Fanurio User Guide

Fanurio Time Tracking

Fanurio User Guide

Fanurio Time Tracking

Publication date December 2017
Copyright © 2006-2017 Fanurio Time Tracking SRL

Table of Contents

1. Introduction 1
    1.1. About Fanurio 1
    1.2. What's new 3
    1.3. Tips for first-time users 4
2. Tutorials 5
    2.1. How to use projects to manage work and expenses 5
    2.2. How to track time for your work 6
    2.3. How to create a project invoice 7
    2.4. How to create an invoice template with your logo and layout 8
3. Using Fanurio 9
    3.1. Setting your preferences 9
        3.1.1. Language 9
        3.1.2. Global hotkeys 9
        3.1.3. Email 10
        3.1.4. Updates 10
        3.1.5. Timer 10
    3.2. Configuring your business 11
        3.2.1. Registration and contact 11
        3.2.2. Currencies 11
        3.2.3. Billing 11
        3.2.4. Projects 12
    3.3. Working with clients 13
        3.3.1. About clients 13
        3.3.2. Hiding inactive clients 14
        3.3.3. Billing clients 14
        3.3.4. Billing foreign clients 14
        3.3.5. Importing clients from CSV 15
        3.3.6. Importing clients from Apple Contacts 15
        3.3.7. Exporting clients 15
    3.4. Working with projects 16
        3.4.1. About projects 16
        3.4.2. Choosing a projects view 16
            3.4.2.1. The projects tree view 17
            3.4.2.2. The projects table view 17
        3.4.3. Configuring optional fields 17
        3.4.4. Numbering projects automatically 18
        3.4.5. Billing projects 18
        3.4.6. Duplicating a project 19
        3.4.7. Hiding finished projects 19
        3.4.8. Using tags to organize projects (table view only) 19
        3.4.9. Searching and filtering projects (table view only) 19
        3.4.10. Exporting projects in the iCalendar format 20
        3.4.11. Creating projects reports 21
    3.5. Working with tasks 22
        3.5.1. About tasks 22
        3.5.2. Configuring optional fields 23
        3.5.3. Billing tasks 23
        3.5.4. Billing a task multiple times 24
        3.5.5. Billing a task at different rates 25
        3.5.6. Marking invoiced tasks as completed 25
        3.5.7. Rounding time for billable tasks 26
        3.5.8. Understanding billable time precision 26
        3.5.9. Planning work with tasks 27
        3.5.10. Tracking progress on tasks 28
        3.5.11. Using categories to organize tasks 28

3.5.12. Using tags to organize tasks 28
3.5.13. Searching and filtering tasks 28
3.5.14. Exporting tasks 29
3.5.15. Creating tasks reports 29
3.6. Working with time entries 31
3.6.1. About time entries 31
3.6.2. Configuring optional fields 31
3.6.3. Recording time manually 32
3.6.4. Recording time with timers 32
3.6.5. Using reminders to control the active timer (smart timing) 34
3.6.6. Recording time to the second 35
3.6.7. Recording quantifiable activities 35
3.6.8. Billing time 36
3.6.9. Importing time from CSV 37
3.6.10. Importing time from iCalendar 39
3.6.11. Using tags to organize time entries 40
3.6.12. Searching and filtering time entries 41
3.6.13. Exporting time entries 41
3.6.14. Creating time reports 41
3.7. Working with expenses 43
3.7.1. About expenses 43
3.7.2. Configuring optional fields 43
3.7.3. Billing expenses 44
3.7.4. Using categories to organize expenses 44
3.7.5. Using tags to organize expenses 44
3.7.6. Searching and filtering expenses 44
3.7.7. Exporting expenses 45
3.7.8. Creating expenses reports 45
3.8. Working with trips 46
3.8.1. About trips 46
3.8.2. Configuring optional fields 47
3.8.3. Billing trips (mileage) 47
3.8.4. Using tags to organize trips 48
3.8.5. Searching and filtering trips 48
3.8.6. Exporting trips 48
3.8.7. Creating trips reports 48
3.9. Working with products 50
3.9.1. About products 50
3.10. Working with invoices 51
3.10.1. About invoices 51
3.10.2. Configuring optional fields 51
3.10.3. Creating a regular invoice 51
3.10.4. Creating a project invoice 52
3.10.5. Using taxes 52
3.10.6. Numbering invoices automatically 53
3.10.7. Billing in multiple currencies 54
3.10.8. Discounting an invoice 55
3.10.9. Discounting individual items 55
3.10.10. Cancelling an invoice 55
3.10.11. Using templates to view, export and email invoices 56
3.10.12. Exporting an invoice 56
3.10.13. Sending invoices by email 56
3.10.14. Searching and filtering invoices 56
3.10.15. Exporting invoices 57
3.10.16. Creating sales reports 57
3.11. Working with payments 59
3.11.1. Recording payments for an invoice 59
3.11.2. Using deposits 59

3.11.3. Searching and filtering payments 59
3.11.4. Exporting payments 59
3.12. Fanurio + QuickBooks 60
3.12.1. Initial setup 60
3.12.2. Exporting time from Fanurio 60
3.12.3. Importing time in QuickBooks Pro 61
3.13. Fanurio + IGG Software's iBiz 62
3.13.1. How to import an iBiz database 62
3.13.1.1. What to do if the conversion fails? 67
3.13.1.2. What to do if the conversion succeeds with errors? 67
3.13.1.3. Which iBiz elements are imported by Fanurio? 67
3.13.2. How is Fanurio different from iBiz? 68
3.13.2.1. Clients 68
3.13.2.2. Projects 69
3.13.2.3. Job events 69
3.13.2.4. Job event groups 71
3.13.2.5. Custom job events 71
3.13.2.6. To Do items 71
3.13.2.7. Timers 71
3.13.2.8. File Cabinet 72
3.13.2.9. Estimates 72
3.13.2.10. Invoices 72
3.13.2.11. Invoice Reminders 72
3.13.2.12. Statements 72
3.13.2.13. Taxes 73
3.13.2.14. Payments 73
3.13.2.15. Templates 73
3.13.2.16. Reports 73
3.13.2.17. Document Monitor 73
3.13.2.18. The workday pane 73
3.13.2.19. iBank integration 73
3.13.2.20. Applescript and Automator integration 74
3.13.2.21. Networking 74
3.13.2.22. Platforms 74
3.14. Keyboard shortcuts 75
4. Templates guide: How to create and edit templates for Fanurio 78
4.1. Getting started 78
4.1.1. Creating an invoice template in 10 minutes or less 78
4.1.2. Installing a template 78
4.1.3. About Freemarker 79
4.2. Using the template editor 80
4.2.1. Basic settings 80
4.2.2. Adjust the look and format using CSS (for advanced users only!) 81
4.3. The anatomy of a simple template 82
4.3.1. Placeholders or how to access fields 82
4.3.2. Built-ins or how to get more details about a field 83
4.3.3. Directives or how to perform operations on fields 83
4.4. Charts (for advanced users) 87
4.4.1. The jfreechart directive 87
4.4.2. Chart types 90
4.4.2.1. Area charts 90
4.4.2.1.1. Category area charts 90
4.4.2.1.2. XY area charts 93
4.4.2.1.3. Time series area charts 96
4.4.2.2. Bar charts 99
4.4.2.2.1. Bar charts 100
4.4.2.2.2. Stacked bar charts 102
4.4.2.3. Line charts 108

4.4.2.3.1. Category line charts 108
4.4.2.3.2. XY line charts 113
4.4.2.3.3. Time series line charts 116
4.4.2.4. Pie charts 120
4.4.2.4.1. Pie charts 120
4.4.2.4.2. Ring charts 122
4.4.3. Chart properties 124
4.4.3.1. Chart 131
4.4.3.2. Title and subtitles 133
4.4.3.3. Legend 134
4.4.3.4. Plot 136
4.4.3.5. Renderer 151
4.4.3.6. Domain axis 163
4.4.3.7. Range axis 170
4.5. Creating templates the easy way 176
4.5.1. Using directives in tables (important!) 176
4.5.2. HyperText Markup Language (.html) 177
4.5.3. Microsoft Word (.docx) 177
4.5.4. OpenDocument Text (.odt) 178
4.5.5. OpenDocument Spreadsheet (.ods) 178
4.6. Creating templates like a professional (HTML and CSS) 179
4.6.1. Page formatting 179
4.6.2. Fonts 182
4.7. Creating templates like a professional (Plain Text and XML) 184
4.7.1. Example: Quicken 184
4.7.2. Example: Scribus (DTP) 185
4.8. A comparison of all supported file formats 187
4.9. Placeholders reference 189
4.9.1. Business 189
4.9.2. Client 190
4.9.3. Invoice 191
4.9.4. Item 198
4.9.5. Payment 205
4.9.6. Deposit 205
4.9.7. Tax 206
4.9.8. Invoice Project 206
4.9.9. Time Entry 209
4.9.10. Task 210
4.9.11. Expense 212
4.9.12. Trip 212
4.9.13. Person 213
4.9.14. Projects Report 213
4.9.15. Time Report 215
4.9.16. Tasks Report 219
4.9.17. Expenses Report 222
4.9.18. Trips Report 224
4.9.19. Sales Report 227
5. Troubleshooting and maintenance 230
5.1. Installing Fanurio 230
5.1.1. Requirements 230
5.1.2. Setting up Fanurio for the first time 230
5.1.2.1. The standalone version 231
5.1.2.2. The server version 232
5.1.3. Upgrading from the standalone version to the server version 232
5.1.4. Running Fanurio over SSL (HTTPS) 233
5.1.4.1. Enabling SSL on the server 233
5.1.4.2. Creating an SSL keystore using the Java keytool 233
5.1.5. Whitelisting trusted IPs 235

5.1.6. How synchronization works in client mode 236
5.1.7. Changing the server address in client mode 237
5.1.8. Changing the password 237
5.1.9. Permissions 237
5.1.9.1. Roles 237
5.1.9.2. Assigning roles 241
5.1.9.3. Granting and revoking project access 241
5.1.10. Registering a license key 242
5.1.11. Importing data from other applications 242
5.1.12. Installing Fanurio on a USB stick 242
5.1.13. Uninstalling Fanurio 243
5.2. User data 245
5.2.1. Location 245
5.2.2. About repositories 246
5.2.3. About backups 247
5.2.4. Creating and restoring backups 247
5.2.5. How single users can sync data between multiple computers 248
5.2.5.1. How file synchronization works 248
5.2.5.2. Using the server version 249
5.2.5.3. Using a file-synchronization service (Dropbox) 249
5.2.6. Exporting the log file 250
5.2.7. Changing settings manually 250
5.2.8. Translating to other languages 251
5.2.9. Installing a custom language file 251
5.2.10. Password encryption 252
5.3. Known issues 253
5.3.1. Printing 253
5.3.2. Tray icon 253
5.3.3. Ubuntu time zone 253
5.3.4. Ubuntu 13.10 mini timer 254
5.3.5. White or black window on Windows 7 254
5.3.6. Email servers with untrusted security certificates 254
5.4. More help 255
6. Changelog 256
6.1. Version 4.0-beta3 (December 12, 2017) 256
6.2. Version 3.2.1 (December 12, 2017) 257
6.3. Version 3.2 (November 2, 2017) 258
6.4. Version 3.1.2 (July 12, 2016) 264
6.5. Version 3.1.1 (September 7, 2015) 265
6.6. Version 3.1 (March 11, 2015) 266
6.7. Version 3.0.1 (March 31, 2014) 274
6.8. Version 3.0 (December 3, 2013) 276
6.9. Version 2.7 (July 9, 2013) 282
6.10. Version 2.6.1 (March 29, 2013) 284
6.11. Version 2.6 (November 6, 2012) 285
6.12. Version 2.5 (February 28, 2012) 288
6.13. Version 2.4.3 (November 10, 2011) 291
6.14. Version 2.4.2 (June 10, 2011) 292
6.15. Version 2.4 (January 31, 2011) 293
6.16. Version 2.3 (August 17, 2010) 295
6.17. Version 2.2 (October 14, 2009) 298
6.18. Version 2.1.1 (July 22, 2009) 300
6.19. Version 2.1 (July 9, 2009) 301
6.20. Version 2.0 (February 26, 2009) 304
6.21. Version 1.11.3 (October 9, 2008) 307
6.22. Version 1.11 (July 29, 2008) 308
6.23. Version 1.10 (April 22, 2008) 310
6.24. Version 1.9 (February 4, 2008) 313

6.25. Version 1.8 (November 12, 2007) 316
6.26. Version 1.7 (September 3, 2007) 318
6.27. Version 1.6 (July 2, 2007) 320
6.28. Version 1.5 (May 28, 2007) 322
6.29. Version 1.4 (April 18, 2007) 324
6.30. Version 1.3 (March 1, 2007) 326
6.31. Version 1.2 (February 12, 2007) 327
6.32. Version 1.1 (January 16, 2007) 328
6.33. Version 1.0.2 (December 5, 2006) 329
6.34. Version 1.0.1 (November 24, 2006) 330
6.35. Version 1.0 (November 10, 2006) 331

Chapter 1. Introduction

1.1. About Fanurio

If you are new to Fanurio, this section provides a quick introduction to what it does and its main features.

Fanurio provides freelancers with the right tools to track time, manage projects and invoice their clients so they can get paid for their work. Fanurio is a cross-platform desktop application that runs on Windows, Mac OS X, Linux and other flavors of Linux.

Fanurio uses billable projects to organize tasks, expenses, trips and products that must be sold to clients. Internal projects can be marked as non-billable. Here are some of the things projects can record.

- Work can be planned, managed and billed using tasks. A task can be used to bill three hours of programming or ten pages of Greek text translated to Latin, it's up to you how you bill your work. Tasks can be billed in units or hours.

- The time spent working on tasks can be recorded manually or with a timer. To make time tracking more accurate, Fanurio can detect if you leave the computer and forget the timer running (idle time). Fanurio has a few reminders to help you start, resume or stop a timer. Instead of relying on your memory and attention to control the timer, you can use these reminders.

- Expenses record money spent for a project. Billable expenses can be billed at their actual cost or marked up.

- Trips record the distance and time travelled with a vehicle whether it's for billing or for tax purposes. Distances can be tracked in miles, kilometers or both.

- Products can be used to bill something to a client that is neither a task nor an expense. A product could be a computer, a set of icons or a monthly fee for website hosting services.

When it comes to billing, Fanurio has many features that can help freelancers get paid.

- Fanurio can export invoices to HTML, PDF, Microsoft Word 2007, OpenDocument and other formats so they can be printed or e-mailed. Invoice templates can be created manually, with a visual editor (Adobe Dreamweaver, Microsoft Word or LibreOffice) or with the built-in template editor.

- Billable time can be rounded up, down or to the nearest specified interval. Fanurio supports complex time rounding rules like "round time up to 15 minutes for each time entry and bill at least 30 minutes".

- Discounts can be used to offer introductory rates (discounted prices) to first time clients or to bill less work if a service takes longer than estimated. Discounts can be applied to individual items or to the whole invoice.

- Fanurio can deal with a whole range of taxes and tax combinations. Invoices can use one, two or more taxes (cummulative or not).

- Fanurio can work with multiple currencies.

Fanurio comes with many reports that help you see how much money you've made or how much time you've worked.

Fanurio can import data from other applications. It can import time from any CSV file and clients from a CSV file or from Apple Contacts. If you need to use data outside Fanurio, you can export it to CSV or Excel.

Fanurio stores its data on the local hard-drive but it can be shared across multiple computers if it's saved in a Dropbox (or a similar service) folder. Fanurio performs automatic backups to ensure the integrity of its data.

Fanurio can be used in other languages than English.

1.2. What's new

If you are using an older version of Fanurio, you may want to read the changelog to see what's new.

We always recommend upgrading to the last version as it may contain critical bug fixes and improvements. Even if you decide not to upgrade, you may still want to read the changelog to be aware of the changes.

1.3. Tips for first-time users

As a first-time user, you should go through all the tutorials to learn the basics. Then you can browse the manual to learn about other features not covered by the tutorials.

If you have any questions, you can always contact us to help you out or go to the troubleshooting section to find answers to common questions.

Chapter 2. Tutorials

The following tutorials are meant to help you get started with Fanurio. Complete all of them and you should learn how to use the important features of the application.

2.1. How to use projects to manage work and expenses

Fanurio uses projects to organize work and expenses. This tutorial shows how to create a fictional project for proofreading and printing a manuscript.

1. Go to Business » New Client to create a client. Enter *Aristotle* in the **Name** field and click **Create**.

   The client will be displayed in the projects tree.

2. Go to Business » New Project to create a project. Enter *Rhetoric* in the **Name** field and click **Create**.

   The project will be displayed under the client in the projects tree. It will be opened automatically and you should see an empty Tasks table on the right. This table displays the tasks of the project.

3. Click the **New** button placed under the tasks table to create a task.

   a. Enter *Proofreading the manuscript* in the **Name** field.

   b. Make sure the **Billable** box is checked.

   c. Click **unit-based** in the **Pricing** field.

   d. Enter *140* in the **Billable Quantity** field.

   e. Enter *pages* in the **Unit of Measure** field.

   f. Enter *6* in the **Price** field.

   g. Click **Create**.

   You'll be charging Aristotle 6 USD (or whatever currency you are using) per page for proofreading a manuscript of 140 pages.

4. Switch the project view to Expenses.

5. Click the **New** button placed under the expenses table to create an expense.

   a. Enter *60* in the **Amount** field.

   b. Enter *Printing the manuscript* in the **Description** field.

   c. Check the **Billable** box.

   d. Click **Create**.

   After proofreading the manuscript, you also have to print three copies. You can't print them yourself so you have to pay someone else to do it. They ask you 20 USD (or whatever currency you are using) for each copy which will cost 60 USD. This is a reimbursable expense, that's why it's marked as billable.

If you want to know more about tasks, there's a section that explains their properties and how they work. Expenses are explained in more detail in a separate section.

2.2. How to track time for your work

You may want to track time for two reasons: to bill your customers if you charge by the hour and to know where the time goes (even if you don't bill it). This tutorial continues the previous lesson and shows how to record time to a task using time entries.

