous lines from the RIX file where all lines other than the first have blank action and ID fields. The lines with blank action and ID fields are called "continuation lines". Consider the following example RIX file where we assume that the "`Notes`" field from the "`CARS`" section can be spread across multiple lines.

```
1  [CARS]
2  Action      →ID        →Name      →Notes
3  ADD         →          →McLaren   →New car
4  ADD         →          →NS-X      →Line #1
5              →          →          →Line #2
6  UPD         →ID.66710  →          →F355
```

This file, then, creates two new `CARS` records in the database and edits a third record. The first new record comes from line 3, the second new record comes from lines 4 and 5, and the edits to the existing record come from line 6. Line 5 in this example would be a continuation line.

The notes for the second record in lines 4-5 would be formatted as follows:

```
Line #1
Line #2
```

The specific interpretation of multiple lines depends on the section and field and is covered further below when we discuss the RIX sections RCLogbook supports.

Data Types

Each *field* in a *database record* carries data of a specific type[2]. As a result, when the RIX file specifies changes to a *field* (through a *value*), the *value* from the RIX file must provide the appropriate type. The specific mapping between *fields*, *database records*, and *values*, is covered later when we discuss the specific sections that RIX supports.

The data types that can appear in the import file are as follows,

Data Type	Description
RIX Action	String that is blank, "ADD", "UPD" and indicates the action to take as well as whether or not the line is a continuation.
RIX ID	Database record unique identifier. These are text strings that uniquely identify an existing database record and should not be modified by the user. The ID is always blank for new records (i.e., when the action is "ADD").
Integer	Any integer number. For example, 97 or -13.
Integer (P)	A positive or zero integer number. For example 97 or 0.
Integer (P, NZ)	A positive and non-zero zero integer number. For example 97.
Real	Any real number. For example, 6.16 or -9.25.
Real (P)	A positive or zero real number. For example, 6.16 or 0.0.
Real List (P)	A ";"-separated list of positive or zero real numbers. For example, "6.16 ; 0.0 ; 1.23".
Boolean	Either "Yes" or "No".
String	A text string. Unless otherwise indicated, strings are reasonably free form (they may not, however, contain a tab character as these delimit values). Exceptions are noted later when the sections are described. To indicate an empty string, the special value "[CLEAR]" should be used (this is necessary since blank fields are always ignored).
Date	A calendar date. The expected format is indicated in the field definition in the header and is based on the current locale. For example, in the US, the default format is "M/D/YY" and December 25, 2011 is represented as "12/25/11".
Time	A time. The expected format is indicated in the field definition in the header and is based on the current locale. For example, in the US the default format is "H:MM A" and five past one in the afternoon is represented as "1:05 PM".
Duration (MS)	A duration in minutes and seconds in the format M:SS. The seconds, SS, must be two characters long and be between 0 and 59. For example, "3:03" is three minutes, three seconds.
Duration (MS, NZ)	A non-zero duration in minutes and seconds in the format M:SS. The seconds, SS, must be two characters long and be between 0 and 59. For example, "3:03" is three minutes, three seconds.
Duration (HMS)	A duration in hours, minutes, and seconds in the format H:MM:SS. The minutes, MM, and seconds, SS, must each be two characters long and be between 0 and 59. For example, "2:03:23" is two hours, three minutes, twenty-three seconds.

A blank *value* is always legal for any of these types. Like blank lines, RCLogbook always ignores blank *values*.

[2] Examples of types include a string, integer number, real number, date, etc.

Changes to the Import Format

The import format that RCLogbook 5.1 uses is not backward compatible with the formats that any prior RCLogbook releases used.

Providing Import Files to RCLogbook

As mentioned above, RCLogbook interprets a user-specified text file to decide what changes it should make to the database. You can provide a text file to RCLogbook through one of the two methods the "Access With" row in the Setup tab selects:

- Web Server – the user connects to RCLogbook through a web browser and selects a text file from their device to upload to RCLogbook.
- Dropbox – RCLogbook uses a text file from Dropbox. This text file may have any name, but must be located in the `/Apps/RCLogbook/Reports` folder in your Dropbox.

Both methods operate in the same fashion once the import RCLogbook retrieves the text file.

