

# JFile Companion Manual

30th November 2004

### **Abstract**

JFile Companion (JFC) is a Mac application that allows the user to export the content of existing JFile databases as well as create new JFile databases and import data into them. The exporting capabilities of JFile is very flexible and with the help of export templates it is possible to create almost any kind of text based reports, including cross-referenced HTML pages.

JFile is a Palm based database application made by Land-J. JFile is simple to use but still very flexible, for more info about JFile please see Land-J web site at <http://www.land-j.com>.

# Installation

## System Requirements

JFC runs on System 8.6 – 9.2.2 if carbonlib is installed, as well as on OS X 10.1 and later. The manual will assume that you are using OS X and all screenshots etc are made on OS X.

## Installation

You can install JFC by dragging it to any location on your harddisk, although it is most common to install it in the Application folder if you're using OS X. When JFC is used for the first time it will create a folder called "JFC Support" in the "Application Support" folder, which can be found inside the "Library" folder that is found in your "home" folder. Inside the "JFC Support" folder JFC will create two sub-folders, more about them later. The screenshot in Figure 1 describes how the folder structure will look like.

## Shareware

JFC is distributed as shareware, this means that you get to try JFC to see if it is a program that you like. When JFC is run in unlicensed mode there is one limitation, you can only import 50 records into a JFile database. If you start using JFC you should pay the shareware fee, currently \$12, and get a serial number, this serial number will unlock JFC and you can import any number of records into a database.

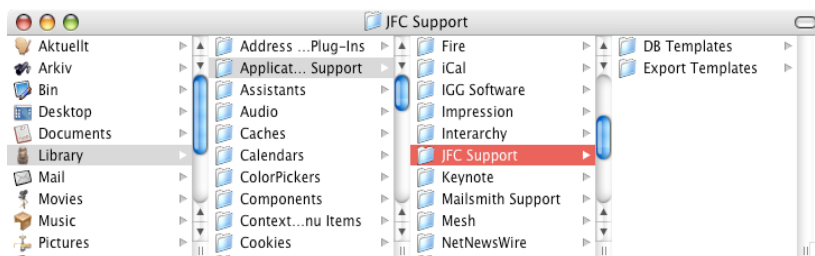


Figure 1: Folder Structure

You can register JFC by going to the following address:

<http://store.eSellerate.net/s.asp?s=STR717615270>

# Exporting JFile Databases

Exporting JFile databases is easy, just double-click the database in the browser window (if it's not open, use the "Browser" command in the "File" menu). JFC will ask you for a file name and then write the content of the database using a tab-delimited format. This text can then be imported into a database, a spreadsheet, etc.

## Exporting Using Export Templates

It's possible for you to customize the output in the way you like, for example creating cross-referenced web pages that can be published on a web server directly. In order to export using templates you must do the following:

1. Create an export template and put it into the "Export Templates" folder inside the "JFC Support" folder. The easiest way to open the folder is to choose the "Open 'Export Templates' Folder" in the "Extras" menu. See the description of Export Templates for more details.
2. Instead of double-clicking the database use the "Convert using template" command in the "Database" menu.
3. JFC will now show you a dialog where you can choose which template to use, see Figure 3.

This dialog will look different depending on the template currently selected. The list to the left will show the files in the "Export Template" folder, select what template to use here. The top field to the right will show comments of the template (if available), below that you will see the currently selected output file/folder. Change this by clicking on the "Set" button.

If the template is a multiple file export, see the Export Templates page, you will be able to define what prefix and suffix to use for file names.

4. When you have made your selections, click "Convert"
5. If you have selected an export template that generates an index, you will be asked which field to use as index test, see Figure 4. Make your selection and click 'Continue'.
6. The database will now be exported according to your specifications.