1. Make sure the *Rhetoric* project is open and click the **New** button to create a task.

   a. Enter *Administrative work* in the **Name** field.

   b. Make sure the **Billable** box is not checked.

   c. Click **Create**.

   You'll create a non-billable task to track the time spent on administrative work, things that you do for the project but are not billed to the client. Now you have two tasks that can be used to track time (*Proofreading the manuscript* and *Administrative work*).

2. Right-click *Administrative work* in the tasks table and select **New Time** from the contextual menu to add time to it.

   a. Enter a past date in the **Date** field.

   b. Type *10:00* in the **Start** field.

   c. Enter *1:30* in the **Time** field.

   d. Type *Initial meeting* in the **Description** field.

   e. Click **Create**.

   The Time column for this task will be updated and will display 1:30.

3. Repeat step #2 and add a new time entry of 45 minutes with the description *Final meeting*. Now the Time column should display 2:15.

4. Repeat step #2, this time for the *Proofreading the manuscript* task and add two time entries. One entry for 6 hours that has the description *First 100 pages* and one entry for 2:52 hours with the description *The last 40 pages*. The Time column for this task should now display 8:52.

   Once you're done, you can notice that the total time below the table is 11:07 which represents the time recorded for all tasks.

5. Go to View » Timesheet to switch the view from Projects to Timesheet.

   The Timesheet view is the place where you can see all time entries. You can return to this view whenever you need to get the big picture or to review your time.

Time entries are explained in more detail in a separate section. You'll learn how to move time entries from one task to another and how to create time reports.

2.3. How to create a project invoice

Once you have one or more billable tasks, expenses or products in a project, you can invoice the client.

This tutorial continues the previous lessons and shows how to create an invoice. It also shows how you can send it to the client.

1. Make sure you are in the Projects view. If you are not, go to View » Projects to change the view.
2. Click the *Rhetoric* project in the Projects tree to select it.
3. Go to Business » New Invoice to create a new invoice. Fanurio will display the Add Project Items window to let you choose the project items (tasks, expenses, trips or products) for the invoice.

   a. The task *Proofreading the manuscript* and the expense *Printing the manuscript* are checked by default. Click Add to add them to the invoice.

   b. Type *1* in the **Number** field.

   c. Click **Create**.

   Once an invoice is created, the view is automatically switched to Invoices in order to display the new invoice.
4. Right-click the invoice to display the contextual menu and select **Export Invoice**.
5. Select a folder where you want to export the invoice if you don't want to use the default one.
6. Click **Export**.

   Fanurio will export the invoice as a PDF file and it will open it. You can send the PDF invoice to Aristotle by email as an attachment or print it and send it by mail.

   The invoice is exported using the default template that comes with Fanurio. The next lesson will show how you can customize this template to look exactly like you want it to.

Invoices are explained in more detail in a separate section. You'll learn how to number then automatically and how to create invoice reports.

2.4. How to create an invoice template with your logo and layout

To create your first invoice template, you only need to follow three easy steps. This tutorial is part of a separate guide that explains how to create a template from scratch.

Chapter 3. Using Fanurio

3.1. Setting your preferences

This chapter presents the most important preferences that can be configured at application level.

- Go to Tools » Options to access the preferences on Windows
- Go to Fanurio » Preferences to access the preferences on Mac
- Go to Edit » Preferences to access the preferences on Linux

3.1.1. Language

Fanurio is used by people all over the world and even though English is used by many people, some prefer to run software in their native language. Fanurio is available officially in the following **languages**:

- Czech
- Dutch
- English
- French
- German
- Italian
- Portuguese
- Romanian
- Spanish

The language can be changed from the **Locale** section. Only the user interface of the application is translated. The user manual and technical support are available only in English.

Fanurio is translated in a few other languages but these translations are not finished yet. The **unfinished languages** are:

- Chinese
- Danish
- Finnish
- Swedish

Read this section if you want to learn how to create your own translation or how to change an existing one. Read this other section to learn how to install a custom translation so that you can use it with Fanurio.

3.1.2. Global hotkeys

Global hotkeys are keyboard shortcuts that can be used from within any running application. They are not enabled by default. To enable them, go to the **System** section. You can use the default shortcuts or click the Edit link to define your own.

Note: Currently, only the Windows and Linux versions of Fanurio have support for global hotkeys.

Table 3.1. Default global hotkeys for the Windows version

Key	Action
Ctrl-Alt-F5	Start new timer immediately
Ctrl-Alt-Shift-F5	Start new timer
Ctrl-Alt-F6	Pause/Resume timer
Ctrl-Alt-F7	Stop timer
Ctrl-Shift-F	Shows the application window

Table 3.2. Default global hotkeys for the Linux version

Key	Action
Ctrl-Alt-Insert	Start new timer immediately
Ctrl-Alt-Shift-Insert	Start new timer
Ctrl-Alt-Home	Pause/Resume timer
Ctrl-Alt-End	Stop timer
Ctrl-Shift-F	Shows the application window

3.1.3. Email

Fanurio can send invoices by email as attachments. Before you do this, make sure you have a valid email address for your business and the outgoing email server is configured correctly.

Go to the **Email** section to enable the email feature and to configure the outgoing email server (SMTP). You can test your settings by clicking the **Test** button.

Note: Read this section to learn how Fanurio encrypts passwords. If you specify a server that uses untrusted certificates, you will need to configure Fanurio to be aware of them.

3.1.4. Updates

Whether we implement new features or improve existing ones, you will be notified automatically when a new version is available. Go to the **Update** section to uncheck this option if you don't want to be notified automatically.

3.1.5. Timer

You can learn more about the smart timing settings by reading this section.

3.2. Configuring your business

Once you open a repository, everything you record in Fanurio is for the same business. A business can manage a list of clients, a list of projects, and so on. A business is set up when a repository is created and it can't be deleted, you can only change its settings from Business » My Business Details.

The settings of a business are organized in four sections and are explained below.

3.2.1. Registration and contact

The name, registration and contact details of your business are optional, but if present they can be included in reports and invoices. On OS X you can update your contact details from Apple Contacts.

Here's what each registration field means:

- **Business Number:** The number assigned to your business when it was registered. Some users need to display this number on their invoices.
- **Tax Number:** The tax number assigned by the government to your business. In many countries, invoices must include the client's VAT number. In such cases, the tax number is the VAT number.
- **Other Number:** A generic field that could be used to record the trade register number.

3.2.2. Currencies

Fanurio needs at least one currency if you use it for billing or to track expenses. If you don't use Fanurio for billing or to track expenses then you don't need to worry about currencies.

Usually, you don't need to define any currency since Fanurio automatically detects your currency from your computer settings. But if it doesn't detect it correctly or if your business bills in multiple currencies, this is the place where you can make the changes.

3.2.3. Billing

Billing is optional and can be enabled or disabled at business level but it can't be disabled if you have at least one invoice or product.

If billing is enabled, you can create regular invoices that bill clients directly and project invoices that bill their projects. Also, if billing is enabled you can configure the following settings:

- **Default terms:** Terms specify when an invoice is due. The default terms specified at business level are used when new clients are created. Clients have their own terms that may be different from the business terms.
- **Catalog:** If your business sells some items more frequently, you can add them to the business catalog. Catalog items make it easier to create regular invoice items.
- **Financial year:** The start date of the financial year makes it possible to select the current or the last financial year when creating sales reports (Reports » Sales Report) or when filtering invoices and payments. The default start date for the financial year is January 1st.
- **Invoice numbering**
- **Taxes**

If billing is disabled, the user interface is updated to remove all billing references:

- The Invoices and Payments views are removed.

- The menu and toolbar actions related to invoices and payments are removed.
- All billing columns are removed from the projects, tasks, expenses and trips tables.
- All billing filters are removed from all views.
- All clients, projects, tasks, expenses and trips are changed to non-billable.
- Projects can no longer manage products.

3.2.4. Projects

The Projects section contains projects-related settings like project numbering. Most settings are organized by the type of element a project can manage (tasks, expenses, trips, and products).

- The Tasks section allows you to manage the list of task categories that you can use to organize tasks. If billing is enabled, you can also:
  - tell Fanurio whether tasks should be marked as completed when they are invoiced,
  - set a default time rounding rule for your billable tasks and
  - enable or disable (not recommened) exact precision for billable time.
- Expenses are optional and can be enabled or disabled at business level. If expenses are enabled, you can also manage the list of expense categories that you can use to organize your expenses.
- Trips are optional and can be enabled or disabled at business level. If trips are enabled and billing is enabled, you can also manage the list of trip rates that you can use to bill your trips.

  Also, if trips are enabled, the Trips section allows you to specify the default distance unit (kilometer or mile) for your trips.
- Products are optional and can be enabled or disabled at business level but only if billing is enabled. If billing is not enabled, products are not available for projects.

3.3. Working with clients

3.3.1. About clients

You need clients to record everything you do for them, billable or not. Clients have the following properties:

- **Name:** The name of the client.
- **Registration**
  - **Business Number:** The number assigned to the business when it was registered. Some users need to display this number on their invoices.
  - **Tax Number:** The tax number assigned by the government to your client's business. In many countries, invoices must include the client's VAT number. In such cases, the tax number is the VAT number.
  - **Other Number:** A generic field that could be used to record the trade register number.
  - **Code:** A code that is useful if you want to number invoices automatically and to include it in the invoice number. For instance, if your client name is Acme Corporation and their code is ACME then their invoices could be numbered like this: ACME-001, ACME-002, etc.
- **Contact**
  - Attention
  - Address, City, State, Zip
  - Phone, Fax, Mobile, Other
  - Email, Website
- **Notes:** Additional notes.
- **Active:** Whether the client is active or not.
- **Billable:** Whether the client is billable or not. If a client is marked as billable, it has a few more fields that define the default billing settings.
  - **Terms:** Payment terms used by default for invoices created for the client.
  - **Tax exempt:** Whether taxes are applied to invoices created for the client.
  - **Region:** Needed if the client is foreign.

Clients can be managed in the Projects view. Here's how to create, edit or delete a client.

- To create a new client, go to Business » New Client or click the **New Client** button from the toolbar.
- To edit a client, right-click it (control-click on Mac OS X) and select **Edit Client** from the popup menu.
- To delete a client, right-click it (control-click on Mac OS X) and select **Delete Client** from the popup menu.

You can also create a client whenever you're creating a project by using the New link next to the Client field.

3.3.2. Hiding inactive clients

If you don't work anymore with a client, you should not delete it. Instead edit it and mark it as inactive. Inactive clients are displayed in gray instead of black.

Then, click the small gears button below the clients list and select Active. Fanurio will display only active clients and hide inactive ones.

3.3.3. Billing clients

Clients can be billable or non-billable. If you don't need to create invoices for a client, you should edit it and mark it as non-billable.

- If a client is non-billable, it can only manage non-billable projects. You can't create invoices for non-billable clients.

  A non-billable client can always be changed to billable.

- If a client is billable, its projects are billable by default but you can mark them as non-billable if needed.

  A billable client cannot be changed to non-billable if it has been invoiced or if it has billable projects with at least one product. If a billable client is changed from billable to non-billable, all its projects, tasks, expenses and trips are changed to non-billable.

If a client is billable, you can specify default billing settings for its invoices and projects. These settings are used by default when a new invoice or project is created for that client.

Let's suppose you have two clients, client A and client B. Client A's work is quick and easy but client B's work is unclear and it takes many emails to determine what they are looking for. That's why for client A you don't need to round time while for client B you need to round it up to 15 minutes to cover for the overhead. Once you make these billing settings at client level, all the projects you create for the client will use the same settings. If you decide to change these settings at client level, existing projects will not be altered.

3.3.4. Billing foreign clients

Some businesses work with both domestic and foreign clients. Foreign clients may want invoices in their own currency with numbers displayed according to the rules of their country. For instance, some countries use comma as decimal separator while others use a dot. Dates may also be formatted differently. Some countries display the month first while others display the day of month.

Fanurio lets you specify if a client is domestic or foreign. If clients are foreign, you can also specify the client's country so that when invoices will be created for them, numbers, dates and currency will be formatted correctly.

Here's how to mark a client as foreign:

1. Go to the Projects view

2. Right-click the client and select Edit Client

3. Go to the Billing tab, check the "This is a foreign client..." option and then select a locale to indicate the country

Let's say that you are from Scotland and most of your clients are from the UK but you also have a few clients from the USA. When dealing with clients from UK, you will be invoicing in GBP while for the American clients the invoices will be in USD. By marking a client as being from the USA, all its invoices will show numbers, dates and currency formatted using American rules.

3.3.5. Importing clients from CSV

If you already have a list of clients, Fanurio can import them. Go to File » Import » Import Contacts from CSV ... to import one or more clients from a CSV (comma-separated value) file.

Most applications (especially e-mail applications) can export their data to such a format. A wizard will assist you along the way to make things easier.

3.3.6. Importing clients from Apple Contacts

Apple Contacts (known as Address Book before OS X 10.8 Mountain Lion) is widely used on OS X to manage contact details but since Fanurio runs on multiple platforms it can't rely on it to store contact details for its clients. To fix this problem, you can import contacts from Apple Contacts into Fanurio.

Fanurio allows you to import contacts stored in Apple Contacts as clients easily. Just go to File » Import » Import Contacts from Apple Contacts ... to choose the contacts you want to import. The import window will not display contacts that have been already imported to prevent you from importing the same contact twice.

Once you import a client in Fanurio, you can keep it in sync with Apple Contacts by ctrl-clicking it and selecting Update from Apple Contacts from the contextual menu. If a client is imported from Apple Contacts, the contextual menu has two more actions Show Contact in Apple Contacts and Edit Contact in Apple Contacts.

Please note that a client is not updated automatically in Fanurio when its contact is updated in Apple Contacts, it must be updated manually. All updates are one way only, from Apple Contacts to Fanurio. If you change the contact details of a client in Fanurio (unlikely if you use Apple Contacts) they won't be sent to Apple Contacts when you update it using Update from Apple Contacts from the contextual menu.

The following table shows how Fanurio maps Apple Contacts fields to its own fields. As a rule, it tries to get the information that it needs from work fields. If the work field is empty, it will try to use the home field but if that is empty too, it will use an empty text.

Table 3.3. Apple Contacts field mapping to Fanurio

Client Field (Fanurio)	Contact Field (Apple Contacts)
Name	Company name or contact display name
Attention	Contact display name
Address	Work street address or home street address
City	Work city name or home city name
State	Work state name or home state name
Zip	Work zip code or home zip code
Country	Work country name or home country name
Phone	Work phone number or home phone numner
Fax	Work fax number or home fax number
Mobile	Mobile phone number or iPhone number
Other	Main phone number
Email	Work email address or home email address
Website	Home page address

3.3.7. Exporting clients

Go to File » Export » Export Clients to export all the clients you recorded in Fanurio to CSV or Excel.

3.4. Working with projects

3.4.1. About projects

Fanurio uses projects to help you organize work, expenses, trips and products you might want to sell to your clients. Internal projects can be marked as non-billable.

Projects have a set of properties that you can use to manage them better:

- **Client:** The client for whom the project is created.
- **Name:** The name of the project.
- **Description:** A description of what the project is about.
- **Number [optional]:** A number that you can use to identify the project. Project numbers can be generated automatically.
- **Reference [optional]:** A reference number that you can use to link your project with an external project.
- **Tags [optional]:** A list of zero or more tags separated by commas. Tags can help you organize projects.
- **Start Date:** The date when the project has been started or will start (today by default).
- **Due Date [optional]:** The date when the project is expected to be finished.
- **Finished:** Whether the project is finished or not.
- **Finished Date:** The date when the project was finished. This date can only be set when a project is marked as finished.
- **Notes:** Additional notes.
- **Billable:** Whether the project is billable or not. If a project is marked as billable, it has a few more fields that define the default billing settings for tasks, expenses and trips.

Here's how to create, edit or delete a project.

- To create a project, go to Business » New Project or click the **New Project** button from the toolbar.
- To edit a project, right-click it (control-click on Mac OS X) and select **Edit Project** from the popup menu.
- To delete a project, right-click it (control-click on Mac OS X) and select **Delete Project** from the popup menu.

You can also create a project whenever creating a task, an expense or a trip by using the New link next to the Project field.

3.4.2. Choosing a projects view

Fanurio has two project views that allow you to see and manage your projects but only one view can be active at one moment.

- The **tree view** shows projects grouped by client in a tree and it's the default view. This view is recommended to users who don't have to manage many clients and projects.
- The **table view** shows projects in a table. This view is recommended to users who need to manage many projects. The projects table can show many project details and it allows projects to be searched.

The active project view can be changed from the menu by accessing View » View Projects as Tree or View » View Projects as Table.

3.4.2.1. The projects tree view

The tree view is used by default and it's recommended to users who don't have to manage many clients and projects. The tree view can be enabled from the menu using View » View Projects as Tree.

This is a rather simple view that shows projects grouped by client in a tree on the left while on the right it shows the contents of the selected project. To help you distinguish between active and inactive clients but also between finished and unfinished projects, Fanurio paints inactive clients and finished projects in gray.

The gears button below the projects tree allows you to configure which clients and projects are displayed in the projects tree. You can use the following filters:

- **client status (any, active, active with unfinished projects, not active)** and
- **project status (any, finished, not finished)**.

The "active with unfinished projects" filter shows only active clients that have ongoing (unfinished) projects. This filter can be very helpful if you have many active clients but only a few ongoing projects.

3.4.2.2. The projects table view

The table view is more complex than the tree view and it shows more details about both clients and projects. The table view can be enabled from the menu using View » View Projects as Table.

This view has three areas:

- The **clients list** is displayed on the left,
- The **projects table** is displayed on the right at the top and
- The **contents of the current project** is displayed on the right at the bottom.

When a client is selected in the clients list, its projects are displayed on the right in the projects table. Then when a project is selected in the projects table, it is displayed at the bottom.

The clients list has an entry for All Clients that lets you see all projects from all clients. The clients list displays the contact name under the client name and the number of ongoing (not finished) projects on the right.