In order to use Dropbox, you must have an active account with Dropbox and sign in to Dropbox through RCLogbook. For more information on obtaining a Dropbox account, see http://www.dropbox.com.

Processing an RIX Import File

RCLogbook processes each section from the RIX file in the order in which they appear in the file. Changes from a section are applied to the database in the order in which they appear in the section.

As a simple example, consider a database made up entirely of "character" records. Each such record in the database describes a character and their "friends". For the purposes of this example, we will assume that the import file uses two different types of sections to represent our database of characters: one that describes the properties of a character (the `CHARACTERS` section), and one that describes the friendships (the `FRIENDS` section)[3].

Assume that the example `CHARACTERS` section includes the following fields,

Field	Type	Notes
Action	RIX Action	Action to apply.
ID	RIX ID	Internal unique ID for the character database record.
Character Name	String	Name of the character. All characters are expected to have unique names.
Sith	Boolean	Is the character a Sith?
Weapon	String	Type of weapon the character uses

The `FRIENDS` section includes the following fields,

[3] The import file is split in this fashion in this example to ensure that you can define all the characters before you define any friendships – that is, it lets the import file define who could be friends before actually specifying who are friends.

Field	Type	Notes
Character Name	String	Name of the character.
Friend Name	String	Name of the friend. This field may be continued across multiple lines, one friend per line.
Apprentice	Boolean	Is the friend an apprentice?

The `FRIENDS` section allows you to use continuation lines to specify multiple friends for a character. Together, these sections allow the import file to express the characters and their properties as they appear in the database.

Starting with an empty database, assume that you begin by importing the following RIX file (remember that not all sections need to appear in a file).

```
RCLogbook 5.1.0 Db v.8 (CD) / Text Format
[CHARACTERS]
Action→ID→Character Name→Sith→Weapon
ADD→→Obi Wan→→Lightsaber
ADD→→Darth Sidious→Yes→Force Lightning
```

Since the database is empty, all of the `Action` fields are “`ADD`” to create new records in the database. After we import this file, the database will contain the following information on two newly added characters.

	Database ID	Character Name	Sith	Weapon	Friend Name	Apprentice
Before Import			*Database Initially Empty*			
After Import	325	Obi Wan		Lightsaber		
	102	Darth Sidious	Yes	Force Lightning		

Each row in this table corresponds to a character record from the database. There are several things to take note of:

- RCLogbook determines the database IDs for database records. Though shown here as integers, actual IDs are typically something that looks like a URL.
- If fields are empty, no changes are made to the database. For example, the “`Sith`” field for Obi Wan was left blank in the RIX file implying it is unchanged in the database.
- This import file does not modify any friendships as it lacks a “`FRIENDS`” section.

To add some friendships, we import the following file:

```
1  RCLogbook 5.1.0 Db v.8 (CD) / Text Format
2  [CHARACTERS]
3  Action→ID→Character Name→Sith→Weapon
4  UPD→325→Obi Wan Kenobi→No
5  ADD→→Commander Cody→No→Blaster
6  [FRIENDS]
7  Character Name→Friend Name→Apprentice
8  Obi Wan Kenobi→Commander Cody→No
```

Here is how the database would look before and after importing this file.

	Database ID	Character Name	Sith	Weapon	Friend Name	Apprentice
Before Import	325	Obi Wan		Lightsaber		
	102	Darth Sidious	Yes	Force Lightning		
After Import	325	Obi Wan Kenobi	No	Lightsaber	Commander Cody	No
	102	Darth Sidious	Yes	Force Lightning		
	500	Commander Cody	No	Blaster		

There are several things to take note of:

- This file makes changes to Obi Wan's record (specifically, updating his name and sith status) in the database through line 4 since the "ID" line 4 specifies identifies that record.
- This file adds a new "Commander Cody" character (line 5) and then makes him a friend of "Obi Wan Kenobi" (line 8).
- In general, section order matters. In this example, the `CHARACTERS` section must proceeed the `FRIENDS` section, since the `FRIENDS` section uses names that might be newly added by the `CHARACTERS` section.