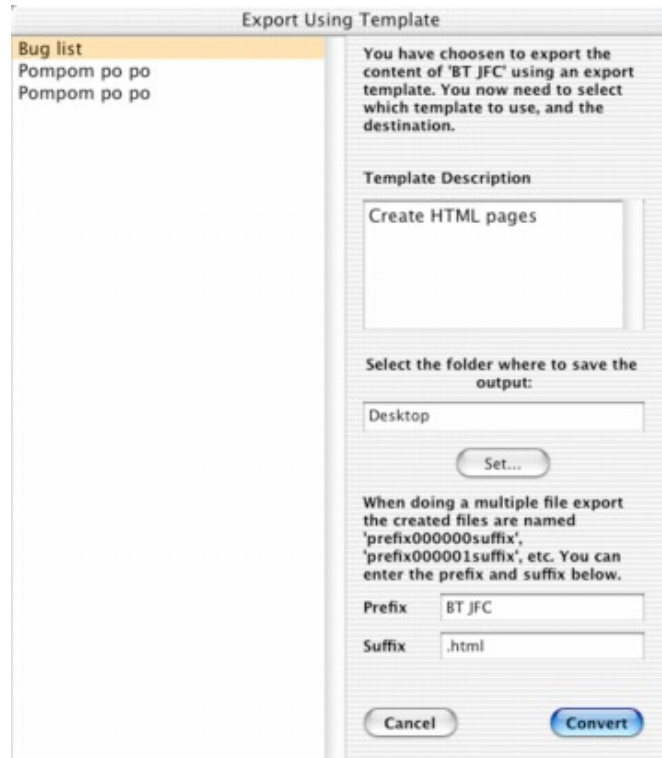


Figure 3: Select what export template to use.

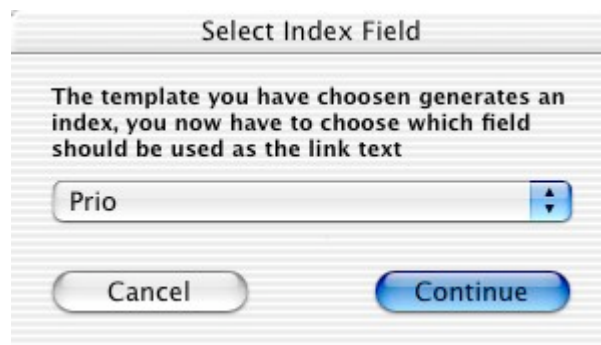


Figure 4: Select what field to use for the index.

# Importing Data

In order to create a JFile database you need two things:

1. A database template that describes how the JFile database should look like, see the page about Import Templates for details.
2. A tab-separated text file, see below, that contains the data you want to import into the JFile database. You can either create this manually or by exporting data from some other application, for example FileMaker or Excel.

When you have these, you can create a JFile database by selecting the "Convert CSV file ..." command in the "File" menu. You will then be presented with the conversion dialog, see Figure 5.

You need to fill in all four fields, the top three are defined by selecting files using the buttons to the right. The fourth field is the name that JFile will display, by default it will be the same as the file name defined in field 3 (except the PDB extension). When all four fields are defined you can click the "Convert" button and the JFile database will be created.

## Tab-separated files

The data should be available as plain text file in one of five formats:

**Tab-separated format** This format separates the different field with a tab character, like this (<tab> represent the tab character):

```
field 1<tab>field 2<tab>field 3<tab>field 4
```

**Comma-separated format, type 1** Here a ',' is used to separate the fields, like this:

```
field 1,field 2,field 3,field 4
```

**Comma-separated format, type 2** Similar to the format above but surrounds each field with "", like this:

```
"field 1","field 2","field 3","field 4"
```

**Semicolon-separated format, type 1** Here a ';' is used to separate the fields, like this:

```
field 1;field 2;field 3;field 4
```

**Semicolon-separated format, type 2** Similar to the format above but surrounds each field with "", like this:

```
"field 1";"field 2";"field 3";"field 4"
```

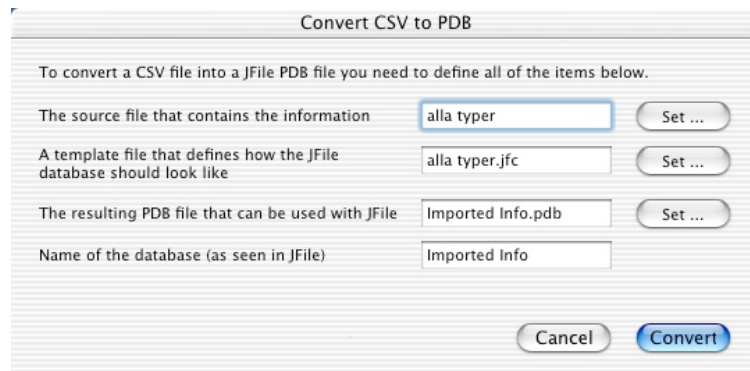


Figure 5: You need to specify these fields to import data into a JFile database.