The gears button below the clients list allows you to configure which clients are displayed in the clients list. You can filter clients by their status (any, active, active with unfinished projects, not active). The "active with unfinished projects" filter shows only active clients that have ongoing (unfinished) projects. This filter can be very helpful if you have many active clients but only a few ongoing projects.

The projects table is what makes this view very useful to people who need to see many details about their projects. The table can display all the fields of a project (name, number, start date, etc) and totals (time, distance, expenses, billable total, etc). Not all columns are visible by default so you may want to right-click the table header to choose the ones that make sense to you. See this section to learn how you can search and filter projects from the projects table.

3.4.3. Configuring optional fields

Because projects have many fields and because not all fields are relevant to all users, you can hide optional fields that you don't use. Optional project fields are fields that can be hidden in the project windows (New Project or Edit Project). Here's how you can configure them:

1. Create a new project (New Project) or edit an existing one (Edit Project).
2. Click the Configure button from the bottom-left corner.

3. Check the fields that you want to be visible and uncheck the ones that you want to hide.

4. Click Done. Fanurio will update the project window to show only selected fields.

For instance, one of the optional fields is Reference. This field only makes sense if projects recorded in Fanurio are linked to external projects that have their own number.

3.4.4. Numbering projects automatically

Automatic project numbering is disabled by default but can be enabled from Business » My Business Details+Projects+Project numbering.

To generate a project number automatically, Fanurio needs a counter and a number format. The number format can be something simple like counter on three or more digits (eg 001, 0001, 00001) or something more complex that involves prefixes, sufixes and dates.

Fanurio uses the Freemarker [http://freemarker.sourceforge.net/] syntax for the number format, the same syntax used for the templates. Here are a few number formats that assume the counter is 1 (one) and the year is 2020.

Table 3.4. Project number formats

Name	Number Format	Sample Number
Four-digit numbers To use more digits in your format, just add more zeros.	${counter?string("0000")}	0001
Fixed prefix format This format builds on the previous one by adding the **ABC-** prefix. Please note that it's important to put the prefix outside ${...}.	ABC-${counter? string("0000")}	ABC-0001
Date prefix format This format uses a variable prefix that changes with the year. The date prefix is generated by the expression **${.now?string("yy")}**. If you need to display the whole year use **${.now?string("yyyy")}** instead. The text "yyyy" represents the date pattern used to format the date which uses the Java date format syntax [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html].	${.now?string("yy")}- ${counter?string("0000")}	20-0001

3.4.5. Billing projects

Projects can be billable or non-billable. If you don't need to create invoices for a project (for instance if it's an internal project), you should edit it and mark it as non-billable.

- If a project is non-billable, it can only record non-billable elements (tasks, expenses or trips). You can't create invoices for non-billable projects.

A non-billable project can be changed to billable only if it belongs to a billable client.

- If a project is billable, its elements (tasks, expenses or trips) are billable by default but you can mark them as non-billable if needed. Also, billable projects can manage products that you may want to bill to your clients. Products can only be added to billable projects.

  A billable project cannot be changed to non-billable if it has been invoiced or if it has at least one product. If a billable project is changed from billable to non-billable, all its tasks, expenses and trips are changed to non-billable.

If a project is billable, you can specify default billing settings for its tasks, expenses and trips in the Billing section. These settings are used when a new element is created for that project.

For instance, you can specify a default hourly rate and a default rule for rounding time. Each time a new task is created, it will have the default rate and rounding specified at project level. If you decide to change these settings at project level, existing tasks will not be altered.

3.4.6. Duplicating a project

If you need to create a new project that's similar to an old one, you can simply create a copy of the old project. Just right-click the old project (control-click on Mac OS X) and select **Duplicate Project**.

Fanurio will create a new project with the same list of tasks and products. Expenses and trips are not copied because it's unlikely that two projects have the same expenses and trips.

3.4.7. Hiding finished projects

Knowing which projects are finished and which aren't can be of great help when you only want to deal with one kind of projects. If you want to focus on your current work, you can choose to see only unfinished projects. But if you want to go over past projects, you can choose to see only finished projects.

To change the finished state of a project, right-click (control-click on Mac OS X) a project and mark it as finished or unfinished. To hide finished projects:

- **Tree view:** click the small gears button below the projects tree and select Unfinished.
- **Table view:** use the Status filter above the projects table and select Unfinished.

Please note that you cannot invoice finished projects and you cannot record time (manually or using the timer) for its tasks.

3.4.8. Using tags to organize projects (table view only)

When creating or editing a project, you can associate one or more tags with it in the Tags field that have to be separated by commas (eg: tag1,tag2,tag3). Tags provide a simple way of recording additional information about a project like type or status to help you organize your projects. Please note that Tags is an optional field that you have to enable.

Once you have one or more projects with tags, you can then filter them in the table view. Just click the small arrow icon from the search field to tell it that you want to search by tags. Then type the name of the tag that you want to search for. The tree view doesn't show project tags.

3.4.9. Searching and filtering projects (table view only)

The projects table available in the table view has a set of filters at the top that allow you to choose which projects are visible. For instance, you can use the filters to see overdue projects that were started last year. Projects can be filtered by:

- **start date**,
- **status (any, finished, not finished, overdue)** and
- **billing status (any, billable, not billable)**.

Once you set these filters, the table footer will display totals for all visible projects. The table can also be filtered using a search field that can search by:

- **name**,
- **number**,
- **reference**,
- **description**,
- **tags** and
- **notes**.

3.4.10. Exporting projects in the iCalendar format

iCalendar [http://en.wikipedia.org/wiki/ICalendar] is the standard Internet format for exchanging calendar information. iCalendar is used and supported by a large number of products, including Google Calendar, Apple Calendar and Reminders (formerly iCal), Microsoft Outlook, Yahoo! Calendar and the Lightning extension for Mozilla Thunderbird.

Follow these steps to export a project as an iCalendar file (.ics):

1. Go to the Projects view.
2. Right-click the project that you want to export and select Export Project to iCalendar... from the popup menu.

The exported calendar contains to-dos (VTODO items created from tasks) and events (VEVENT items created from time entries). However, not all applications are able to import both types of items because not all of them are 100% compatible with the iCalendar format. Most applications can import events but only some of them can import to-dos.

Table 3.5. iCalendar compatibile products

Product	VEVENT (time entries)	VTODO (tasks)
Lightning extension for Mozilla Thunderbird	Yes	Yes
Apple Reminders	No	Yes
Apple Calendar	Yes	No
Google Calendar	Yes	No
Microsoft Outlook	Yes	No
Yahoo! Calendar	Yes	No

Here are two scenarios where exporting a project as an iCalendar file might be helpful.

Scenario 1: Share a project with a client using Apple Calendar and Reminders

OS X (10.8 and later) has two applications that can import iCalendar files: Calendar and Reminders (formerly iCal). Apple Calendar imports only events (time entries) while Apple Reminders imports only to-dos (tasks).

Once you import an iCalendar file in Apple Calendar [http://support.apple.com/kb/PH11524] you can visualize your time entries using the calendar views of the application. If you store your calendars in iCloud, you can also share them with a client who needs to know what you did.

When you import an iCalendar file in Apple Reminders [http://support.apple.com/kb/PH11835] you can see which tasks are done, which ones are in progress and when they are due. If you store your reminders list in iCloud, you can access it from iPhone or share it with a client.

Scenario 2: View your work using Google Calendar

Google Calendar can import events from iCalendar files [https://support.google.com/calendar/answer/37118]. This means that you can import in Google Calendar the time entries recorded on a project so you can have a better look at what you did. You can also share a calendar with other people who need to know what you did.

3.4.11. Creating projects reports

A projects report is an easy way to analyze the activity of one or more projects. For instance, if you need to see what you did last month, a project report will show you everything (tasks, time, expenses, trips and products) you recorded on your projects during that time.

Go to Reports » Projects Report to create a report for your projects. You can use the following filters to specify which project elements (tasks, expenses, trips and products) should be included in the report:

- **Projects** specifies the projects included in the report.
- **Date** specifies the date interval that is analyzed.
- **Billing** specifies whether the report should include billable or non-billable elements.
- **Invoiced** specifies whether the report should include invoiced or not invoiced elements.

Fanurio comes with one default template:

- **Projects by Client** shows the activity on the selected projects in the specified date interval.

If you need to create your own templates, please see this section to learn what placeholders you can use.

3.5. Working with tasks

3.5.1. About tasks

A task is an activity that needs to be accomplished within a defined period of time. Fanurio uses tasks to help you manage work for your projects. Here's what you can record with a task:

- **Project:** The project where you want to record the task. The project is mandatory as you can only record tasks on projects.
- **Name:** The name of the task.
- **Description:** A small description of what the task is about.
- **Reference [optional]:** A reference number that you can use if your tasks are imported from other applications.
- **Category [optional]:** A category that can help you organize your tasks.
- **Tags [optional]:** A list of zero or more tags separated by commas. Tags can also help you organize tasks and they are more flexible than categories.
- **Estimated Time [optional]:** How much time the task is expected to take.
- **Start Date:** The date when the task has been started or will start (today by default).
- **Due Date [optional]:** The date when the task is expected to be completed (the project due date by default if this field is visible, none otherwise).
- **Completed:** Whether the task is completed or not.
- **Completed Date:** The date when the task was completed.
- **Notes:** Additional notes.

If a project is billable, tasks have a few more fields. Tasks from non-billable projects don't show these fields:

- **Billable:** When checked, it indicates that the task is billable. Billable projects can have both billable and non-billable tasks.
- **Pricing:** Specifies how the task is billed: in units or in hours. If a task is billed in units then Billable Quantity must be entered manually but if a task is billed by the hour the Billable Quantity is determined automatically from the billable time entries.

  If a task is billed by the hour, you can also specify how time is rounded to determine the Billable Quantity.
- **Billable Quantity:** Indicates the quantity that can be billed for a task. Billable quantity can be discounted.
- **Price:** The price per unit of whatever the task bills. Price can be discounted.

There are several ways you can create a task:

1. Go to Business » New Task.
2. Go to the Tasks view and use the **New** button.
3. Open a project and go to the **Tasks** section. Then click the **New** button.
4. When you create a time entry or when you start the timer, you can also create a new task. Just click the **New** link button.

Tip

Although you can create as many tasks as you want, you shouldn't create one each time you are working on something. Instead, create tasks only for the major activities of the project. Then add one or more time entries to each task to keep track when you've done that kind of work.

Let's say that you need to keep track of how much time you spend on the phone with the client for a certain project. Instead of creating a task each time you talk to him or her, you should create a single task and then add time to that task for each call.

By doing this, your project will contain several manageable tasks that have complete time logs.

3.5.2. Configuring optional fields

Because tasks have many fields and because not all fields are relevant to all users, you can hide optional fields that you don't use. Optional task fields are fields that can be hidden in the task windows (New Task or Edit Task). Here's how you can configure them:

1. Create a new task (New Task) or edit an existing one (Edit Task).
2. Click the Configure button from the bottom-left corner.
3. Check the fields that you want to be visible and uncheck the ones that you want to hide.
4. Click Done. Fanurio will update the task window to show only selected fields.

For instance, one of the optional fields is Reference. This field only makes sense if tasks recorded in Fanurio are imported from other systems. The Reference field allows you to connect a task in Fanurio with a task-like object from other systems.

A similar field is Estimated Time. Some people need to work a limited amount of time on a task and in this case the Estimated Time field is very useful because it can be used to determine the remaining time. Other people don't have a time limit and they can spend as much time as needed on a task, in this case the Estimated Time field is not useful and can be hidden.

3.5.3. Billing tasks

Tasks can be used to manage work but they can also be used to bill it. A task can be used to bill three hours of programming or ten pages of Greek text translated to Latin, it's up to you how you bill your work. Tasks can be billed in units or hours.

Here's how to specify the billing settings for a task:

1. Check the Billable box.
2. Choose whether the task is billed in units or in hours.
   - If a task is billed in units then you will have to manually enter the quantity in the Billable Quantity field.
   - If a task is billed in hours then the billable quantity is calculated automatically from the recorded time. You can also use a time rounding rule to indicate how time is converted in hours. See the next section for more details.
3. Enter a unit of measure in the field next to the quantity field.

   By default, Fanurio uses **unit** for unit-based tasks and **hours** for hourly-rated tasks but you can enter any other unit of measure. For instance, if you have a task that bills a translation job you may want to use **words** as the unit of measure.

4. Enter a price in the Price field.

Another way to specify the billing settings for a task is to use a billable task category. If a task uses a billable task category, the task will use the billing settings of its category by default.

The next section explains in detail what happens when a task is invoiced.

3.5.4. Billing a task multiple times

Tasks can be billed once or multiple times. If a task is billed multiple times, at some point it has both invoiced work and not yet invoiced (or billable) work. To understand this better, let's consider the following example.

Example 1: Let's suppose that in January you started to work on a task called Consulting for one of your clients and this task is going to be finished in March. It's February 11 and you've already worked 10 hours in January and 2 hours this month. Since you are billing your clients each month, you need to bill her for the 10 hours from January. Once you create the invoice for January, the task Consulting will have both invoiced work (10 hours) and not invoiced work (2 hours).

In order to help you see all this information, Fanurio has several fields.

- **Billable Quantity:** Shows the billable quantity or the quantity that can be invoiced and hasn't been invoiced yet.

  For the Consulting task, Billable Quantity is 12 hours before the task is invoiced and 2 hours after it is invoiced. When everything is invoiced for a task, this field is 0 (zero).

- **Billed Quantity:** Shows the invoiced quantity or the sum of all invoiced quantities for that task.

  For the Consulting task, Billed Quantity is 0 hours before the task is invoiced and 10 hours after it is invoiced. When everything is invoiced for a task, this field shows the sum of all invoiced quantities.

- **Quantity:** Billed Quantity + Billable Quantity

- **Billable Total:** Price x Billable Quantity

  When everything is invoiced for a task, this field is 0 (zero) because Billable Quantity will then be 0.

- **Billed Total:** Shows the total value of everything that was invoiced for the task.

  When everything is invoiced for a task, this field is equal to Total because Billable Total will be 0.

- **Total:** Billed Total + Billable Total

Except for the **Billable Quantity**, **Billable Total** and **Total** fields, the other fields are not visible by default in the Tasks table. If you need to see them, just right click the header of one of the table columns and you can choose which columns are visible.

If you need to see the billing history of a task, all you have to do is edit it and go to the **Billing History** tab. This tab is displayed only when a task has been invoiced at least once. Also, if a task has been invoiced you can't change its Pricing attribute (you can't bill it once in hours and then in units) or make it non-billable.

The first example shows what happens when an hourly-rated task is billed. Here's a second example that shows what happens when a unit-based task is billed.

Example 2: Let's suppose you have a translation job and you need to translate 100 pages. In order to track time for this job and bill it, you create a task called Translation and mark it as billable. You set the pricing for the job in units and enter 100 in the Billable Quantity field. You also set a price and enter 'pages' in the unit of measure field. Please note that for unit-based tasks, the billable quantity

has to be entered manually as opposed to hourly-rated tasks for which billable quantity is calculated automatically. When you invoice this task, Fanurio will create a service item to bill all 100 pages and it will set Billable Quantity for the task to 0. If you edit the service item to bill less pages, for instance 40 pages, then Fanurio will update the Billable Quantity for the task to contain the remaining pages, in this case 60 pages.

3.5.5. Billing a task at different rates

When you invoice a task, Fanurio actually creates a service item with the same billing settings in order to invoice it but this doesn't mean you can't use more than one rate to bill a task. To understand this better, let's consider the following example.

Let's suppose you have a task called Consulting that must be billed using two different rates: regular hours are charged at $100/hour and overtime hours are charged at $150/hour. Because tasks can only be configured with one rate, you enter $100 in the Price field. At the end of the month, the task has 5 regular hours and 2 overtime hours. Here's what you need to do to bill the task in this case:

1. Create an invoice.

2. Add the task with all its time (7 hours) to the invoice. Fanurio will create a service item that bills 7 hours at $100/hour.

3. Edit the service item and remove the 2 hours of overtime.

4. Invoice the task again for the remaining 2 hours. Fanurio will create a service item that bills 2 hours at $100/hour.

5. Edit the second service item and change its price to $150.

Now the invoice has two service items that bill the same task for both regular hours and overtime hours.

3.5.6. Marking invoiced tasks as completed

Tasks can be marked as completed either manually by editing them or automatically when they are invoiced.

Go to Business » My Business Details+Projects+Tasks to configure whether Fanurio marks invoiced tasks as completed. You can choose one of the following values:

- **Automatically:** If a task is invoiced and its remaining billable quantity is zero (in other words, there's nothing else to bill) then **Fanurio will automatically mark it as completed**. This setting makes sense if you **bill your tasks only once**. Let's consider the following example.

  You start working on a task. When you're done, you create an invoice to bill it and Fanurio automatically marks it as completed.

- **Prompt me:** If a task is invoiced and its remaining billable quantity is zero (in other words, there's nothing else to bill) then **Fanurio will ask if you want to mark it as completed**. This setting makes sense if you **bill your tasks more than once**. Let's consider the following example.

  In January, you start working on a task that will end in February. At the end of the first month, you bill this task and Fanurio asks you if you want to close it but because it's not done yet, you don't mark it as completed. The next month, the same thing happens but this time you mark it as completed.

- **Never:** Nothing happens when a task is invoiced. It's up to you to mark tasks as completed when they're done. This is the default value.

Fanurio uses the following algorithm to calculate the completed date when configured to use the **Automatically** or **Prompt me** modes:

- If the task has at least one time entry, it will use the end date of the last time entry. Time entries represent working sessions for a particular task so it makes sense to use the date of the last time entry to determine when the task ended. If there are no new time entries then the date of the last time entry is the date when the task ended.
- If a task doesn't have any time entries then it will use the end date of the billing period.
- If the billing period is undefined (all dates) then the completed date is set to **now**.

3.5.7. Rounding time for billable tasks

Unlike tasks billed in units where the billable quantity must be entered manually by the user, the billable quantity of hourly-rated tasks is calculated automatically by converting the total billable time to hours.