To see how continuation lines work, consider the following import file:

```
1   RCLogbook 5.1.0 Db v.8 (CD) / Text Format
2   [CHARACTERS]
3   Action→ID→Character Name→Sith→Weapon
4   [NEW]→→R2-D2→No→
5   [NEW]→→Annakin Skywalker→Yes→Lightsaber
6   [NEW]→→Darth Vader→Yes→Lightsaber
7   [FRIENDS]♍□■
8   Character Name→Friend Name→Apprentice
9   Obi Wan Kenobi→R2-D2→No
10  +→Annakin Skywalker→Yes
11  Darth Vader→Darth Sidious→No
12  Darth Sidious→Darth Vader→Yes
```

Here is how the database would look before and after importing this file.

	Database ID	Character Name	Sith	Weapon	Friend Name	Apprentice
Before Import	325	Obi Wan Kenobi	No	Lightsaber	Commander Cody	No
	102	Darth Sidious	Yes	Force Lightning		
	500	Commander Cody	No	Blaster		
After Import	325	Obi Wan Kenobi	No	Lightsaber	Commander Cody R2-D2 Annakin Skywalker	No No Yes
	102	Darth Sidious	Yes	Force Lightning	Darth Vader	Yes
	500	Commander Cody	No	Blaster		
	123	R2-D2	No			
	657	Annakin Skywalker	No	Lightsaber		
	700	Darth Vader	Yes	Lightsaber	Darth Sidious	No

There are several things to note:

- A continuation line is used to add two more friends to “Obi Wan Kenobi”. Lines 9-10 in the import file together refer to a single character record (that of Obi Wan Kenobi, specifically). In this simple database scenario, you could also accomplish this with two separate “Obi Wan Kenobi” lines.
- Continuation lines are not needed in this example if you only want to add a single friend.
- Note that there is no way to delete a friend through the import file. Any friendship relationships the “Friends” section establishes **add** to the existing relationships (for example, see how Obi Wan’s record changes).

These examples should give you a feel for how the import works. The next section of this document will look specifically at the section types that RCLogbook supports.

Section Types

This portion of the guide summarizes the sections that may appear in an RCLogbook RIX file. Each description includes a table that summarizes the fields in the section. The columns in these tables include:

- *C* – The order of the field (or column number) within a row.
- *Field* – Name of the field, this appears in the field definition row of the section.
- *Type* – Type of data in the field, for the complete list of possible types, see the table in the "Data Types" section on page 7.
- *RN* – "Y" if the field is required for new records, "N" otherwise.
- *Notes* – Details of interest on the field.

The remainder of this section considers each type of section RCLogbook uses in turn.

Models

The `MODELS` section provides general information on the models defined in the database. Each line (excluding the section name and field definition header lines) in the `MODELS` section corresponds to a single model. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the model database record.
3	Model Name	String	Y	Name of the model. All models are expected to have unique names.
4	Type	String	Y	Type of model. Legal values include: • Airplane • Sailplane • Helicopter • Multicopter • Boat • Car • Robot No other values are permitted.
5	Vendor	String	N	Vendor of the model.
6	Category	String	N	Category of the model.
7	Purchase Price (*F*)	Real (P)	N	Purchase price of the model. “F” defines the currency of the price based on the current locale.
8	Retired?	Boolean	N	Identifies whether or not the model is retired.
9	Damaged?	Boolean	N	Identifies whether or not the model is damaged.
10	Total Events	Integer (P)		Total number of events on the model.
11	Total Time	Duration (HMS)	N	Total time on the model in H:MM:SS.
12	Last Event Date (*F*)	Date	N	Date of the last event. “F” defines the format of the date based on the current locale. For example, “F” is “M/D/YY” in the US. This field must be specified if “Last Event Time” is not blank.
13	Last Event Time (*F*)	Time	N	Time of the last event. “F” defines the format of the time based on the current locale. For example, “F” is “H:MM A” in the US. This field must be specified if “Last Event Date” is not blank.
14	Logs Batteries?	Boolean	N	Identifies whether or not the model logs battery information.
15	Logs Fuel?	Boolean	N	Identifies whether or not the model logs fuel information.
16	Fuel Capacity (*F*)	Real (P)	N	Fuel capacity of the model. “F” defines the expected units of the value. Fuel units are selected through the application preferences.
17	Total Fuel (*F*)	Real (P)	N	Total fuel consumed by the model. “F” defines the expected units of the value. Fuel units are selected through the application preferences.
18	Default Fuel	String	N	Default fuel to use for events on this model. The value should match the name of one of the fuels defined in the database.
19	Default Prop	String	N	Default propeller to use for events on this model. The value should match the name of one of the propellers defined in the database.
20	Default Style	String	N	Default event style to use for events on this model. The value should match the name of one of the styles defined in the database.
21	Notes	String	N	User notes on the model. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the `Notes` field.