In all formats one line represents one record. JFC tries to automatically figure out which format to use, it will look at the first line of the file and do the following steps:

1. If the line contains at least one tab-character it will be considered to be in tab-separated format.
2. JFC will count the number of ',' and ';' in the line, if the number of ',' are greater than ';' it will be considered to be a comma-separated file otherwise a semicolon-separated file.
3. The next step is to determine if the file is of type 1 or 2, if the line starts and ends with a " " it is considered to be of type 2 otherwise of type 1

# Designing Databases

In JFile you can create different databases by defining how the different fields should behave. You can do the same in JFC by creating a "template". A template is a description of how a JFile database should look like, JFC allows you to "join" this description with your data to create a JFile database.

So, if you on a regular basis want to import some data into a JFile database you define a template and save it to disk. Then each time you want to create a new database you just "join" the data with the same template.

To create a template, select "New DB Template" from the "Template" menu, this will open the "template definition window", see Figure 6

Select a field in the list and define its properties to the right. The properties are exactly the same as in JFile. Note that to be able to define a field the previous field need to be defined. When the database look the way you like you save it to disk as usual.

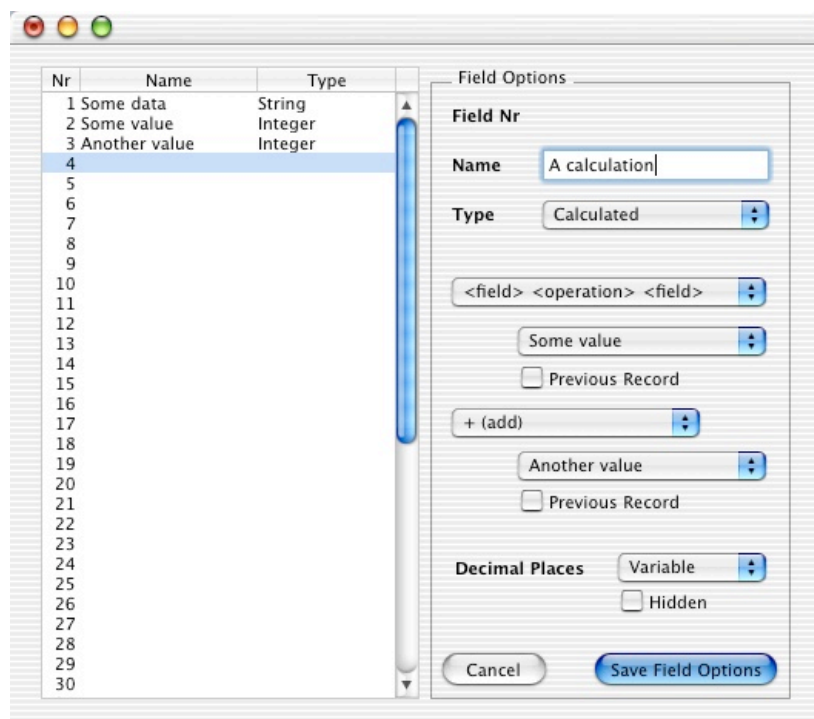


Figure 6: Use this dialog to specify how a database should look like.

# Export Templates

The standard behaviour of JFC is to export a JFile database to a tab-separated text file. This means that each record is represented as one line of text in the file, with each field separated by a tab-character. Although this works very well when the data is going to be imported into a desktop database, it doesn't work as well when you want to format the data in some other way, like for example creating HTML pages.

But with the help of "Export Templates" it becomes possible for you to define how the exported data should look like. It is for example possible to directly create cross-linked HTML pages that directly can be published on a web server.