Let's suppose you have a billable task with two uninvoiced time entries (t1 = 16 minutes and t2 = 32 minutes). Here's how its billable quantity is calculated in three different cases:

a. Time rounding is not enabled

   Quantity = 0:16 + 0:32 = 0:48 = 0.8 hours

b. Time is rounded up to 15 minutes for *the sum of all the time entries*

   Quantity = **round(**0:16 + 0:32**)** = **round(**0:48**)** = 1:00 = 1.0 hours

c. Time is rounded up to 15 minutes for *each time entry*

   Quantity = **round(**0:16**)** + **round(**0:32**)** = 0:30 + 0:45 = 1:15 = 1.25 hours

When creating a time rounding rule, you can also specify a minimum amount of time to bill. By default, the minimum field is set to 0 but you can change it to create rules like "round time to the nearest 15 minutes interval but bill at least 30 minutes".

Note: Fanurio uses the hour format (1:15) when dealing with time and the decimal format (1.25) when dealing with billable quantities.

The easiest way to round time for a task is to edit it and click the link next to hourly pricing option. However, if you use the same rounding rule for all the tasks of a project, you should set that as a default billing setting for that project.

If you use the same settings for all your projects then you can define a default time rounding rule under Business » My Business Details+Projects+Tasks that will be applied to all new projects you create after that. It will not alter the settings of existing projects or tasks. If you want the new rule to also be applied to existing projects and tasks, you will have to change them by hand.

3.5.8. Understanding billable time precision

The precision method used to calculate billable time is a business setting that rarely (maybe never) needs to be changed. The current version uses exact precision by default but before version 3.1 the default was set to two-decimal precision. If you need to change this setting, go to Business » My Business Details+Projects+Tasks.

In order to calculate the value of billable time (e.g. 20 minutes), Fanurio converts it to a decimal number. To do this it can either use exact precision (0.333...3, unlimited number of decimals) or two-decimal precision (0.33, two decimals).

- When using **exact precision**, the total is calculated accurately down to the penny but the number of hours may not always be a two-decimal number (0.333...3). If the number of hours has more than two decimals (0.333...3) then it may not be practical to print it on invoices and if the number is

rounded to two decimals (0.33) it will not be accurate. That's why when using exact precision, time should be printed in hour format (00:20) and not in decimal format (0.333...3).

- When using **two-decimal precision**, the number of hours is guaranteed to be a two-decimal number all the time (0.33). This type of precision is recommended if you need to print time on your invoices in decimal format. Since the number of hours is rounded to two decimals, the total will be larger or smaller depending on whether rounding is done up or down.

Let's see how the two precision methods influence the value of billable time for a task billed at a rate of $60/hour.

Table 3.6. Differences between exact precision and two-decimal precision when calculating billable time

Example	Rate	Billable Time	Hours / Total (exact)	Hours / Total (two-decimal)	Difference (exact - two-decimal)
1	$60/hour	00:15	0.25 / $15.00	0.25 / $15.00	$0.00
2	**$60/hour**	**00:20**	**0.333...3 / $20.00**	**0.33 / $19.80**	**$0.20**
3	$60/hour	00:30	0.50 / $30.00	0.50 / $30.00	$0.00
4	**$60/hour**	**00:40**	**0.666...6 / $40.00**	**0.67 / $40.20**	**-$0.20**

Examples 1 and 3 show that if billable time converts to a number of hours with two decimals (0.25, 0.50), the two methods yield the same results so it's irrelvant which one is used. Examples 2 and 4 show the rounding differences between the two methods.

Choosing between the two methods may sound complicated and if it does, just go with the default option (exact precision). However, the best solution is to make this choice irrelevant by rounding time to 6, 15 or 30 minutes so that billable time always converts to a number of hours with two decimals.

3.5.9. Planning work with tasks

Although Fanurio has tasks like other project management applications, it doesn't have features like task dependencies, task hierarchies or Gantt charts. This helps us keep things simple and manageable. We are also counting on the fact that our users work on less complex projects that can be managed without such features.

Tasks have the following fields that can help you plan your work:

- **Start Date:** The date when the task has been started or will start.
- **Due Date:** The date when the task is expected to be completed.
- **Completed:** Whether the task is completed or not.
- **Completed Date:** The date when the task was completed.

Except for the **Start Date** field, the other fields are not visible by default in the Tasks table. We're assuming most projects created in Fanurio are simple projects that don't actually require any planning. If you need to see them, just right click the header of one of the table columns and you can choose which columns are visible.

The Tasks table has two filters that can help you plan your work better. The Status filter allows you to see completed, not completed or overdue tasks while the Due Date filter allows you to see tasks by their due date. For instance, you could use it to see all tasks that are due this week.

3.5.10. Tracking progress on tasks

Fanurio can track progress only if you specify a time estimate. The estimated time and the actual recorded time (the sum of all time entries) are used to calculate the progress and the remaining time.

Progress-related columns like **Estimated Time**, **Remaining Time** and **Progress** are not visible by default in the Tasks table. If you need to see them, just right click the header of one of the table columns and you can choose which columns are visible.

If the **Estimated Time** and **Remaining Time** columns are visible, Fanurio also shows their totals at the bottom of the Tasks table.

3.5.11. Using categories to organize tasks

Tasks can have an optional category (eg Design, Tech Support, etc) that can be used to organize them.

The list of task categories can be managed from Business » My Business Details+Projects+Tasks. Task categories can also be specified when a task is created or edited using the **New** link button.

Tasks can be searched or filtered by category in the Tasks view. Just click the small arrow icon from the search field to tell it that you want to search by category. Then type the name of the category that you want to search for.

3.5.12. Using tags to organize tasks

When creating or editing a task, you can associate one or more tags with it in the Tags field that have to be separated by commas (eg: tag1,tag2,tag3). Tags provide a simple way of recording additional information about a task like milestone name.

If you need to tag multiple tasks, go to the Tasks view and select them. Then right-click to display the contextual menu and select Edit Tags.

Once you have one or more tasks with tags, you can then filter them in the Tasks view. Just click the small arrow icon from the search field to tell it that you want to search by tags. Then type the name of the tag that you want to search for.

3.5.13. Searching and filtering tasks

To learn more about the tasks you record in Fanurio, go to the Tasks view and use the filters above the tasks table. The table will display only those tasks that match the selected filters. For instance, you could use these filters to see all active (not completed) or overdue tasks.

Tasks can be filtered by:

- **client status (any, active, not active)**,
- **project status (any, finished, not finished)**,
- **status (any, completed, not completed, overdue)**,
- **due date**,
- **billing status (any, billable, not billable)** and
- **invoiced status (any, invoiced, not fully invoiced)**.

  Please note that since tasks can be billed more than once, tasks displayed by the "invoiced" and "not fully invoiced" filters may overlap. For instance a task that takes two months and was billed

once, will be both "invoiced" (because it was billed once) and "not fully invoiced" (because it has to be billed for the second month).

- "Invoiced" will show all tasks that were invoiced but this doesn't mean some of them cannot be invoiced again.
- "Not fully invoiced" will show all tasks that may be invoiced but this doesn't mean they weren't already invoiced. By "may be invoiced" we mean new billable tasks or billable tasks with a billable quantity different from zero (i.e. they have something to bill).

Once you set these filters, the table footer will display totals for all visible tasks. The table can also be filtered using a search field that can search by:

- **name**,
- **description**,
- **category**,
- **tags** and
- **client;project**.

The **client;project** option allows you to filter tasks by client, or project name.

If you have a client called **Aristotle** with a project called **Rhetoric**, just type **Aristotle;Rhetoric** and it will display this exact project. If you type **Aristotle** it will display all tasks for this client. The semicolon is very important as it helps Fanurio distinguish between fields.

3.5.14. Exporting tasks

Fanurio can export tasks to CSV and Excel. To export them, you can either:

- Go to File » Export » Export Tasks to export all the tasks you recorded in Fanurio or
- Go to the Tasks view and use the filters above the table to specify the tasks you want to export. Then right-click the table to display the contextual menu and select **Export Tasks** to export the visible tasks.

3.5.15. Creating tasks reports

The previous section explains how you can filter or search the list of tasks so you can get a quick insight about your activities. However, if you need to use this information outside Fanurio you can either export the list of tasks as a CSV or Excel file or create a tasks report. Tasks reports are more flexible because they allow you to use a template to format data.

Tasks reports also allow you to see time and money details for a specific period of time, something that is not possible in the Tasks view. For instance, if a task starts in January and ends in March, a tasks report can show what happened on that task in terms of time and money in January and hide information from the other months.

Go to Reports » Tasks Report to create a report for your tasks. You can use the following filters to specify which tasks should be included in the report:

- **Projects** specifies the projects included in the report.
- **Date** specifies the date interval when tasks where active. A task is active since it starts (Start Date) until it ends (Completed Date).
- **Billing** specifies whether the report should include billable or non-billable tasks.

- **Invoiced** specifies whether the report should include invoiced or not fully invoiced tasks. See this section for more details on how this filter works.

Fanurio comes with the following templates:

- **Tasks by Client and Project** shows the time and money earned by each task from the selected projects in the specified date interval.
- **Tasks Progress by Client and Project** displays the estimated time, the actual recorded time and the progress for the tasks included in the report.

If you need to create your own templates, please see this section to learn what placeholders you can use.

Note: Tasks reports evaluate how much your tasks are worth. They don't tell how much money you've made from invoicing your clients. Sometimes there can be a difference between the two. Suppose you have a project worth $1100 and you invoice it with a $100 discount for $1000. The report will tell you the project is worth $1100 but that's not how much you've asked for it. If you need to analyze your invoices, then you need to use sales reports.

3.6. Working with time entries

3.6.1. About time entries

A time entry represents the time spent doing something for a task. It is defined by a start time, elapsed time, pause time and a finish time. If you work from 09:00 AM until 11:00 AM and you take two 15 minutes breaks, the elapsed time will be 1:30 hours and the pause time will be 0:30 hours.

Time entries have the following properties:

- **Project:** The project where you want to record the time.
- **Task:** The task where you want to record the time. The task is mandatory as you can only record time on tasks.
- **Date:** The date when the time entry is recorded.
- **Start:** The time when the activity started.
- **Time:** Work time.
- **Pause:** Break time.
- **Finish:** The time when the activity ended. Finish = Start + Time + Pause.
- **Units [optional]:** The number of units done if the activity is quantifiable. For instance, translators may want to record the number of translated words to correlate them to the actual time spent to translate them. This field is optional because most people don't need to track time for quantifiable activities. See this section for more details.
- **Description:** A small description of what the time entry is about.
- **Tags [optional]:** A list of zero or more tags separated by commas. Tags can also help you organize time entries. See this section for more details.

Depending on how you want to enter time, Fanurio lets you choose how to do it.

- **Relatively to start**: You enter the start time and the elapsed time and then Fanurio calculates the end time by adding the elapsed time to the start time.
- **Relatively to finish**: You enter the elapsed time and the finish time and then Fanurio calculates the start time by subtracting the elapsed time from the end time.
- **Both**: You enter the start and finish times and then Fanurio calculates the elapsed time by subtracting the start time from the finish time.

If a task is billable, its time entries are billable by default but they can be marked as non-billable.

- **Billable**: When checked, it indicates that the time entry is billable. Billable tasks can have both billable and non-billable time entries.

3.6.2. Configuring optional fields

Because time entries have many fields and because not all fields are relevant to all users, you can hide optional fields that you don't use. Optional time entry fields are fields that can be hidden in the time entry windows (New Time or Edit Time). Here's how you can configure them:

1. Create a new time entry (New Time) or edit an existing one (Edit Time).
2. Click the Configure button from the bottom-left corner.

3. Check the fields that you want to be visible and uncheck the ones that you want to hide.

4. Click Done. Fanurio will update the time entry window to show only selected fields.

For instance, one of the optional fields is Tags. This field only makes sense if you need to organize your time entries using tags.

3.6.3. Recording time manually

Whether you forget to record time or you are not in the front of the computer most of the day, you can always record time manually. Please note that you can only add time to a task and not directly to a client or a project.

There several ways you can add time to a task:

1. Go to Business » New Time to add time to the currently selected task. If no task is selected, you will have to specify one. This is probably the fastest way to enter time in Fanurio.

2. Go to the Timesheet view and use the **New** button. When you use this method and the date filter is set to a specific date, time is added by default to that date. This method is very useful if you need to enter time for previous dates as it saves you from specifying the date for each time entry.

3. Open a project and select the task where you want to add time. Then right-click it and select **New Time** from the contextual menu.

4. Edit a task and go to the Time section to add time to that task.

Time fields accept time in both hour and decimal format. Here are a few input examples:

- 2:30 - enter the number of hours and minutes.
- :30 - enter the number of minutes.
- 2.5 or 2,5 - enter the number of hours.
- .5 or ,5 - enter the number of hours.
- 2 - enter the number of hours.

The major drawback of manual time recording is that you need to remember the times and duration of each task. That's why a better method to track time for your activities is to use a timer.

3.6.4. Recording time with timers

Tracking time is a lot easier when you have a timer. Unlike manual time recording, a timer will save you the trouble of remembering the exact time when you started, stopped or paused a task.

Fanurio can manage multiple timers but only one can be active, all the other timers are paused. When a new timer is started, the active one is paused and the new one becomes active. This feature is very useful if you start working on something (task A) and then you get a call from a client that forces you to interrupt what you were doing and work on something else (task B). In this case, you start a timer for task A and then when the client calls, you pause the timer (interrupt task A) and you start a new one for task B. When task B is over, you stop its timer and save the time. Then you can resume the timer for task A.

Since a timer is an important tool, Fanurio provides several ways to access it. You can access the timer from:

- the Timer menu,
- the toolbar,

- the tray icon menu,
- the iTunes-like mini timer,
- the taskbar button's thumbnail toolbar on Windows 7 or
- using global hotkeys on Windows and Linux.

The following actions can be used to control the timers from one of the places mentioned above:

- **Start New Timer**: This action starts a new timer. If a project is open and a task is selected, Fanurio will start a timer for that task.

  If a timer is already active (paused or running), that timer will be paused and the new one will become active.

- **Start New Timer...**: This action lets you start the timer for a specified task and attach a description to it. When the timer will be stopped, Fanurio will add time to this task with the specified description. If a project is open and a task is selected, Fanurio will suggest to start the timer for that task.

  If a timer is running and a new one is started in the past, the active timer will be paused in the past and the new one will be started from that time. Let's suppose you are working on task A and a client calls. In the middle of the call, you realize that you want to time the conversation so you start a new timer but since you already started the phone conversation 10 minutes ago, you want to start the timer in the past, 10 minutes ago. Doing so will pause the timer for task A 10 minutes ago.

- **Start New Timer >**: This action lets you start a new timer for one of the recent tasks.

- **Timers**: If you started at least one timer, this action shows all the timers. When you select one of the timers, it automatically becomes active and it's resumed. Recent timers are at the top of the list.

  The Timers action from the toolbar also shows the number of timers using a badge icon so you can easily see how many timers you have started.

- **Pause / Resume Timer**: This action pauses or resumes the active timer.

- **Pause / Resume Timer**...: This action pauses or resumes the active timer in the past.

- **Stop Timer**: This action stops the active timer and opens a New Time window to save the recorded time.

- **Edit Timer**: This action lets you change the task and the description of the active timer. It also shows the time when it was started, the elapsed time and the pause time.

- **Discard Time...**: This action discards time from the active timer.

  For instance if you started a timer one hour ago but at some point you took a 15 minutes break, you can use this action to discard this time and keep the timer running. Once you do that, the timer will show it's running for 45 minutes instead of one hour.

- **Transfer Time...**: This action discards time from the active timer and saves it to a task.

  For instance if you started a timer one hour ago but at some point you did something else for 15 minutes, you can use this action to discard this time and to add it to the task you've been working on. Once you do that, the timer will show it's running for 45 minutes instead of one hour.

The state of the timers is saved regularly on disk just in case there's a power outage and the application is terminated prematurely. If this happens, the timers will be restored the next time Fanurio will be restarted.

Important: If you are not recording time in seconds, Fanurio will round the time recorded by a timer to the nearest minute when it's stopped. For instance:

- If the timer shows **01:29:15** (1 hour, 29 minutes and 15 seconds) then this time will be rounded down to **01:29:00** (1 hour, 29 minutes).
- If the timer shows **01:29:35** (1 hour, 29 minutes and 35 seconds) then this time will be rounded up to **01:30:00** (1 hour, 30 minutes).

Although a timer is a major improvement over manual time recording, it's worthless if you don't remember to use it. When you have lots of work on your head, paying extra attention to a timer is the last thing you want to do. That's why Fanurio has smart timing, an even better method to track time.

3.6.5. Using reminders to control the active timer (smart timing)

A timer is useless if you don't remember to use it. To solve this problem, Fanurio has a few reminders to help you start, resume or stop a timer. Instead of relying on your memory and attention to control the timer, you can use these reminders. We call this feature smart timing.

Smart timing is not enabled by default. You have to enable it from:

- Tools » Options on Windows
- Fanurio » Preferences on Mac OS X
- Edit » Preferences on Linux

Smart timing is how Fanurio figures out what you are doing in order to record time accurately. It uses idle time detection and a set of reminders to do that.

1. **Reminders**: If you are working on the computer, Fanurio doesn't know what you are doing but if you enable smart timing, it will try to learn that from you. Fanurio can ask you repeatedly (you can specify the frequency) what you want to do with the timer.
   - If no timer is running, it will ask you if you want to start one or if you want to do it later.
   - If the active timer is paused, it will ask you if you want to resume it or if you want to do it later.
   - If the active timer is running, it will ask you if you want to stop it or if you want to leave it running.
2. **Idle time detection**: One thing Fanurio can figure out without asking is if you leave the computer while the timer is running. In this case it will ask you to do something with the time you've been away.

As we said, smart timing is about figuring out what the user is doing. If the timer is stopped, Fanurio will try to learn whether it should be started or not. The same happens when the timer is paused or if it's running.

To understand how smart timing works, let's see the following examples.