Batteries

The `BATTERIES` section provides general information on the batteries defined in the database. Each line (excluding the section name and field definition header lines) in the `BATTERIES` section corresponds to a single battery. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the battery database record.
3	Battery Name	String	Y	Name of the battery.
4	Chemistry	String	Y	Chemistry of the battery. Legal values include: • LiPo • LiIon • LiFe • NiCd • LiHV • NiMH • Other No other values are permitted.
5	Vendor	String	N	Name of the vendor or manufacturer of the battery
6	Purchase Price (*F*)	Real (P)	N	Purchase price of the battery. “F” defines the currency of the price based on the current locale.
7	Retired?	Boolean	N	Identifies whether or not the battery is retired.
8	In Storage?	Boolean	N	Identifies whether or not the battery is in storage state.
9	C Rating	Integer (P, NZ)	N	C rating for the battery pack.
10	Capacity	Integer (P, NZ)	Y	Total capacity of the battery pack in mAh.
11	Cells (S)	Integer (P, NZ)	N	Number of series cells in the battery pack.
12	Cells (P)	Integer (P, NZ)	N	Number of parallel cells in the battery pack.
13	Total Cycles	Integer (P)	N	Total number of cycles on the battery pack.
14	Last Cycle Date (*F*)	Date	N	Date of the last cycle. “F” defines the format of the date based on the current locale. For example, “F” is “M/D/YY” in the US. This field must be specified if “Last Cycle Time” is not blank.
15	Last Cycle Time (*F*)	Time	N	Time of the last cycle. “F” defines the format of the time based on the current locale. For example, “F” is “H:MM A” in the US. This field must be specified if “Last Cycle Date” is not blank.
16	Notes	String	N	User notes on the model. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the `Notes` field.

Pilots

The `PILOTS` section provides general information on the pilots defined in the database. Each line (excluding the section name and field definition header lines) in the `PILOTS` section corresponds to a single pilot. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the pilot database record.
3	Pilot Name	String	Y	Name of the pilot.

Continuation lines cannot appear in this section.

Categories

The `CATEGORIES` section provides general information on the model categories defined in the database. Each line (excluding the section name and field definition header lines) in the `CATEGORIES` section corresponds to a single model category. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the category database record.
3	Category Name	String	Y	Name of the category.

Continuation lines cannot appear in this section.

Styles

The `STYLES` section provides general information on the event styles defined in the database. Each line (excluding the section name and field definition header lines) in the `STYLES` section corresponds to a single style. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the style database record.
3	Style Name	String	Y	Name of the style.

Continuation lines cannot appear in this section.

Fuels

The FUELS section provides general information on the fuels defined in the database. Each line (excluding the section name and field definition header lines) in the FUELS section corresponds to a single fuel. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the location database record.
3	Fuel Name	String	Y	Name of the fuel.
4	Price (*F*)	Real	Y	Price of the fuel per unit volume. “F” defines the currency and unit volumes “F” based on the current locale.
5	Notes	String	N	User notes on the fuel. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the Notes field.

Locations

The LOCATIONS section provides general information on the locations defined in the database. Each line (excluding the section name and field definition header lines) in the LOCATIONS section corresponds to a single location. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the location database record.
3	Location Name	String	Y	Name of the location.
4	Latitude	Real	Y	Latitude of the location in degrees (between -180 and 180).
5	Longitude	Real	Y	Longitude of the location in degrees (between -90 and 90).
6	Notes	String	N	User notes on the location. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the Notes field.