## The Basics

The idea is quite simple: you create a plain text file and add "tags" where you want to insert different fields. For example:

```
First Name: #First Name#
Last Name: #Last Name#
Email: #Email#
```

Now when JFC exports a database it will replace the "tags" (which always have the form '#Some kind of text#') with data from the database. In this case, the database would need to have three fields called "First Name", "Last Name" and "Email". JFC will look for tags that matches the field names and insert the field data instead. Example: If we have a JFile database with the following content:

First Name	Last Name	Email
Donald	Duck	donald@duck.com
Daisy	Duck	daisy@duck.com

JFC could (more about the different options below) create an output file that would look like this:

```
First Name: Donald
Last Name: Duck
Email: donald@duck.com
First Name: Daisy
Last Name: Duck
Email: daisy@duck.com
```

Or by creating a more advanced template it would be possible to get a result like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```

<title>Email addresses</title>
<meta name="generator" content="BBEdit 7.0.4">
</head>
<body>
<table align="center" border="1">
  <tr><th>First Name</th><th>Last Name</th><th>Email</th></tr>
  <tr><td>Donald</td><td>Duck</td><td>donald@duck.com</td></tr>
  <tr><td>Daisy</td><td>Duck</td><td>daisy@duck.com</td></tr>
</table>
</body>
</html>

```

## Details

Unfortunately, it might look that it is a very complicated task to create an Export Template and to be honest the content might look a bit daunting at (perhaps a template editor will be added some time later). But it's not *that* complicated, please read on.

## The two main parts of a export template

An export template consists of two parts: the option definition part and the description part. A more complicated export template might look like this:

```

#multipleFileExport:Pompom po po#
#useIndex:plutt.htm#

#repeatStart#
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>File nr #number#</title>
  <meta name="generator" content="BBEdit 7.0.4">
</head>
<body>
This is file nr #number#<br>
#HTMLfirst# - #HTMLprev# - #HTMLindex# - #HTMLnext# - #HTMLlast#
</body>
</html>
#repeatEnd#

#indexStart#
#headStart#
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Index</title>
  <meta name="generator" content="BBEdit 7.0.4">
</head>
<body>
<ol>
#headEnd#

#repeatStart#
<li>#indexText#</li>
#repeatEnd#
#footStart#
</ol>
</body>
</html>
#footEnd#
#indexEnd#

```

The first two lines are the option definition part and the rest is the description part.

## Details about the option defintion part

It is possible to control how an export should look and behave by using different standard options. JFC assumes that these options are located at the top of the export template, one option per line, and as soon as it doesn't find a valid option it assumes that the description part has started. Available options are:

**#multipleFileExport#** or **#multipleFileExport:Template name#** Tells JFC that this is a 'multiple files template' (see below). It's possible to define an optional name

of the template. This name is only used by the JFC and is not saved to exported files.

**#singleFileExport#** or **#singleFileExport:Template name#** Tells JFC that this is a 'single file template' (see below). It's possible to define an optional name of the template. This name is only used by the JFC and is not saved to exported files.

**#description:This is a description of the template#** This is a short description of the template, only used by JFC

**#useIndex#** or **#useIndex:indexName#** Create an index for the exported files. Only useful when doing a multiple file export in HTML format. Requires that the template contains an index part. Default name for index file is "index.html", if a file with this name exists it will be named "index1.html" etc. In an optional name, 'indexName' above, is given it will form the name of the index file.

**#HTMLprevLabel:xxxxx#** Label for 'prev' links, if this option isn't defined the text 'Previous' will be used.

**#HTMLnextLabel:xxxxx#** Label for 'next' links, if this option isn't defined the text 'Next' will be used.

**#HTMLfirstLabel:xxxxx#** Label for 'first' links, if this option isn't defined the text 'First' will be used.

**#HTMLlastLabel:xxxxx#** Label for 'last' links, if this option isn't defined the text 'Last' will be used.

**#HTMLindexLabel:xxxxx#** Label for 'index' links, if this option isn't defined the text 'Next' will be used.

## Multiple and Single File Export Templates

JFC can export data in two ways: either by saving all records into a single file (Single File Export Template) or by creating a new file for each record exported (Multiple Files Export Template). A typical use for single file export would be when you want to create a web page with a table of the records, like the Donald Duck example above. A multiple file export is suitable when you have a lot of data and/or long notes for each record, for example a description of the Star Trek Enterprise episodes. By using multiple file export you will get each episode on a separate page.

## Special tags for HTML export

The options listed below will not become meaningful until you know that JFC defines 5 special tags that can be used when you want to create web pages (only useful when multiple file export is used). These special tags are:

**#HTMLnext#** This tag will be replaced by a HTML link to the next page (record). In the case of the last record it will be replaced by an empty string.

**#HTMLprev#** This tag will be replaced by a HTML link to the previous page (record). In the case of the first record it will be replaced by an empty string.

**#HTMLfirst#** This tag will be replaced by a HTML link to the first page (record). If only one record is exported it will be replaced by an empty string

**#HTMLlast#** This tag will be replaced by a HTML link to the next page (record). In the case of the last record it will be replaced by a empty string.

**#HTMLindex#** Only useful for multiple file export. Inserts a relative URL to the index page

## Example

The following example is included in the distribution. The JFile database 'Example' consists of four fields:

1. Priority
2. Date
3. What
4. Description

The export should generate a set of web pages, one for each record + an index page, that can easily be navigated.

Step one is to design how the index page should look like, this is done using your favourite HTML editing tool. For the example above it would look like this (with line numbers to the left):