Example 3.1. Reminder that the timer is running

Let's assume the reminder is set to 10 minutes and the timer is already running for 32 minutes. Since I started the timer, Fanurio asked me three times (at 10, 20 and 30 minutes) if I want to stop it or if I want to keep it running. Each time I just pressed ESC to cancel the reminder dialog and to keep the timer running.

The fourth time when it asks me (at 40 minutes), I realize I finished working on my task and I choose to stop the timer. I will assign 40 minutes to the task I was working on.

Tip

When the reminder dialog is displayed, you can postpone the decision by pressing ESC.

Example 3.2. Idle time notification

Let's assume idle time notification is set to 10 minutes. That means Fanurio will notify me if I'm away from the computer for more than 10 minutes.

I start the timer and after 40 minutes I leave the computer for a coffee break. When I return after 15 minutes, I see a notification dialog where Fanurio asks me what to do with these 15 minutes. I decide to discard them. The timer will continue to run and to show it's been started 40 minutes ago instead of 55. The 15 minutes I've been away are considered pause time.

Besides **Discard**, the idle notification dialog has two other options: **Transfer** and **Keep**. Use **Transfer** if you've worked on something else in the meanwhile (a client was on the phone for instance) and **Keep** if you've been working on the same thing but you didn't touch the computer.

Tip

To tell Fanurio that you want to keep the time and leave the timer running, you can also press ESC or ENTER.

Best practices

- Adjust the idle time interval if you take shorter breaks. It is set to 15 minutes by default.
- Make sure the reminders are not too frequent as they may become annoying.

  If you usually do long tasks, you don't want to be reminded each 10 minutes that the timer is running. You could set the reminder to 20 or 30 minutes.
- Use only those reminders that you find useful. You don't have to enable all the reminders.

3.6.6. Recording time to the second

By default, Fanurio tracks time in minutes but it can be configured to track time in seconds if you need that kind of precision. To enable this feature, check the "display time with seconds" box in the settings window.

- Tools » Options+Locale on Windows
- Fanurio » Preferences+Locale on Mac OS X
- Edit » Preferences+Locale on Linux

This setting configures Fanurio to display seconds for the selected time format. For instance, if Fanurio is configured to use the 12-hour time format, it displays 09:30:10 AM when seconds are enabled and 09:30 AM when seconds are not enabled.

When the time format is configured to display seconds, Fanurio also displays durations in seconds. For instance, if a time entry starts at 09:30:10 AM and ends at 09:50:40 AM then the duration is 00:20:30 (20 minutes and 30 seconds). By durations we mean durations visible in the user interface (tables, input fields, etc), invoices and reports except for QuickBooks reports (*.iif) because they accept only durations in hours and minutes.

Important: Once you enable this feature, you shouldn't disable it because existing time entries will lose their seconds when edited.

3.6.7. Recording quantifiable activities

Time entries allow you to record the time you spend doing *something*. For some people, that *something* is a quantifiable activity that needs to record both the time and the quantity so they can analyze their productivity.

For instance, when translating a text from Latin to Greek in 1.5 hours some people may also want to record how many words it has. If the text is 600 words long, Fanurio can show that they've translated 400 words / hour and that it took them 9 seconds to translate one word. These stats allow users to see how they perform over time and make better estimates in the future.

To enable this feature, please follow these steps:

1. Go to the Timesheet view.
2. Double-click an existing time entry to edit it (or click to create a new one). This will display the Edit Time window.
3. Click the Configure button from the bottom-left corner to show the list of optional fields.
4. Check the Units box to enable it and click Done. This will display a new field called Units in the Edit Time window.
5. Click Done to close the Edit Time window.

Then you also need to configure the table to display the new field:

1. Right-click any column name from the Timesheet table. This will display a contextual menu with all the table columns.
2. Click Select Columns from the contextual menu. This allows you to configure which columns are visible.
3. Check the columns Units, Units/Hour and Time/Unit.
4. Click Done to make the columns visible.

The table footer displays the total for the Units column and averages for the Units/Hour and Time/Unit columns. By using the Date filter available above the Timesheet table, you can easily see productivity stats (Units/Hour and Time/Unit) for the day, week or month.

Important: Once you enable this feature, the Start New Timer window will also display a Units field.

3.6.8. Billing time

To bill time, you need to add one or more billable time entries to a billable task. The task defines the pricing for all its time entries.

Time entries added to a billable task are billable by default but you can change them to non-billable if you don't want to bill them.

A time entry can track both the actual recorded time and the billing time. This allows you to control the amount of time that is billed to the customer without losing the actual recorded time. It also allows you to create accurate performance reports using the actual recorded time and detailed customer reports using the billing time.

A time entry that took 22 minutes to complete could be billed as 0 minutes, 22 minutes or 30 minutes, it's up to you to decide how much time you want to bill. However, deciding how much billing time to enter for each time entry can be a pain. That's why each time entry has a default billing time that is calculated using its task settings.

If you're billing by the hour, setting a time rounding rule for the task is the best solution to calculate billing time automatically. If you want to change the billing time manually, you can override the default value and enter the actual amount of time you want to bill.

Here's how the default billing time is calculated:

- **For time entries added to tasks billed in units: default billing time = recorded time**

  recorded time = 12 minutes => default billing time = 12 minutes

- **For time entries added to tasks billed in hours without rounding: default billing time = recorded time**

  recorded time = 12 minutes => default billing time = 12 minutes

- **For time entries added to tasks billed in hours with rounding for each time entry: default billing time = rounded recorded time**

  recorded time = 12 minutes (each rounded up to 15 minutes) => default billing time = 15 minutes

- **For time entries added to tasks billed in hours with rounding for the sum of all time entries: default billing time = a fraction of the rounded task time**

  recorded times = [12 minutes, 24 minutes] (sum rounded up to 30 minutes) => default billing times = [20 minutes, 40 minutes]

  For this type of rounding, the default billing time can't be changed by users because time entries are handled as a group and not individually.

3.6.9. Importing time from CSV

Whether you want to switch to Fanurio from some other application or import time from a mobile app (iOS, Android, Blackberry), Fanurio has an import wizard that can import time from **any** CSV file. Just go to File » Import » Import Timesheet... and select CSV for source.

When you select a CSV file for import, Fanurio will try to detect its format so it can read its time records. If a format is not found, it will ask you to create one so that it knows how to map data from the CSV file to its own fields. Fanurio has predefined formats for the following applications:

- iTimeSheet [http://itimesheet.free.fr] [iPhone]
- TimeLogger [http://www.appideas.com/node/4] [iPhone]
- HoursTracker [http://hourstrackerapp.com/] [iPhone]
- Time Tracker [http://code.google.com/p/time-tracker-mac/] [OS X]
- Toggl [http://www.toggl.com] [web]
- Freckle [http://letsfreckle.com/] [web]
- BizTrackIt [http://www.shrunkenhead.biz/biztrackit.html] [BlackBerry]

A CSV format tells Fanurio how to map columns from the CSV file to one of the following fields:

- **Client:** name of the client.
- **Project:** name of the project.
- **Task:** name of the task where time is saved.
- **Date:** date when the time entry was recorded (eg 2012-07-22).
- **Start:** start time when the time entry was recorded (eg 17:29:59:999 which is almost 5:00 PM).
- **Time:** total recorded time (eg 02:30:00 which means 2.5 hours).

- **Description:** description associated with the time entry.
- **Tags:** tags associated with the time entry (must be a list of tags separated by commas).

To map columns from a CSV file to Fanurio, it usually means that you have to associate a column from the CSV file with a field in Fanurio. For instance, if you have a CSV file that has a column called **Customer** then you may want to map it to the **Client** field. Some fields like the **Time** field, need additional information. For instance, if your CSV file has a column called **Elapsed time (minutes)** then you need to specify this column for the **Time** field but you also need to select Minutes in the format field so that Fanurio knows how to interpret it.

Important: If you have problems creating a format for your CSV file, contact us and we'll create it for you.

Once you have a format that Fanurio can use to parse the file, you can select the time entries that you want to import. Before importing them, you can adjust the client, project and task names by double-clicking their table cells. You can also tag all selected time entries by using the **Tag** button.

If the clients, projects and tasks do not exist, they will be created automatically by Fanurio.

The following examples show to handle various mapping scenarios. A CSV format is actually a set of mapping expressions, one for each field. Fanurio uses Freemarker for these expressions, the same language that it uses for invoice templates.

Example 3.3. Mapping a field to a column

In order to connect a field from Fanurio to a column from the CSV file, you must select the column from the drop down box associated with that field. That box also shows the value from the first row for that column so that you can know what you are importing.

If you have a column called **Customer** then the expression that maps this column to the **Client** field is displayed below.

```
${column("Customer")}
```

Example 3.4. Mapping a field to a fixed text

The CSV file may not have columns for each field. For instance, some applications save time directly on the project instead of saving it on tasks like Fanurio does. In that case, you can't map a certain column to the **Task** field. If your file doesn't have a column that can be mapped to the **Task** field then you should enter some text in the text box of that field (eg Activity, Task, Job or any other name). All the time entries will be imported to tasks with that name.

The **Task** field accepts any text but other fields accept only text formatted in a certain way.

- **Client:** any text. All the time entries will be imported to the same client.
- **Project:** any text. All the projects will have the same name.
- **Task:** any text. All the tasks will have the same name.
- **Date:** text formatted as yyyy-MM-dd (eg 2012-07-22).
- **Start:** text formatted as hh:mm:ss:SSS (eg 17:29:59:999).
- **Time:** text formatted as hh:mm:ss (eg 02:30:00).
- **Description:** any text
- **Tags:** a list of comma-separated tags

Example 3.5. Not mapping a field to a column

The **Description** and **Tags** fields may not have a column where they can be mapped. In that case, you shouldn't select a column for them and leave their text box empty.

Example 3.6. Splitting columns

Some applications only have projects and the only way you can record both the client name and the project name in the same field is to use some separator. For instance, if you recorded your time on a project called **Aristotle-Rhetoric** that contains both the client name and the project name, then you can split it and extract the first part for the client name and the second part for the project name.

If you have a column called **Job** that contains both the client name and the project name separated by a minus (-) then the expression for the **Client** field is displayed below.

```
${column("Job")?split("-")[0]}
```

The expression for the **Project** field is displayed below.

```
${column("Job")?split("-")[1]}
```

Example 3.7. Mapping the date and start fields

The date, start and time fields are the only fields that require additional formatting. So, when mapping a column to these fields, you need to pay attention to their format.

For instance, if the CSV file contains a **Date** column that has values like 4/28/2010 then the expression that maps this column to the **Date** field is displayed below.

```
${column("Date")?date("M/dd/yyyy")}
```

For instance, if the CSV file contains a **Time In** column that has values like 20:21:00 then the expression that maps this column to the **Start** field is displayed below.

```
${column("TimeIn")?time("hh:mm:ss")}
```

For more details on date and time patterns, see this page [http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html].

3.6.10. Importing time from iCalendar

Fanurio allows users who prefer to organize their work with a calendar application to import calendar events as time entries. The recommended way to do this is to create a separate calendar for each project so you can export them as iCalendar files (.ics).

iCalendar [http://en.wikipedia.org/wiki/ICalendar] is the standard Internet format for exchanging calendar information. iCalendar is used and supported by a large number of products, including Google Calendar, Apple Calendar (formerly iCal), Microsoft Outlook, Yahoo! Calendar and the Lightning extension for Mozilla Thunderbird.

Follow these steps to import the events of an iCalendar file (.ics) as time entries in a project:

1. Go to the Projects view.
2. Right-click the project where you want to import the events of an iCalendar file and select Import Time from iCalendar... from the popup menu.
3. Select an iCalendar file (.ics).
4. Make sure the import period is set correctly. By default, the import period is set to [project start date ... project due date].

5. Click Import to import the events as time entries.

Fanurio will import new events and update existing ones under some conditions:

- Invoiced time entries will not be updated if they are reimported to prevent invoices from being altered.

  For instance, at some point you import a calendar and invoice the imported time entries. After that, you go back to your calendar application and edit some of the events that were already imported and you add a few more. When you will import the calendar again, the changes that you made to the invoiced time entries will not be imported.

- Only newer time entries will be updated (older versions of an event are skipped).

  If you import the events of a calendar regularly (for instance at the end of each week), only new events are imported and events that you changed in the calendar after the last import.

Importing recurring events

When importing time from an iCalendar file, Fanurio allows you to specify an import period that limits the events that are imported.

For recurring events, if the period is not bounded (for instance it is set to All Dates or it is set to import everything after a certain date) Fanurio will use a bounded period nevertheless. That's because a recurring event may occur indefinitely and Fanurio can't convert it to an infinite number of time entries. If the period is not bounded, Fanurio will import all occurrences of a repeating event from when it was started until the time when the import is performed.

If your calendar file contains recurring events it's probably best that you import them after they occurred otherwise you may import future events that may be cancelled. In this case, the import period should be a past period like last week or last month.

Reimporting recurring events with a different occurrence time (unlikely)

If you reimport in Fanurio a recurring event that has a changed start time, its new occurrences will be imported along the old ones. Although it's unlikely that you import a recurring event in Fanurio and then edit all its occurrences in your calendar application, we're documenting this case here just to let you know how Fanurio deals with it.

Let's assume you have a recurring event for "Meeting from 9:30 to 10:30 each Monday" and you import some of its occurrences in Fanurio. For some unknown reason (that's why we say this is unlikely) you need to change all occurrences of this event to "Meeting from 9:00 to 10:00 each Monday". The next time you will import this event in Fanurio you will have time entries for both the old event and the new event.

If you need to edit an event in a calendar, make sure you edit specific occurrences or only future occurrences so that the past occurrences are not altered.

3.6.11. Using tags to organize time entries

When you record a time entry, you need to have a client, a project and a task. For some people this solution is all they need to organize their time entries but others want more flexibility.

You can associate one or more tags to a time entry in the Tags field. They have to be separated by commas (eg: tag1,tag2,tag3). Once you have one or more entries with tags, you can then filter them in the Timesheet view. Just click the small arrow icon from the search field to tell it that you want to search by tags.

Tags are useful if you want to track time for multiple persons. You could use a tag for each person to know who did what.

3.6.12. Searching and filtering time entries

To learn more about the time you record in Fanurio, go to the Timesheet view and use the filters above the time entries table. The table will display only those time entries that match the selected filters. Time can be filtered by:

- **client status (any, active, not active)**,
- **project status (any, finished, not finished)**,
- **task status (any, completed, not completed)**,
- **invoiced status (any, invoiced, not invoiced)**,
- **billing status (any, billable, not billable)** and
- **date**.

Once you set these filters, the table footer will display the total of all visible time entries. For instance, you could use these filters to see how much time you've recorded on a date, week or month.

The table can also be filtered using a search field that can search by:

- **description**,
- **tag**,
- **client;project;task** and
- **invoice**.

The **client;project;task** option allows you to filter time entries by client, project, or task name.

If you have a client called **Aristotle** with a project called **Rhetoric** and a task called **Proofreading the manuscript**, just type **Aristotle;Rhetoric;Proofreading the manuscript** and it will display this exact task. If you type **Aristotle** it will display all time entries for this client whereas if you type **;Rhetoric** it will display all time entries for the project. The semicolon is very important as it helps Fanurio distinguish between fields.

3.6.13. Exporting time entries

Fanurio can export time to CSV, Excel, QuickBooks (*.iif) and XML. To export time, you can either:

- Go to File » Export » Export Timesheet to export all the time entries you recorded in Fanurio or
- Go to the Timesheet view and use the filters above the table to specify the time entries you want to export. Then right-click the table to display the contextual menu and select **Export Timesheet** to export the visible time entries.

3.6.14. Creating time reports

The previous section explains how you can filter or search the list of time entries so you can get a quick insight about your time. However, if you need to use this information outside Fanurio you can either export the list of time entries as a CSV or Excel file or create a time report. Time reports are more flexible because they allow you to use a template to format data.

Go to Reports » Time Report to create a report for your time. Just like in the Timesheet view, you can use several filters to specify which time entries should be included in the report:

- **Projects** specifies the projects included in the report.

- **Date** specifies the date interval when time was recorded.
- **Billing** specifies whether the report should include billable or non-billable time entries.
- **Invoiced** specifies whether the report should include invoiced or not invoiced time entries.

Fanurio comes with a set of default templates that can be selected from the Template drop-down box:

- **Time Statistics** displays various time statistics and charts like:
  - total recorded time,
  - number of working days,
  - average time per working day,
  - billable vs non-billable time,
  - invoiced vs not invoiced time,
  - hours by date, month or year depending on the report interval,
  - top clients, projects and task categories, and
  - distribution of time across clients, projects and task categories.
- **Timelog** shows the recorded time in detail and in chronological order.

  You may want to use it at the end of the day to see what you did. Please note that you can get a similar result if you go to the Timesheet view and restrict visible time entries to today.
- **Timelog by Date** shows the recorded time in detail and in chronological order but time entries are grouped by date.

  This template is similar to Timelog except that it groups time entries by date and it displays subtotals for each date.
- **Time Summary by Task and Date** displays time totals by task and date.

  You may want to use this template at the end of the week to see how much time you worked each day and on which tasks.
- **Time Summary by Task and Week** displays time totals by task and week.

  You may want to use this template at the end of the month to see how much time you worked each week and on which tasks.
- **Time Summary by Task and Month** displays time totals by task and month.

  You may want to use this template at the end of the year to see how much time you worked each month and on which tasks.
- **Timelog by Client, Project and Task** shows how much time was spent on each project, in detail.

  Time entries are grouped by client, project and task.
- **Timelog by Date, Client, Project and Task** shows how much time was spent on each day by project.

  This template is a combination of Timelog by Date and Timelog by Client, Project and Task. For each day, it shows time entries grouped by client, project and task.

If you need to create your own templates, please see this section to learn what placeholders you can use.

3.7. Working with expenses

3.7.1. About expenses

Expenses help you record money that you spend for a project. Here's what you can record with an expense:

- **Project:** The project where you want to record the expense. The project is mandatory as you can only record expenses on projects.
- **Date:** The date when the expense was made.
- **Amount:** The amount spent.
- **Description:** A small description of the expense.
- **Reference [optional]:** A reference number like the receipt number.
- **Category [optional]:** A category that can help you organize your expenses.
- **Tags [optional]:** A list of zero or more tags separated by commas. Tags can also help you organize expenses and they are more flexible than categories.
- **Notes:** Additional notes that you may need to make in case the description field is not suitable.