Propellers

The `PROPELLERS` section provides general information on the propellers defined in the database. Each line (excluding the section name and field definition header lines) in the `PROPELLERS` section corresponds to a single propeller. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the propeller database record.
3	Prop Name	String	Y	Name of the propeller.
4	Vendor	String	N	Latitude of the location in degrees (between -180 and 180).
5	Units	String	Y	Units that the diameter and pitch are measured in. Legal values include: • in • cm No other values are permitted.
6	Blades	Integer (P, NZ)	Y	Number of blades on the propeller, must be at least 2.
7	Diameter	Real (P, NZ)	Y	Diameter of the propeller in the units specified in the units column.
8	Pitch	Real (P, NZ)	Y	Pitch of the propeller in the units specified in the units column.
9	Notes	String	N	User notes on the propeller. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the `Notes` field.

Maintenance

The `MAINTENANCE` section provides general information on the maintenance log entries defined in the database for various models. Each line (excluding the section name and field definition header lines) in the `MAINTENANCE` section corresponds to a single maintenance log event along with its associated model. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the maintenance log database record.
3	Model	String	Y	Name of the model that the maintenance event is for. This name must exactly match one of the models in the database.
4	Activity	String	Y	Action for maintenance log event.
5	Performed on Date (*F*)	Date	Y	Date of the maintenance event. "F" defines the expected format of the date based on the current locale. For example, "F" is "M/D/YY" in the US. This field must be specified if "Time" is not blank.
6	Performed at Time (*F*)	Time	Y	Time of the last maintenance date. "F" defines the expected format of the time based on the current locale. For example, "F" is "H:MM A" in the US. This field must be specified if "Date" is not blank..
7	Model Event Count	Integer (NZ)	Y	Event number (e.g., flight for aircraft) after which the maintenance occurred.
3	Model Total Time	Duration (HMS)	Y	Total time on model in H:MM:SS when the maintenance occurred.
3	Maintenance Cost (*F*)	Real (P)	Y	Cost of the maintenance. "F" defines the currency of the price based on the current locale.
7	Notes	String	N	User notes on the maintenance log event. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the `Notes` field.

The import file does not allow you to move an existing log entry from one model to another.

Cycles

The `CYCLES` section provides general information on the cycles defined in the database for various batteries. Each line (excluding the section name and field definition header lines) in the `CYCLES` section corresponds to a single cycle of a battery. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply.
2	ID	RIX ID	Y	Internal unique ID for the cycle database record.
3	Cycle Number	Integer (P, NZ)	N	The cycle number for the battery. This field is shown for reference and is appropriately updated automatically. It may be left blank for new batteries.
4	Battery	String	Y	Name of the battery that the cycle is for. This name must exactly match one of the batteries in the database.

C	Field	Type	RN	Notes
5	Ignore in Plots?	Boolean	N	Indicates whether or not the cycle should be ignored and not included in performance plots.
6	Discharge Date *(F)*	Date	Y	Date of the last discharge. “F” defines the expected format of the date based on the current locale. For example, “F” is “M/D/YY” in the US. This field must be specified if “Discharge Time” is not blank.
7	Discharge Time *(F)*	Time	Y	Time of the last discharge. “F” defines the expected format of the time based on the current locale. For example, “F” is “H:MM A” in the US. This field must be specified if “Discharge Date” is not blank.
8	Discharge Duration	Duration (MS)	Y	Duration of discharges in M:SS.
9	Discharge Pack Voltage	Real (P)	N	Final battery pack resting voltage in V after all discharges.
10	Discharge Pack Resistance	Real (P)	N	Final battery pack resistance in mOhm after all discharges.
11	Discharge Cell Voltage	Real List (P)	N	Final battery pack per-cell resting voltages in V after all discharges. The list is in order of P then S, so for a 2S/2P pack, values in the list are in the order “1P/1S ; 1P/2S ; 2P/1S ; 2P/2S”.
12	Discharge Cell Resistance	Real List (P)	N	Final battery pack per-cell resistances in mOhm after all discharges. The list is the same format as the discharge cell voltage list.
13	Charge Date *(F)*	Date	Y	Date of the last charge. “F” defines the expected format of the date based on the current locale. For example, “F” is “M/D/YY” in the US. The charge date and time should be later than the discharge date and time.
14	Charge Time *(F)*	Time	Y	Time of the last charge. “F” defines the expected format of the time based on the current locale. For example, “F” is “H:MM A” in the US. The charge date and time should be later than the discharge date and time.
15	Charge Amount	Integer (P)	Y	Amount of energy put back into battery pack during the charge in mAh. This value must be greater-than zero.
16	Charge Pack Voltage	Real (P)	N	Final battery pack resting voltage in V after all charges.
17	Charge Pack Resistance	Real (P)	N	Final battery pack resistance in mOhm after all charges.
18	Charge Cell Voltage	Real List (P)	N	Final battery pack per-cell resting voltages in V after all charges. The list is the same format as the discharge cell voltage list.
19	Charge Cell Resistance	Real List (P)	N	Final battery pack per-cell resistances in mOhm after all charges. The list is the same format as the discharge cell voltage list.
20	Cycle Notes	String	N	User notes on the cycle. This field may be continued and span multiple lines.