```

1: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2:   "http://www.w3.org/TR/html4/loose.dtd">
3: <html>
4: <head>
5:   <title>Task Index</title>
6:   <meta name="generator" content="BBEdit 7.0.4">
7: </head>
8: <body>
9: <ol>
10: <li><a href="task000001.html">Task description C</a></li>
11: <li><a href="task000002.html">Task description B</a></li>
12: <li><a href="task000003.html">Task D</a></li>
13: <li><a href="task000000.html">Task description A</a></li>
14: </ol>
15: </body>
16: </html>

```

This HTML code actually contains of three parts: the 'header' - lines 1-9, a repeating body - lines 10-13, and a footer lines 14-16. The repeating part is actually the same line repeated for each record that is exported. So what we need to do is to write some template commands that explains this for JFC. It would look like this (with line numbers to the left):

```

1: #indexStart#
2: #headStart#
3: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4:   "http://www.w3.org/TR/html4/loose.dtd">
5: <html>
6: <head>
7:   <title>Task Index</title>
8:   <meta name="generator" content="BBEdit 7.0.4">
9: </head>
10: <body>
11: <ol>
12: #headEnd#
13:
14:
15: #repeatStart#
16: <li>#indexTitle#</li>
17: #repeatEnd#
18: #footStart#
19: </ol>
20: </body>
21: </html>
22: #footEnd#
23: #indexEnd#

```

**Lines 1 and 23** tells JFC that this is the part that describes how the index should look like.

**Lines 2 and 12** delimits the 'header'.

**Lines 3-11** is the actual HTML code, compare to above

**Lines 15 and 17** delimits the 'repeating' part.

**Line 16** is the part of the index that is repeated for each record. Here we also see the tag `#indexTitle#` which is where JFC should put the identifying text for each record

**Line 18 and 22** describes where the 'footer' is.

The next step is to design how the page for each record should look like. The HTML for one of the example pages looks like this (not pretty but ...):

```

1: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2:   "http://www.w3.org/TR/html4/loose.dtd">
3: <html>
4: <head>
5:   <title>Task</title>
6:   <meta name="generator" content="BBEdit 7.0.4">
7: </head>
8: <body>
9:   <table bgcolor="#FFCC99" width="400" align="center">
10:    <tr bgcolor="#99CCFF"><th colspan="2"><big>My Task List</big></th></tr>
11:    <tr><th align="right">Priority</th><td>3</td></tr>
12:    <tr><th align="right">Date</th><td>2003-8-19</td></tr>
13:    <tr><th align="right">What</th><td>Task description B</td></tr>
14:    <tr><th align="right">Desc</th><td>A description of task B</td></tr>
15:  </table>
16:  <table width="100%" bgcolor="#cccccc" width="500" align="center">
17:    <tr><td width="100%"><a href="task000001.html">First</a></td>
18:    <td width="100%"><a href="task000001.html">Previous</a></td>
19:    <td align="center" width="100%"><a href="taskindex.html">Index</a></td>
20:    <td width="100%" align="right"><a href="task000003.html">Next</a></td>
21:    <td width="100%" align="right"><a href="task000000.html">Last</a></td>
22:  </tr>
23: </table>
24: </body>
25: </html>

```

This is repeated for every record (except for the actual values and links), so the interesting parts are the values and the links. Compare this with the actual template:

```

1: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2:   "http://www.w3.org/TR/html4/loose.dtd">
3: <html>
4: <head>
5:   <title>Task</title>
6:   <meta name="generator" content="BBEdit 7.0.4">
7: </head>
8: <body>
9:   <table bgcolor="#FFCC99" width="400" align="center">
10:    <tr bgcolor="#99CCFF"><th colspan="2"><big>My Task List</big></th></tr>
11:    <tr><th align="right">Priority</th><td>#Priority#</td></tr>
12:    <tr><th align="right">Date</th><td>#Date#</td></tr>
13:    <tr><th align="right">What</th><td>#What#</td></tr>
14:    <tr><th align="right">Desc</th><td>#Description#</td></tr>
15:  </table>
16:  <table width="100%" bgcolor="#cccccc" width="500" align="center">
17:    <tr><td width="100%">#HTMLfirst#</td>
18:    <td width="100%">#HTMLprev#</td>
19:    <td align="center" width="100%">#HTMLindex#</td>
20:    <td width="100%" align="right">#HTMLnext#</td>
21:    <td width="100%" align="right">#HTMLlast#</td>
22:  </tr>
23: </table>
24: </body>
25: </html>

```

Note how the values have been replaced with tags that represents the fields (`#Date#`, `#What#`, etc) in lines 11–14, and how navigational links are inserted by the special tags described earlier, lines 17–21. The get JFC to recognize that this the page the lines needs to be surrounded by `#repeatStart#` and `#repeatEnd#` tags. The complete template looks like this:

```

#multipleFileExport:Task List#
#useIndex:taskindex.html#
#description:Creates HTML pages that lists various tasks to complete#

```

```
#repeatStart#
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Task</title>
  <meta name="generator" content="BBEdit 7.0.4">
</head>
<body>
<table bgcolor="#FFCC99" width="400" align="center">
  <tr bgcolor="#99CCFF"><th colspan="2"><big>My Task List</big></th></tr>
  <tr><th align="right">Priority</th><td>#Priority#</td></tr>
  <tr><th align="right">Date</th><td>#Date#</td></tr>
  <tr><th align="right">What</th><td>#What#</td></tr>
  <tr><th align="right">Desc</th><td>#Description#</td></tr>
</table>
<table width="100%" bgcolor="#cccccc" width="500" align="center">
  <tr><td width="100%">#HTMLfirst#</td>
    <td width="100%">#HTMLprev#</td>
    <td align="center" width="100%">#HTMLindex#</td>
    <td width="100%" align="right">#HTMLnext#</td>
    <td width="100%" align="right">#HTMLlast#</td>
  </tr>
</table>
</body>
</html>
#repeatEnd#

#indexStart#
#headStart#
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Task Index</title>
  <meta name="generator" content="BBEdit 7.0.4">
</head>
<body>
<ol>
#headEnd#

#repeatStart#
<li>#indexTitle#</li>
#repeatEnd#
#footStart#
</ol>
</body>
</html>
#footEnd#
#indexEnd#
```

# Preferences

You can modify how JFC behave by changing the preferences. When you first open the preferences you will be presented with the window you can see in Figure 7. Using the tabs at the top you can select what what options to set.

The first tab allows you to specify if JFC should open the database browser when it is launched. By default JFC uses the extension ".csv" when it converts a PDB file to a text file, if you for some reason would like to have another default extension you can define it here.

In the second tab you can define where JFC should look for the backup and install folder for your Palm. JFC tries to find these folders by itself but in the case it can't find them you have the ability to tell JFC where they are here.

The third tab, Figure 9, you can define the encoding that should be used / expected on different files. What does encoding mean? Computers uses number for representing characters, for example 'A' is represented by the number 65 on most computers, and this works well as long as the text only contains digits, A-Z, a-z and various punctuation characters. However most languages uses other characters, for example Swedish uses the characters Ñ, Ł and Ž. The problem is that these characters gets different numbers on different computers, for example on the Mac 'A' has traditionally been represented by 197 (MacRoman), but most Unix machines uses 197 (ISO Latin 1) for the same character set, etc. The most common encodings today are MacRoman, ISO Latin-1, Windows Latin-1, UTF-8 and UTF-16. JFC can read and generate text in several encodings, currently the first four in the list, and this preference tab allows you to choose between them.

The fourth panel, Figure 10 allows you define some additional conversions that are useful when you export to a CSV file. The first defines a replacement string for newlines that are encountered during conversion (they can cause problem when imported into a database), the default value is is to replace them with a space. The checkbox allows you to define if CSV export should mean "tab separated fields" or if a record should look like this