If a project is billable, expenses have a few more fields. Expenses from non-billable projects don't show these fields:

- **Billable:** When checked, it indicates that the expense is billable. Billable projects can have both billable and non-billable expenses.
- **Total:** Indicates the how much is charged for the expense. The default is to charge the amount spent but you can use the Markup link to bill more.

There are several ways you can record an expense:

1. Go to Business » New Expense.
2. Go to the Expenses view and use the **New** button.

   When you use this method and the date filter is set to a specific date, expenses are added by default to that date. This method is very useful if you need to enter expenses for previous dates as it saves you from specifying the date for each expense.
3. Open a project and go to the **Expenses** section. Then click the **New** button.

3.7.2. Configuring optional fields

Because not all expense fields are relevant to all users, you can hide optional fields that you don't use. Optional expense fields are fields that can be hidden in the expense windows (New Expense or Edit Expense). Here's how you can configure them:

1. Create a new expense (New Expense) or edit an existing one (Edit Expense).
2. Click the Configure button from the bottom-left corner.
3. Check the fields that you want to be visible and uncheck the ones that you want to hide.
4. Click Done. Fanurio will update the expense window to show only selected fields.

For instance, one of the optional fields is Reference. This field only makes sense if you need to record a receipt number or some other number that identifies your payment.

3.7.3. Billing expenses

If an expense is marked as billable, you can specify its total. By default, the total is the same as its amount but that can be changed. You can also specify the total by using a markup.

When a billable expense is added to an invoice, Fanurio creates an expense item for it so that it can be invoiced. Expense items have the same name as the expense category. If an expense doesn't have a category then the expense item has the generic name 'Expense'.

Expense items have Quantity equal to 1 which means that you will resell everything. If you are buying goods and you need to resell a certain amount (Quantity is not 1) then you should record your purchase as a non-billable expense and bill it using a product item. Product items allow you to specify both the quantity and the price for each individual product.

3.7.4. Using categories to organize expenses

Expenses can have an optional category that can be used to organize them.

The list of expense categories can be managed from Business » My Business Details+Projects +Expenses. Expense categories can also be specified when an expense is created or edited using the **New** link button.

Expenses can be searched or filtered by category in the Expenses view. Just click the small arrow icon from the search field to tell it that you want to search by category. Then type the name of the category that you want to search for.

3.7.5. Using tags to organize expenses

When creating or editing an expense, you can associate one or more tags with it in the Tags field. They have to be separated by commas (eg: tag1,tag2,tag3). If you need to tag multiple expenses, go to the Expenses view and select them. Then right-click to display the contextual menu and select Edit Tags.

Once you have one or more expenses with tags, you can then filter them in the Expenses view. Just click the small arrow icon from the search field to tell it that you want to search by tags. Then type the name of the tag that you want to search for.

3.7.6. Searching and filtering expenses

To learn more about the expenses you record in Fanurio, go to the Expenses view and use the filters above the expenses table. The table will display only those expenses that match the selected filters. Expenses can be filtered by:

- **client status (any, active, not active)**,
- **project status (any, finished, not finished)**,
- **invoiced status (any, invoiced, not invoiced)**,
- **billing status (any, billable, not billable)** and
- **date**.

Once you set these filters, the table footer will display the total amount of all visible expenses. For instance, you could use these filters to see how much you've spent on a date, week or month.

The table can also be filtered using a search field that can search by:

- **category**,
- **description**,
- **reference**,
- **tags**,
- **client;project** and
- **invoice**.

The **client;project** option allows you to filter expenses by client, or project name.

If you have a client called **Aristotle** with a project called **Rhetoric**, just type **Aristotle;Rhetoric** and it will display this exact project. If you type **Aristotle** it will display all expenses for this client. The semicolon is very important as it helps Fanurio distinguish between fields.

3.7.7. Exporting expenses

Fanurio can export expenses to CSV and Excel. To export them, you can either:

- Go to File » Export » Export Expenses to export all the expenses you recorded in Fanurio or
- Go to the Expenses view and use the filters above the table to specify the expenses you want to export. Then right-click the table to display the contextual menu and select **Export Expenses** to export the visible expenses.

3.7.8. Creating expenses reports

The previous section explains how you can filter or search the list of expenses so you can get a quick insight about how you spent your money. However, if you need to use this information outside Fanurio you can either export the list of expenses as a CSV or Excel file or create an expenses report. Expenses reports are more flexible because they allow you to use a template to format data.

Go to Reports » Expenses Report to create a report for your expenses. Just like in the Expenses view, you can use several filters to specify which expenses should be included in the report:

- **Projects** specifies the projects included in the report.
- **Date** specifies the date interval when time was recorded.
- **Billing** specifies whether the report should include billable or non-billable expenses.
- **Invoiced** specifies whether the report should include invoiced or not invoiced expenses.

Fanurio comes with a set of default templates that can be selected from the Template drop-down box:

- **Expenses** shows the list of expenses sorted by date and totals at the bottom.
- **Expenses by Category** shows expenses grouped by category.
- **Expenses by Client and Project** shows expenses grouped by project.

If you need to create your own templates, please see this section to learn what placeholders you can use.

3.8. Working with trips

3.8.1. About trips

Trips help you record the distance and time you travel with a vehicle. If you need to track taxi fares or similar travel expenses, please see the expenses section for more details. Here's what you can record with a trip:

- **Project:** The project where you want to record the trip. The project is mandatory as you can only record trips on projects.
- **Date:** The date when the trip was made.
- **Distance:** The total distance of the trip. Fanurio tracks distances in miles (mi) and kilometers (km) but we can introduce other distance units if necessary. The default distance unit is defined under Business » My Business Details+Projects+Trips.
- **Description:** A small description of the trip.
- **Start Time [optional]:** The time when the trip started.
- **End Time [optional]:** The time when the trip ended.
- **Start Location [optional]:** The location where the trip started.
- **End Location [optional]:** The location where the trip ended.
- **Tags [optional]:** A list of zero or more tags separated by commas. Tags can help you organize trips and record additional information like the vehicle name.

If a project is billable, trips have a few more fields. Trips from non-billable projects don't show these fields:

- **Billable:** When checked, it indicates that the distance recorded by the trip is billed at the specified rate (explained below). Billable projects can have both billable and non-billable trips.
- **Rate:** Indicates how much is charged for each unit (mile or kilometer). Trip rates can either be defined in the New Trip window by using the New link button or under Business » My Business Details+Projects+Trips.

There are several ways you can record a trip:

1. Go to Business » New Trip.
2. Go to the Trips view and use the **New** button.

   When you use this method and the date filter is set to a specific date, trips are added by default to that date. This method is very useful if you need to enter trips for previous dates as it saves you from specifying the date for each trip.
3. Open a project and go to the **Trips** section. Then click the **New** button.
4. Right-click a trip in the trips table and select **Duplicate Trip** from the contextual menu to create a copy of the selected trip.
5. Right-click a trip in the trips table and select **New Reverse Trip** from the contextual menu to create a copy of the selected trip that reverses the start and end locations.

3.8.2. Configuring optional fields

Because not all trip fields are relevant to all users, you can hide optional fields that you don't use. Optional trip fields are fields that can be hidden in the trip windows (New Trip or Edit Trip). Here's how you can configure them:

1. Create a new trip (New Trip) or edit an existing one (Edit Trip).
2. Click the Configure button from the bottom-left corner.
3. Check the fields that you want to be visible and uncheck the ones that you want to hide.
4. Click Done. Fanurio will update the trip window to show only selected fields.

For instance, two of the optional fields are Start Time and End Time. These fields only makes sense if you need to know how long your trips were. If so, you may also want to make the Duration column visible in the trips table.

3.8.3. Billing trips (mileage)

When you're billing a trip, you are actually billing its distance at a specified rate.

Trip rates are a key element when it comes to billing trips because you can't specify a price for each trip, instead you must define one or more rates that are shared by all billable trips. Trip rates can be defined in the New Trip window by using the New link button or under Business » My Business Details +Projects+Trips. You can always define new rates and disable the ones that you'll no longer use.

Let's consider the following example to understand how trips are billed. The HM Revenue & Customs mileage rates [http://www.hmrc.gov.uk/rates/travel.htm] are used in the UK and they are different for cars, motorcycles and bicycles. For our example, we'll only use the car and bicycle mileage rates:

- **Car:** 0.45 GBP / mile
- **Bicycle:** 0.20 GBP / mile

Let's suppose you recorded the following trips in May last year:

1. May 01: 20 mi (Car)
2. May 02: 30 mi (Car)
3. May 03: 50 mi (Car)
4. May 04: 4 mi (Bicycle)
5. May 05: 6 mi (Bicycle)

This means you travelled **100 miles** by car and **10 miles** by bicycle. When invoicing these trips, Fanurio will group them by rate under two different mileage items: Car and Bicycle. Each mileage item will have the same name and price as the rate and the quantity set to the total distance recorded on trips billed with that rate. In other words, your invoice will show like this:

Table 3.7. Sample mileage invoice

Item	Quantity	UM	Price	Total
Bicycle	10	mi	0.20 GBP	2 GBP
Car	100	mi	0.45 GBP	45 GBP
Total				47 GBP

For each mileage item you can show individual trips if you need to but this is optional.

3.8.4. Using tags to organize trips

When creating or editing a trip, you can associate one or more tags with it in the Tags field. They have to be separated by commas (eg: tag1,tag2,tag3). If you need to tag multiple trips, go to the Trips view and select them. Then right-click to display the contextual menu and select Edit Tags.

Once you have one or more trips with tags, you can then filter them in the Trips view. Just click the small arrow icon from the search field to tell it that you want to search by tags. Then type the name of the tag that you want to search for.

3.8.5. Searching and filtering trips

To learn more about the trips you record in Fanurio, go to the Trips view and use the filters above the trips table. The table will display only those trips that match the selected filters. Trips can be filtered by:

- **client status (any, active, not active)**,
- **project status (any, finished, not finished)**,
- **invoiced status (any, invoiced, not invoiced)**,
- **billing status (any, billable, not billable)** and
- **date**.

Once you set these filters, the table footer will display the total distance of all visible trips. For instance, you could use these filters to see how much you've travelled on a day, week or month.

The table can also be filtered using a search field that can search by:

- **description**,
- **tags**,
- **rate**,
- **client;project** and
- **invoice**.

The **client;project** option allows you to filter trips by client, or project name.

If you have a client called **Aristotle** with a project called **Rhetoric**, just type **Aristotle;Rhetoric** and it will display this exact project. If you type **Aristotle** it will display all trips for this client. The semicolon is very important as it helps Fanurio distinguish between fields.

3.8.6. Exporting trips

Fanurio can export trips to CSV and Excel. To export them, you can either:

- Go to File » Export » Export Trips to export all the trips you recorded in Fanurio or
- Go to the Trips view and use the filters above the table to specify the trips you want to export. Then right-click the table to display the contextual menu and select **Export Trips** to export the visible trips.

3.8.7. Creating trips reports

The previous section explains how you can filter or search the list of trip so you can get a quick insight about how you much you travelled. However, if you need to use this information outside Fanurio you

can either export the list of trips as a CSV or Excel file or create a trips report. Trips reports are more flexible because they allow you to use a template to format data.

Go to Reports » Trips Report to create a report for your trips. Just like in the Trips view, you can use several filters to specify which trips should be included in the report:

- **Projects** specifies the projects included in the report.
- **Date** specifies the date interval when time was recorded.
- **Billing** specifies whether the report should include billable or non-billable trips.
- **Invoiced** specifies whether the report should include invoiced or not invoiced trips.

Fanurio comes with a set of default templates that can be selected from the Template drop-down box:

- **Trips** shows the list of trips sorted by date and totals at the bottom.
- **Trips by Client and Project** shows the distance travelled for the selected projects in the specified date interval.

If you need to create your own templates, please see this section to learn what placeholders you can use.

3.9. Working with products

3.9.1. About products

Projects can manage tasks, expenses and trips but billable projects can also manage products.

Products may seem an unnecessary concept but they provide a mechanism to bill anything that cannot be billed using tasks, expenses or trips. A product could be used to sell a computer, a set of icons or the monthly fee for website hosting services.

Let's consider the following example. You are a graphic designer who has created a set of icons. You license the icons to some of your clients for a small fee. How should you record this in Fanurio?

- It doesn't make sense to create a task to bill the icons because you are not spending time again to create them. Tasks are used to bill work and you're not working this time.
- It doesn't make sense to create an expense because you are not purchasing the icons from someone else. Expenses are used to record money that you spend and you're not spending anything this time.
- This is not a trip.

So, the only solution is to use something else and that is a product.

3.10. Working with invoices

3.10.1. About invoices

Invoices allow you to bill goods and services to your clients.You can create both regular invoices that bill a client directly and project invoices that bill your projects (tasks, expenses, trips, products).

An invoice has the following properties:

- **Client:** The client for whom the invoice is created.
- **Attention [optional]:** Overrides the client attention field if you need to send the invoice to a different person.
- **Number:** A number that identifies the invoice. Invoice numbers can be generated automatically.
- **Date:** The date when the invoice was issued.
- **Terms:** Specifies the date when the invoice is due in days relatively to the invoice date.
- **Reference [optional]:** A reference number like the purchase order number.
- **Period [optional]:** A free text field that contains the invoicing period. It can be a date range **June 1 - June 30**, a month **June 2000** or something else like **Q1/2000**. This field is filled-in automatically when an actual period is selected in the Add Project Items window.
- **Currency:** The currency used for the invoice. If the invoice has items with different currencies than this one, they are converted to the invoice currency based on the invoice exchange rates.

  This field is visible only if the application uses two or more currencies.
- **Taxes:** The tax group used for the items of the invoice. If None is selected, no taxes are applied.

  This field is visible only if taxes are enabled.
- **Notes:** Additional notes.

3.10.2. Configuring optional fields

Because not all invoice fields are relevant to all users, you can hide optional fields that you don't use. Optional invoice fields are fields that can be hidden in the invoice windows (New Invoice or Edit Invoice). Here's how you can configure them:

1. Create a new invoice (New Invoice) or edit an existing one (Edit Invoice).
2. Click the Configure button from the bottom-left corner.
3. Check the fields that you want to be visible and uncheck the ones that you want to hide.
4. Click Done. Fanurio will update the invoice window to show only the selected fields.

For instance, one of the optional fields is Reference. This field only makes sense if you need to record a purchase order reference number.

3.10.3. Creating a regular invoice

A regular invoice is the most simple invoice you can create in Fanurio. In order to create an invoice, you need to:

- Go to Business » New Invoice or
- Click the New Invoice button from the toolbar or

- Right-click a billable client and select New Invoice from the contextual menu.

Once the **New Invoice** window is displayed, you can start adding regular items to the invoice by clicking the New button.

If your business sells some items more frequently, you can add them to the business catalog that can be found under Business » My Business Details+Billing+Catalog. Catalog items make it easier to create regular invoice items.

For instance, as a web-designer you could define an item called **Hosting** that is charged $10/month. Then, every time you need to bill this item to a client, you just create a new regular item based on this catalog item. This will save you time from filling-in the item fields each time.

3.10.4. Creating a project invoice

Regular invoices (see above) can help you bill your clients but they are not helpful if you need to bill your clients by project. A project invoice is a regular invoice that bills one or more projects, it's not a different type of invoice.

You can create a project invoice for a client if it has at least one billable project with one or more billable elements (tasks, expenses, trips or products). In order to create an invoice, you need to:

- Go to Business » New Invoice or
- Click the New Invoice button from the toolbar or
- Right-click a billable client and select New Invoice from the contextual menu (this will select all its projects by default) or
- Right-click a billable project and select New Invoice from the contextual menu.

Once the **New Invoice** window is displayed, Fanurio will display the **Add Project Items** window to let you choose the project elements (tasks, expenses, trips and products) that you want to add to the invoice. Click the Add button and Fanurio will add a project item to the invoice for each selected element.

- Tasks are billed as service items,
- Expenses are billed as expense items,
- Trips are billed as mileage items and
- Products are billed as product items.

If you add a project item to an invoice but then change your mind, you can always delete it. When you delete a project item, the project element that it bills can be billed again. For instance, when you delete an expense item, the expense can be billed again.

3.10.5. Using taxes

Taxes are disabled by default but can be enabled from Business » My Business Details+Billing+Taxes.

Once you enable taxes, they will be applied to the invoices you create. If some of your clients are tax exempt, you can mark them as "tax exempt" in the Billing section and no tax will be applied to their invoices.

To quickly define your taxes, we recommend that you use the **Tax Wizard**. Only if the wizard doesn't do what you want, you should try to build a tax group manually.

To deal with a whole range of possible taxes and tax combinations, Fanurio uses two concepts: the **Tax** and the **Tax Group**. A **Tax** has a name and a default rate while a **Tax Group** contains one or more taxes. In order to apply taxes to an invoice, you need to group taxes in a tax group. Even if you have just one tax.

Rounding tax totals

When calculating tax totals for invoices, Fanurio will round amounts that have more than three decimals using one of the following rules. The default rounding rule for new invoices can be specified in Business » My Business Details+Billing+Taxes.

Please note that Fanurio will use the default rule only when creating a new invoice. If you change it at business level, existing invoices will not be changed. Before deciding which rounding rule you should be using, please consult an accountant.

- **round half even:** Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor. This method is also called unbiased rounding, convergent rounding, statistician's rounding, Dutch rounding, Gaussian rounding, odd–even rounding, or bankers' rounding.

  This is the default rounding rule used by Fanurio.

  - 100.454 is rounded to 100.45
  - *100.455 is rounded to 100.46*
  - 100.456 is rounded to 100.46
  - *100.465 is rounded to 100.46*
  - *100.475 is rounded to 100.48*

- **round half up:** Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up. This method is also called round half away from zero or commercial rounding and it is often used for currency conversions and price roundings.