Continuation lines may appear in this section for the `Notes` field.

Cycles are expected to have a charge date and time that always follows the discharge date and time. When editing an existing cycle through the import file, the import file may not change either the charge or discharge date and time such that this expectation is not met. Doing so will cause RCLogbook to report an error.

The import file does not allow you to move an existing cycle from one battery to another.

It is recommended that you group all cycles for a given battery together and place the cycles in most- to least-recent order (by their discharge date and time). This organization will help the import process go more quickly.

RCLogbook handles cycle numbering automatically. Though there is a cycle number field in the import (column 3), it is there primarily for your reference. When editing cycles, the app will determine the appropriate cycle number and update the database as needed.

In addition, when adding a new cycle RCLogbook automatically updates the owning battery appropriately. For example, the total cycle count will be updated as will the charge state and last cycle time.

Events

The `EVENTS` section is the most complex section in the import format primarily because the changes it describes can cover several different types of database records.

The `EVENTS` section provides general information on the events defined in the database for various models along with any battery and cycle information associated with the event. Each line (excluding the section name and field definition header lines) in the `EVENTS` section corresponds to a single event along with its associated model, batteries, and cycles. The fields on each line include,

C	Field	Type	RN	Notes
1	Action	RIX Action	Y	Action to apply to event database record.
2	ID	RIX ID	Y	Internal unique ID for the event database record.
3	Event Number	Integer (P, NZ)	N	The event number for the model. This field is shown for reference and is appropriately updated automatically. It may be left blank for new events.
4	Outcome	String	N	Outcome of the event. Allowed values are numbers between 1 and 4 (inclusive), and “C” (for crashed).
5	Event Date (*F*)	Date	Y	Date of the event. “F” defines the expected format of the date based on the current locale. For example, “F” is “M/D/YY” in the US. This field must be specified if “Event Time” is not blank.
6	Event Time (*F*)	Time	Y	Time of the event. “F” defines the expected format of the time based on the current locale. For example, “F” is “H:MM A” in the US. This field must be specified if “Event Date” is not blank.
7	Event Duration	Duration (MS)	Y	Duration of the event in M:SS.
8	Model	String	Y	Name of the model that the event is for. This name must exactly match one of the models in the database.
9	Pilot	String	Y	Name of the model that the event is for. This name must exactly match one of the models in the database.
10	Location	String	N	Name of the location that the event takes place at. This name must exactly match one of the locations in the database.
11	Fuel	String	N	Name of the location that the event takes place at. This name must exactly match one of the locations in the database.
12	Propeller	String	N	Propeller used in this event. The value should match the name of one of the propellers defined in the `Propellers` section of the import file.

C	Field	Type	RN	Notes
13	Style	String	N	Event style for this event. The value should match the name of one of the styles defined in the `Styles` section of the import file.
14	Fuel Consumed (*F*)	Real (P)	N	Total fuel consumed during the event. "F" defines the units.
15	Event Notes	String	N	User notes on the event.
16	Action	RIX Action	Y	Action to apply to cycle.
17	ID	RIX ID	Y	Internal unique ID for the cycle database record.
Columns 18-36 match columns 3-20 described earlier in the CYCLES section				

Continuation lines may appear in this section for the `Notes` fields as well as the cycles information in columns 16-36.