```
"Content of field 1","Content of field 2"
```

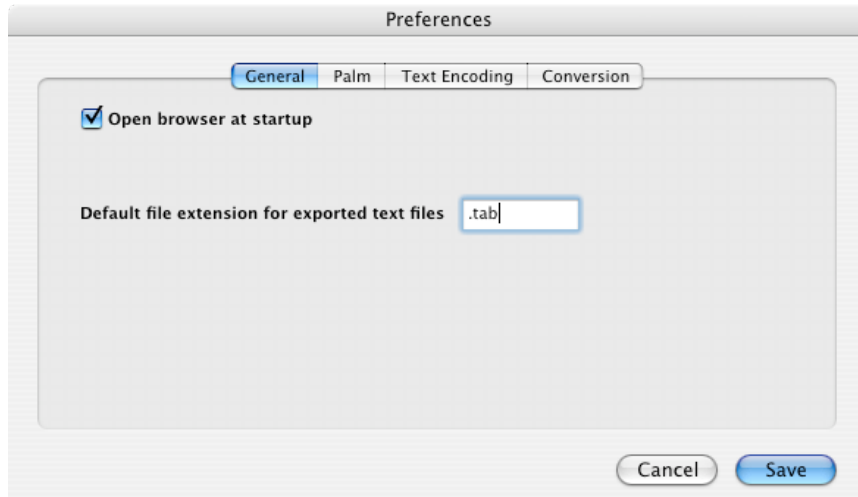


Figure 7: Some general preferences

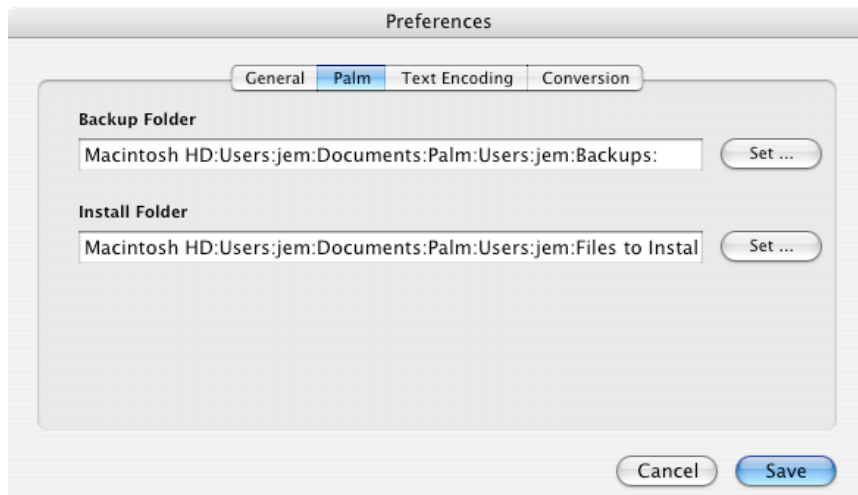


Figure 8: Define where JFC can find the backup folder for your Palm

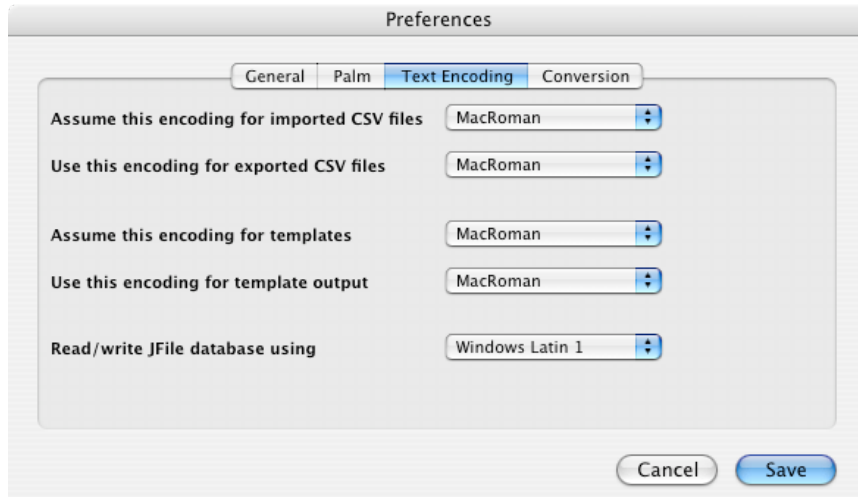


Figure 9: Define the encoding of different files.

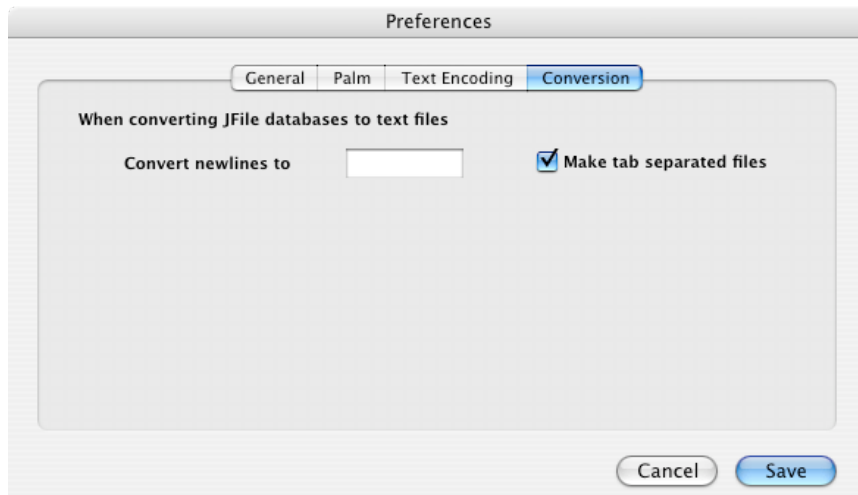


Figure 10: Define some details of how the conversion should be done.

# Support

In the case you have questions, bug reports (spelling mistakes and strange sentences counts as bugs) or suggestions regarding JFile Companion please use web site at <http://support.mootjelitt.se> or send an email to [info@mootjelitt.com](mailto:info@mootjelitt.com). You should will usually receive an answer within a day, however on certain occations there might be a delay upto two weeks.

Don't hesitate in contacting Möötelitt in the case you need a custom version of JFC.

# Version History

## 1.0.1 Changes since previous release

- added an option in the preferences to automatically open the browser window on launch (default on)

## 1.0 First official release

### 1.0b2 Changes since previous release

- The Finders info window should now show the proper version number.
- JFC should now open only JFile databases and give a warning message if some other type of file is encountered.
- If an encrypted database is encountered an error message will be displayed and the conversion will be aborted. Please file a feature request if you think that JFC should support conversion of encrypted databases.