  Fanurio uses this rounding rule by default when importing an iBiz database starting with version 4.0.

  - 100.454 is rounded to 100.45
  - *100.455 is rounded to 100.46*
  - 100.456 is rounded to 100.46

- **round half down:** Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.

  - 100.454 is rounded to 100.45
  - *100.455 is rounded to 100.45*
  - 100.456 is rounded to 100.46

3.10.6. Numbering invoices automatically

Automatic invoice numbering is disabled by default but can be enabled from Business » My Business Details+Billing+Invoice numbering.

To generate an invoice number automatically, Fanurio needs a counter and a number format. The number format can be something simple like counter on three or more digits (eg 001, 0001, 00001) or something more complex that involves prefixes, sufixes and dates.

Invoices can be numbered sequentially for all clients or separately for each client. If you need to number invoices separately for each client, Fanurio will use the invoice counter defined at client level. The invoice counter of a client is incremented only when an invoice is created for that client. Edit a client and go to the Billing tab to access its invoice counter.

Fanurio uses the Freemarker [http://freemarker.sourceforge.net/] syntax for the number format, the same syntax used for the templates. Here are a few number formats that assume the counter is 1 (one) and the year is 2020.

Table 3.8. Invoice number formats

Name	Number Format	Sample Number
Four-digit numbers To use more digits in your format, just add more zeros.	${counter?string("0000")}	0001
Fixed prefix format This format builds on the previous one by adding the **ABC-** prefix. Please note that it's important to put the prefix outside ${...}.	ABC-${counter? string("0000")}	ABC-0001
Client code prefix format The invoice number format can use two placeholders that access the code and name client fields. Include **${client.code}** or **${client.name}** if you need to build a more complex format. This format uses the **${client.code}** placeholder and assumes the client code is set to **ACME**.	${client.code}-${counter? string("0000")}	ACME-0001
Date prefix format This format uses a variable prefix that changes with the year. The date prefix is generated by the expression **${.now?string("yy")}**. If you need to display the whole year use **${.now?string("yyyy")}** instead. The text "yyyy" represents the date pattern used to format the date which uses the Java date format syntax [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html].	${.now?string("yy")}- ${counter?string("0000")}	20-0001

3.10.7. Billing in multiple currencies

Every invoice has a base currency that is specified when the invoice created. If the invoice items use a different currency, you have to enter exchange rates for each currency different from the base currency. Exchange rates can only be entered manually.

Let's consider the following example to understand when this can be helpful. You have a business based in Germany that does business with clients from across the EU and the USA. European clients

are billed in Euros while American clients are billed in US dollars. When working with US clients, you still record expenses in Euros but in the end you have to bill them in US dollars.

3.10.8. Discounting an invoice

With Fanurio you can discount service items or you can discount the total invoice value. In either case, invoices can show both the discounted values and the regular ones. Discounts can be specified as a percentage or as a fixed value. Below we'll show you how discounts can be used to create more accurate invoices.

Note: If you want to customize the invoice template to display discount information, take a look at the placeholders available for items and invoices.

Making a discount for the entire invoice is easy. When you create an invoice, just specify a discount for the total invoice value as a percentage or as a fixed sum.

Let's say that the total invoice value (before taxes) is $1,330.00 and you want to round it to $1,300.00. In this case, you apply a fixed discount of $30.00 and the invoiced value will be $1,300.00. The discount is not taxed if taxes apply to the invoice.

3.10.9. Discounting individual items

The total invoice discount makes sense when you need to discount the entire invoice but not if you want to discount only certain items. You may want to lower your rate or you may want to bill less work (hours or units). Let's see a few discount examples and how they can help you.

Example 3.8. Bill using a lower rate

You may want to offer introductory rates to first time clients. Let's say that you charge $100 for a service but this time you want to charge $80. When you create an item to record work for that service, just discount its rate using a fixed discount of $20. You could also use a percentage discount of 20% to get the same result.

Example 3.9. Bill less work

If you agreed to 10 hours for a service but it took you 12.5 hours then you can discount them to match the initial agreement. Just apply a fixed discount of 2.5 hours and you'll only bill 10 hours. On the invoice, you can show that it took you 12.5 hours but you only billed 10 hours.

Example 3.10. Don't bill work

Another situation is when you are offering a service that should be included on the invoice but should not be charged. In this case, create an item and discount its units (hours) by 100%. The total invoiced value for such an item will be zero.

Once you have one or more discounted items, you can easily see which ones are discounted. Discounted items have a red corner in the items table. Just move the mouse over them (don't click, just keep it over) to see more details.

3.10.10. Cancelling an invoice

Sometimes an invoice will never be paid because the client refuses to do so or because it goes out of business. To handle cases like these when you know an invoice will never be paid by the client, you can just cancel it. Please note that only unpaid invoices can be cancelled. You can also cancel partially paid invoices.

Once cancelled, you can used the filters above the invoices table to show or hide them. Use the **Status** filter and select **Not Paid** or **Overdue**. Then you can filter invoices by their **Cancelled** status.

3.10.11. Using templates to view, export and email invoices

If you want to view, export or email an invoice, you will need to use an invoice template. Templates specify the visual design and content of an invoice document.

Since most business have different requirements when it comes to the layout of their invoices, Fanurio has a **very powerful** template engine that can handle many file formats (HTML, Microsoft Word, OpenDocument Text, XML, plain text). If you already create your invoices in one of these formats, it's very likely that they can be easily transformed into templates for Fanurio.

We've created a separate guide just for templates. To get started, you only need to read the introduction.

3.10.12. Exporting an invoice

Fanurio can export invoices to various formats depending on what type of template you are using. By default, Fanurio comes with an .html template that can be used to export invoices to .html and .pdf.

To export an invoice:

1. Go to the Invoices view,
2. Locate the invoice you want to export,
3. Right-click the invoice to display the contextual menu,
4. Select **Export Invoice**.

3.10.13. Sending invoices by email

You can send invoices by email to your clients right from Fanurio. Just go to the Invoices view, select the invoice you want to email, right-click it and select **Email Invoice** from the contextual menu.

You can send invoices by email if the following conditions are met:

1. The email feature is enabled and configured correctly
2. You entered your email address in the **Contact** section from Business » My Business Details. You need to do this so that the email is sent on your behalf.
3. You entered an email address for the client.

3.10.14. Searching and filtering invoices

To learn more about your invoices, go to the Invoices view and use the filters above the table. The table will display only those invoices that match the selected filters. Invoices can be filtered by:

- **status:** paid, unpaid or overdue,
- **cancelled:** yes or no,
- **date (creation)**,
- **date (paid)**,
- **reference (purchase order)**,
- **number** or
- **client**.

Once you set these filters, the table footer will display totals for:

- **balance** (how much money you must receive),
- **total** (how much money the invoices are before taxes) and
- **taxes** (total tax money charged to your clients if you use taxes).

For instance, you could use these filters to see all overdue invoices issued this year for a certain client. The table footer will help you see how much money the client owes you.

3.10.15. Exporting invoices

Fanurio can export invoices to CSV and Excel. To export them, you can either:

- Go to File » Export » Export Invoices to export all the invoices you recorded in Fanurio or
- Go to the Invoices view and use the filters above the table to specify the invoices you want to export. Then right-click the table to display the contextual menu and select **Export Invoices** to export the visible invoices.

3.10.16. Creating sales reports

The previous section explains how you can filter or search the list of invoices so you can get a quick insight about your sales. However, if you need to use this information outside Fanurio you can either export the list of invoices as a CSV or Excel file or create a sales report. Sales reports are more flexible because they allow you to use a template to format data.

Go to Reports » Sales Report to create a report for your invoices. Just like in the Invoices view, you can use several filters to specify which invoices should be included in the report:

- **Client** indicates if you create the report for a specific client or for all clients.
- **Date** specifies the date interval when invoices were issued.
- **Paid Date** specifies when invoices were paid (if paid).
- **Status** specifies whether invoices are paid or not.

Fanurio comes with a set of default templates that can be selected from the Template drop-down box:

- **Sales Statistics** displays various salesstatistics and charts like:
  - total sales,
  - number of months with invoices,
  - average sales per month,
  - paid, due and overdue amounts,
  - sales by month and client,
  - unpaid amount by age, and
  - outstanding clients.
- **Invoices** shows the list of invoices sorted by date and displays totals at the bottom.

  This template is useful if you need to create a statement for a client or if you simply need to list invoices chronologically.

- **Invoices Summary by Client and Month** displays totals by client and month.

  This template is useful if you need to analyze sales by client and month.

- **Invoices Summary by Client and Year** displays totals by client and year.

  This template is useful if you need to analyze sales by client and year.

- **Invoices Summary by Client** displays totals by client.

  This template is useful if you need to analyze sales by client.

- **Invoices by Client** is similar to **Invoices** except that it groups invoices by client and displays subtotals for each client.

  This template is useful if you need to analyze sales by client.

- **Invoices by Month** is similar to **Invoices** except that it groups invoices by month of billing (Date field) and displays subtotals for each month.

  This template is useful if you need to analyze sales by month.

- **Invoices by Year** is similar to **Invoices** except that it groups invoices by year of billing (Date field) and displays subtotals for each year.

  This template is useful if you need to analyze sales by year.

- **Invoices by Month of Payment** is similar to **Invoices** except that it groups invoices by month of payment (Paid Date field) and displays subtotals for each month.

  This template is useful if you need to see how much money you've been paid each month.

- **Invoices by Year of Payment** is similar to **Invoices** except that it groups invoices by year of payment (Paid Date field) and displays subtotals for each year.

  This template is useful if you need to see how much money you've been paid each year over several years.

If you need to create your own templates, please see this section to learn what placeholders you can use.

3.11. Working with payments

3.11.1. Recording payments for an invoice

When you receive payment for an invoice, go to the Invoices view and select the invoice. Then right-click it and select **New Payment** from the contextual menu to record the payment.

To see, edit or delete payments, go to the Payments view.

3.11.2. Using deposits

All billable clients have a deposits account. You can use this account to record money you receive in advance for your work. There are several ways to record a deposit from a client:

1. Go to Business » New Deposit.
2. Go to the Projects view and right-click a client to display the contextual menu. Select New Deposit from the contextual menu.
3. Go to the Projects view and right-click a client to display the contextual menu. Select Edit Client from the contextual menu to edit the client then go to the Deposits tab and click New.

Once you have money in a client account, you can use it to pay invoices.

The **New Payment** window allows you to withdraw money from the deposits account to pay an invoice. Just click the apply deposit box and the specified amount will appear as a negative transaction in the deposits account.

3.11.3. Searching and filtering payments

To learn more about the payments you receive, go to the Payments view and use the filters above the payments table. The table will display only those payments that match the selected filters. Payments can be filtered by:

- **date**,
- **invoice**,
- **reference** or
- **client**.

Once you set these filters, the table footer will display the total of all visible payments. For instance, you could use these filters to see how much money a client has paid you this year.

3.11.4. Exporting payments

Fanurio can export payments to CSV and Excel. To export them, you can either:

- Go to File » Export » Export Payments to export all the payments you recorded in Fanurio or
- Go to the Payments view and use the filters above the table to specify the payments you want to export. Then right-click the table to display the contextual menu and select **Export Payments** to export the visible payments.

3.12. Fanurio + QuickBooks

If you are a QuickBooks user, you can track time with Fanurio and then import it in QuickBooks to bill your clients. Fanurio can export time to an .iif file that can then be imported by QuickBooks Pro.

3.12.1. Initial setup

Before you can export time from Fanurio for QuickBooks Pro, you need to make some settings for the import to go smoothly. You can make these settings by hand or you can use a wizard to make them automatically. We highly recommend that you use the wizard as it can save you a lot of time.

In both cases, you will need an .iif file that has to be exported from QuickBooks. Make sure QuickBooks Pro is running and go to Timer » Export Lists for Timer (TimerList.iif) to export the file that will be used to configure Fanurio.

A. Automatic setup

Go to File » Import » Import from QuickBooks to launch the wizard. It will ask you to locate the TimerList file mentioned above.

The wizard will help you make all the settings, import the customers and items that you want to use from Fanurio. Please note that you can run this wizard at any time, existing customers and items will not be imported again.

B. Manual setup

1. Go to Tools » Options and select the Third Party panel. Enable QuickBooks export and enter your QuickBooks company id in the Company Create Time field.

   You can get your company id from the TimerList file. Open it with a text editor and get the value of the COMPANYCREATETIME field from the TIMERHDR line.

2. Go to Business » My Business Details and select the Company panel. Enter your QuickBooks company name in the Name field.

   You can get your company name from the TimerList file. Open it with a text editor and get the value of the COMPANYNAME field from the TIMERHDR line.

3. Go to Business » My Business Details and select the Contact panel. Enter your QuickBooks employee name in the Attention field.

   You can get your employee name from the TimerList file. Open it with a text editor and get the value of the NAME field from the EMP line.

4. Go to Business » My Business Details and select the Projects panel. Create task categories with the same names as those of the invoice items you have in QB.

   You can get the names of your invoice items from the TimerList file. Open it with a text editor and get the values of the NAME field from the INVITEM lines.

5. Add one or more clients that have the same name as the ones you have in QuickBooks.

   You can get the names of your clients from the TimerList file. Open it with a text editor and get the values of the NAME field from the CUST lines.

Once you complete the initial setup, you can start recording time with Fanurio.

3.12.2. Exporting time from Fanurio

See this section to learn how to export time to an *.iif file that can be imported by QuickBooks Pro.

3.12.3. Importing time in QuickBooks Pro

Here's how to import time in QuickBooks Pro from an .iif file:

1. Make sure QuickBooks is running

   We strongly recommend backing up your QuickBooks data before doing your first import.

2. Go to Timer » Utilities » Import » Timer Activities

3. Find the exported .iif file on your computer and select it for import.

   If QuickBooks warns you that the import file was created with an older version, click OK (Fanurio can't guess which version of QuickBooks you are running).

4. A small timer window will open showing the results of the import. Click "View Report" to verify your time data was imported correctly.

5. Close the status window and you are done!

3.13. Fanurio + IGG Software's iBiz

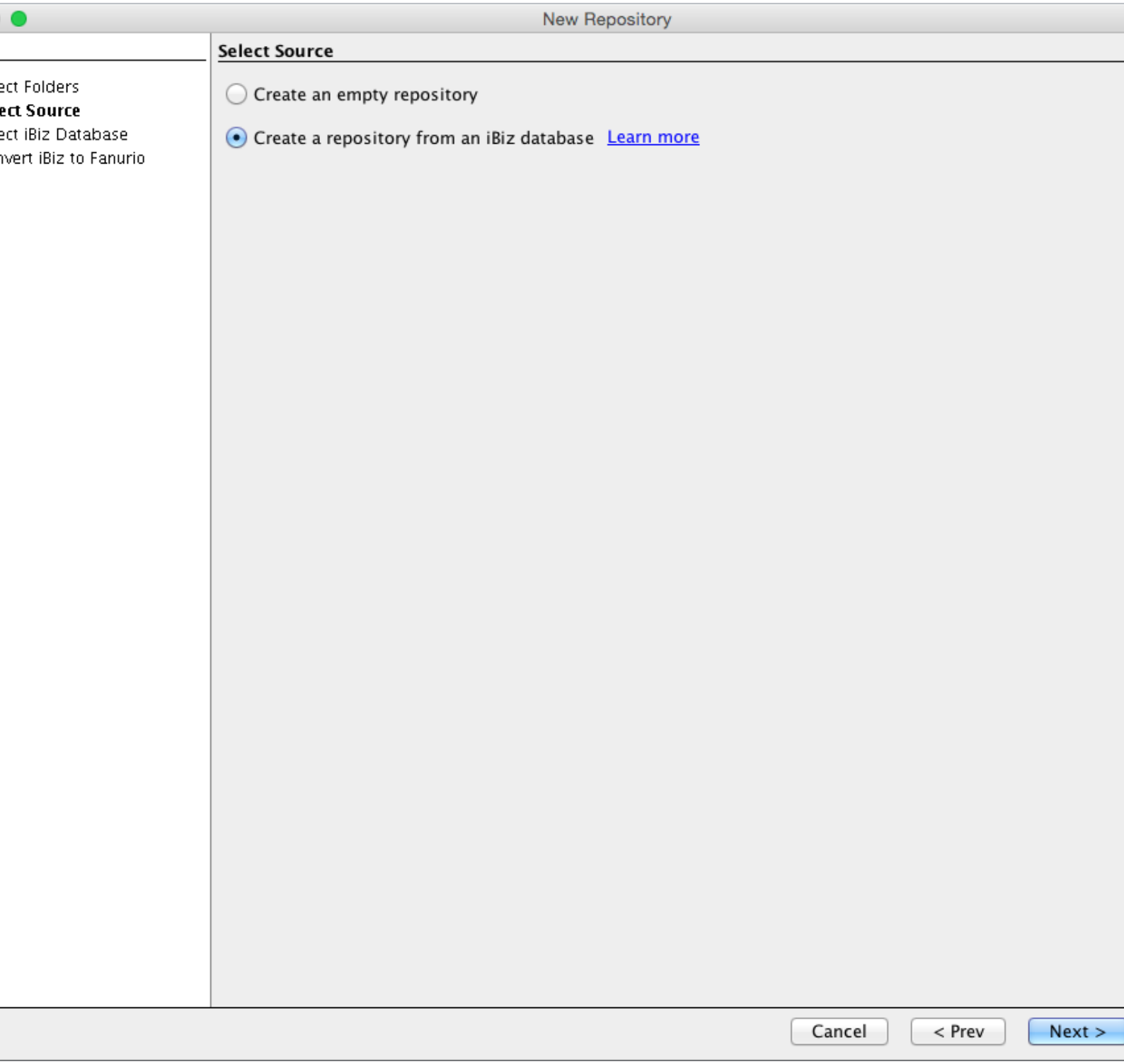
IGG Software [http://www.iggsoftware.com/] have announced [http://www.iggsoftware.com/blog/2013/10/the-story-of-ibiz-and-saying-goodbye/] that they are officially ceasing the development of iBiz, their time tracking and billing application. To help iBiz users move to a new solution and keep their old iBiz data, we've created an import module that converts an iBiz (4.1.4 and older) database to a Fanurio repository.