The `EVENTS` section is slightly different from other sections in the format in that it can include both events and battery cycles. Columns 1-15 provide information on the event while columns 16-36 provide information on any battery cycles associated with the event (these columns will be empty for an event that does not log batteries).

It is helpful to consider a few examples to illustrate how RCLogbook interprets lines that appear in the `EVENTS` section. To simplify the examples, we will use an abbreviated format for the `EVENTS` section that includes only columns 1 (event action), 2 (event ID), 8 (model), 15 (event notes), 16 (cycle action), 17 (cycle ID), 19 (battery), and 36 (cycle notes) from the format listed above.

```
1  [EVENTS]
2  Action    ID        Model     Notes     Action    ID        Battery   Notes
3  ADD➔      ➔         Extra➔    Favorite
```

This example adds a single event to the database on the "Extra" model. This event has no associated cycles. Assuming the event uses a battery,

```
1  [EVENTS]
2  Action    ID        Model     Notes     Action    ID        Battery   Notes
3  ADD➔      ➔         Extra➔    Like!➔    ADD➔      ➔         TP #1➔    Warm
```

This operates the same as the previous example, however it also associates a new cycle on battery "TP #1" with the event. Using continuation lines, you can associate cycles on several batteries with the event,

```
1  [EVENTS]
2  Action    ID        Model     Notes     Action    ID        Battery   Notes
3  ADD➔      ➔         Extra➔    Like!➔    ADD➔      ➔         TP #1➔    Warm
4  ➔         ➔         ➔         ➔         ADD➔      ➔         TP #2➔    Hot
```

Here, the event now uses two batteries, "TP #1" and "TP #2" and the event covers lines 3-4. In addition to continuation lines for additional cycles, there may also be continuation lines for the notes. In the next example, the notes for both the event on model "Extra' and the cycles on batteries "TP #1" and "TP #2" are set to:

```
Line #1
Line #2
```

Specifically,

```
1  [EVENTS]
2  Action    ID        Model     Notes     Action    ID        Battery   Notes
3  ADD→      →         Extra→    Line 1→   ADD→      →         TP #1→    Line 1
4  →         →         →         Line 2→   →         →         →         Line 2
5  →         →         →         →         ADD→      →         TP #2→    Line 1
6  →         →         →         →         →         →         →         Line 2
```

In this example, lines 3-6 encode the event. Lines 3-4 and 5-6 encode the two cycles on batteries.

RCLogbook handles event and cycle numbering automatically. Though there are event and cycle number fields in the import (columns 3 and 20, respectively), they are there primarily for your reference. When editing events, the app will determine the appropriate event and cycle numbers and update the database as needed.

It is recommended that you group all events edits for a given model together and place the events in most- to least-recent order (by date and time). This organization will help the import process go more quickly.

General Guidelines

Here are some general guidelines to help make the import file robust

- It is best to place sections in the import file in the order: `PILOTS`, `CATEGORIES`, `STYLES`, `PROPELLERS`, `FUELS`, `LOCATIONS`, `MODELS`, `BATTERIES`, `EVENTS`, `CYCLES`, and `MAINTENANCE`.
- Remember that later changes over-ride earlier changes. For example, a particular database cycle record can appear in both the `EVENTS` section and the `CYCLES` section. If you make different changes to the record in each of these sections, the last change will "stick".
- Only put things in the import file that you want to change. This implies that you should blank out the values of any fields that you are not changing in a particular set of records and also limit the contents of the import file to the set of records that you want to change.
- When adding or editing cycles associated with an event, it is helpful to only include them in the `EVENTS` section. There is no need to use both the `EVENTS` and `CYCLES` section.
- A template for an import file can be generated from the database access page. You can also use the text export format as a guide.

Following these guidelines will help make sure the import does what you want it to do.

About clevertangerine software

clevertangerine software is a very small software company made up of crafty *Citrus Reticulata* who are committed to developing cool and useful software for the iOS. Though they are just hack pilots, the fruit also love to fly and aspire to someday be able to 3D with the middle of the pack. RCLogbook was the result of tiring of scribbling on scraps of paper at the field to track battery usage.