- It's now possible to insert a new field anywhere in the field list: select a field, choose "Insert field" and the selected field and those below it will be moved down one step and the new field will be inserted in its place.
- Fixed a bug when converting "true" CSV files where a comma inside a field would be interpreted as a field separator.
- Fixed a bug where the menubar wouldn't update properly at startup.

### 1.0b1 Changes since previous release

- It's now possible to tell JFC what newlines should be replaced with when a JFile database is exported to a CSV field.
- JFC will now see files that end in ".csv", ".txt" as well as plain text files.
- It's now possible to do "true" comma separated export (uncheck the "Make tab separated file").
- Fixed a bug where extra records would be exported under certain circumstances.
- Fixed a bug which prevented JFC to run on OS 9. Note that there is a special classic version that should be used on OS 9.

### 1.0d8 Changes since previous release

- Major rewrite of code, this should not be noticed by the user
- Fixed a bug where the save option button didn't activate

- Several bug fixes
- Fields can now be reordered
- Fields can now be deleted

**1.0d7** Non-public release

**1.0d6** Changes since previous release

- Fixed a bug that would prevent JFile from detecting certain records.
- Added automatic detection of data file format, see "Import" chapter for details
- Set better default values for column and data width.

**1.0d5** Changes since previous release

- Added support for export templates
- JFC creates a folder in the Application Support folder: "JFC Support". Used for storing JFC related files.
- The encoding popups should now work

**1.0d4** First public beta