This guide tells you everything you need to know to migrate from iBiz to Fanurio: how to migrate an iBiz database, what to do if conversion errors occur, how iBiz data is imported in Fanurio and finally a detailed analysis of how each iBiz feature is supported by Fanurio.

3.13.1. How to import an iBiz database

You can import an iBiz database when you're setting up Fanurio or when you're creating a new repository (File » New Repository...). The steps are the same but the following screenshots are taken from the **New Repository** setup guide.

First, you need to select the **Create a repository from an iBiz database** option when you reach the **Select Source** step.



New Repository - Select Source

Click Next to go to the next step, **Select iBiz Database**. This is where you specify the iBiz folder and a few more settings.

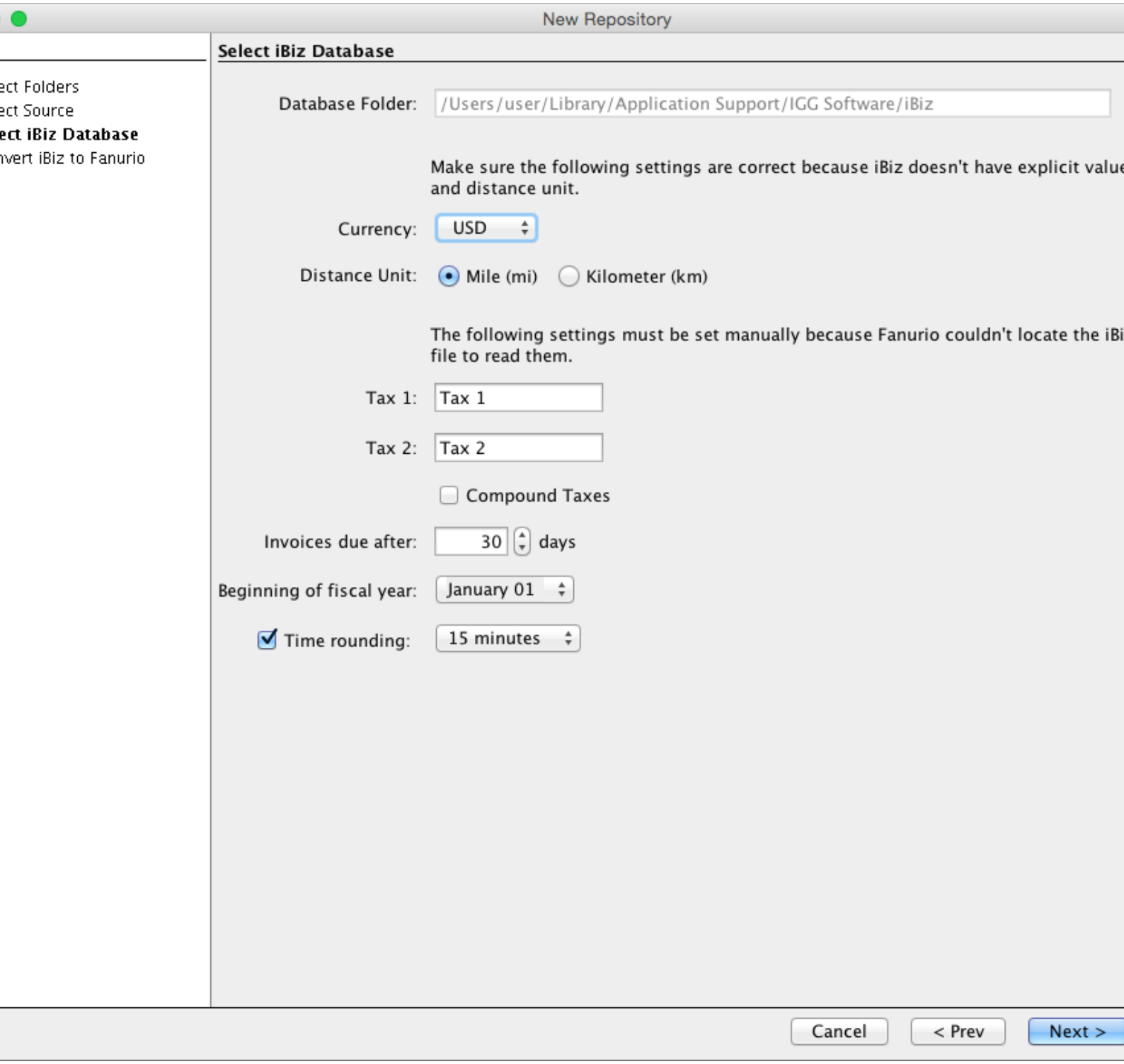
Note

Usually you don't have to select a folder because Fanurio will detect it automatically but if you're installing Fanurio on a new computer that doesn't have iBiz, you will have to create an iBiz folder from an iBiz backup file. To do this you need a backup file, let's suppose you

have one called **backup.ibizbackup** that's saved on your Desktop. Rename it to **backup.zip** and then double click it to extract the iBiz folder from the archive to your Desktop. You can now click the Browse... button to select it as your iBiz folder.

iBiz doesn't have explicit values for currency and distance unit so you need to make sure Fanurio detects them correctly. Fanurio can handle multiple currencies and it can track distances in both miles and kilometers.

Besides currency and distance unit, Fanurio needs a few more settings like the tax names and whether they are compounded or not to import the iBiz database. These settings are read from the iBiz preferences file. If Fanurio doesn't find the preferences file, you must set them manually. Of all these settings, the setting that indicates whether taxes are compounded is **very important** because if it's not set correctly, the invoice totals will not be calculated correctly by Fanurio.



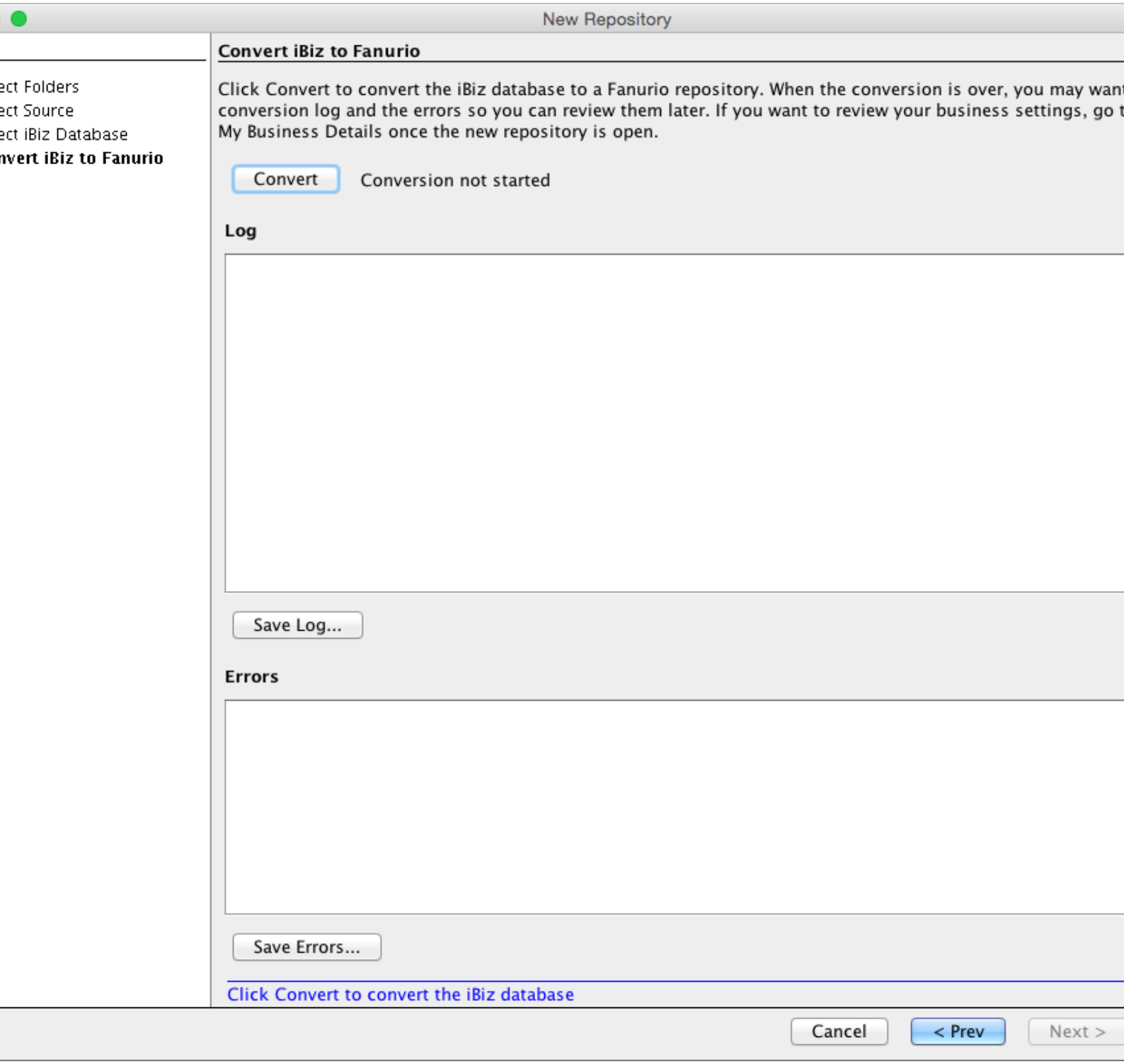
New Repository - Select iBiz Database

Click Next to go to the next step, **Convert iBiz to Fanurio**. This is where the iBiz database is converted to a Fanurio repository.

Just click Convert to start the conversion. Once the conversion starts, Fanurio will display what it does in the Log area. Conversion errors (if any) will also be displayed in the Error area.

- If the conversion fails, the status label next to the Convert button will tell you that it failed (see below for instructions in case the conversion fails or if it succeeds with errors).

- If the conversion succeeds, the status label next to the Convert button will tell you that it succeeded.



New Repository - Convert iBiz to Fanurio

Note

Please keep in mind that even though Fanurio reports a successful conversion, there may be differences that we are not aware of yet. We do our best to provide an accurate import module but since iBiz was not developed by us there may be things that we've missed.

To make sure your data is imported correctly, you should create reports both in iBiz and Fanurio and compare them. For instance, you may want to check the total amount of all invoices is the same in both applications. In Fanurio, you can see this amount in the Invoices view at the bottom of the invoices table.

Click Finish to close the guide and open the new repository that contains the iBiz data.

3.13.1.1. What to do if the conversion fails?

If the conversion fails, click the Save Log... button to save the conversion log, then send it to us. We'll analyze the error and send you instructions on how to fix it.

3.13.1.2. What to do if the conversion succeeds with errors?

Sometimes, even if the conversion succeeds, you may see conversion errors displayed in the Errors area. In these cases, you should click the Save Errors... button to save the errors and then click Finish to open the repository. You will then have to open the errors file to see the errors that have to be fixed.

From what we know so far, these errors are caused by invoiced projects that were moved to other clients or by invoiced job events that were changed after they were invoiced.

Moved invoiced projects

If you invoiced a project and moved it to another client then Fanurio will not be able to import the invoice for that project. In such cases, you need to use iBiz to move the project to its original client and run the import wizard again.

Altered invoiced job events

If you have a $100 job event and you invoice it, the invoice total will be $100. But once you create the invoice, you can change the job event to $50 and the invoice will remain unchanged. Because iBiz doesn't lock a job event once it's invoiced, this may lead to conflicting situations where a project job event has a different value ($50) than the invoice job event ($100) although they are the same job event. When Fanurio imports an invoice from iBiz, it uses the project value of a job event and not the invoice value.

Here's how this error looks like:

```
ACME: The amount for invoice 001/invoice-uuid doesn't match. iBiz = $100.00; Fa
```

Unfortunately, these errors can't be fixed automatically so you need to fix them manually. Use the client name (ACME in this example) and the invoice number (001 in this example) to locate the invoice in iBiz and Fanurio and compare them to see what's different. To locate an invoice by its number in Fanurio, go to the Invoices view and use the search field above the table to search by invoice number.

You can either fix these differences in iBiz and run another conversion (recommended) or you can fix them in Fanurio. To fix them in iBiz, you should compare the invoice with the project and fix any project job events that are different. In the above example, the invoice would should a job event with a total of $100 but the job event from the project would have a total of $50. The project job event needs to be changed to $100.

If you need help with this, please contact us.

3.13.1.3. Which iBiz elements are imported by Fanurio?

Fanurio imports the following elements:

- Clients. Contact details are imported from Apple Contacts only if it contains them. If you import your iBiz database from a backup file on a different computer that doesn't have your clients in Apple Contacts then Fanurio will not import any contact details, it will import only the client's name.

- Projects
- To Do items as non-billable tasks
- Global job events (Timed, Quantity, Flat) as task categories
- Job events (Quantity, Flat) as tasks
- Job events (Timed) as tasks with one time entry. The time recorded on the timed job event is added to a task as a time entry with the same time.
- Job events (Mileage) as trips
- Job events (marked as expense) as expenses
- Invoices
- Statements with late fees as regular invoices with one item for the late fee
- Payments
- Taxes
- Preferences (if iBiz is installed on the same computer as Fanurio)
  - tax names
  - whether taxes are compounded
  - time rounding rule
  - beginning of fiscal year
  - invoice due days

Fanurio doesn't import the following elements:

- Timer logs. They can't be used to determine the actual time recorded on a timed job event.
- File references from the File Cabinet.
- Invoice reminders
- Estimates and estimate job events. A solution would be to import them as separate projects.
- Taxes when an invoice uses one tax for some events and another one for the others, in other words not all events use the same taxes. A solution would be to import such taxes as separate invoice items.

For more details on how Fanurio is different from iBiz, see the following section.

3.13.2. How is Fanurio different from iBiz?

This section contains a detailed analysis of how each iBiz feature is supported by Fanurio and is meant to help iBiz users switch to Fanurio faster.

Some features are not supported yet but we're constantly improving Fanurio so if you want to see them implemented in future releases, please let us know. The more people request a certain feature, the more likely it is it will be implemented in a future version.

3.13.2.1. Clients

iBiz needs Apple Contacts to store the contact details of its clients but Fanurio doesn't. Fanurio stores everything in its database. However, Fanurio allows you to update the contact details of a client stored in Apple Contacts manually. Fanurio also has additional fields for its clients that allow you to record

registration information like business number or tax number that you may want to include in your invoices.

iBiz considers a client inactive if it doesn't have open invoices or outstanding balances on its billing account. Fanurio allows you to mark a client as active or inactive.

Fanurio doesn't have an equivalent for the iBiz client pop-up inspector. You can find out how much money a client owes you in the Invoices view if you filter invoices by client name.

Fanurio doesn't have client groups.

To learn more about how Fanurio deals with clients, see the clients section.

3.13.2.2. Projects

Fanurio projects have the same fields as iBiz projects (name, number, notes, start date and due date) and a few more such as description, reference and tags. Fanurio projects don't have a status field but they can be marked as completed. If you need to track other project statuses, you can use tags. Overdue projects are displayed in red and completed projects are displayed in gray.

In iBiz all projects are billable but in Fanurio billing is optional, you can have both billable and non-billable projects. If a project is billable, you can specify a default billing rate for its tasks.

Fanurio has two project views, a tree view and a table view. The table view is similar to the projects view available in iBiz and is recommended for former iBiz users.

Fanurio doesn't sync projects automatically with iCal but it can import time from an iCalendar file and it can export projects to an iCalendar file.

To learn more about how Fanurio deals with projects, see the projects section.

3.13.2.3. Job events

iBiz uses a list of job events to record various things on a project like hourly-rated or flat fee activities, expenses, items bought or sold, and mileage logged on a vehicle.

Unlike iBiz, Fanurio doesn't use **a single list of job events of different types** but **different lists for each type of element** (tasks, expenses, trips, and products). Using multiple lists instead of just one list allows you to record specific information for each type of element, for instance tasks can have due dates while trips can record distances. In iBiz all types of job events have almost the same fields (name, type, rate, taxes, notes) except for the quantity field which has a different name depending on the type of the job event (time, quantity, mileage). Prior to version 3.0 Fanurio used a single list model as well but this model has some limitations and we decided to replace it with a multiple list model. You can read more about this decision in the release notes.

Here's how each type of job event can be represented in Fanurio using tasks, expenses, trips and products.

- **Timed**

  According to the iBiz manual, this type is used to "*bill according to an hourly rate*" and that "*time can be entered manually or logged automatically with timers*".

  An iBiz timed job event has a Time field that contains its total time and two fields (To and From) that record the start and finish dates of the job event. The total time is not always the difference between From and To and can be changed manually to an arbitrary value. A timed job event could be used to record a single working session (for instance 'replied to email, 15 minutes') or multiple working sessions (for instance 'wrote text for brochure, 6 hours in 3 days'). iBiz can also round time but once it rounds it the actual recorded time is lost and replaced with the rounded value.

  In order to replace timed job events, you need to use tasks and time entries in Fanurio. Fanurio uses tasks to help you manage and bill work for your projects. A task has one or more time entries that

represent the time spent working on it. For instance, if you have a service called 'Phone support' and you need to bill 15 minutes then you have to create a task and a time entry for the 15 minutes of work. In some cases, this may be more work because you need to create two objects (a task and a time entry) but recording time as a separate object also has some advantages especially if you need to round it.

Fanurio can handle complex time rounding rules like "round time to the nearest 15 minutes interval but bill at least 30 minutes". When a billable task rounds time, it doesn't lose the time recorded on its time entries like iBiz. The actual recorded time is different from the billable time.

We recommend using fewer tasks with multiple time entries. For instance, if you need to keep track of how much time you spend on the phone with the client for a certain project, instead of creating a task each time you talk to him or her, you should create a single task and then add time to that task for each call.

When you invoice a task, its time entries can be displayed by the invoice (it depends what template you're using) so you can show a detailed log of what you did.

- **Flat rate**

  According to the iBiz manual, this type is used to *"bill according to a flat fee, whether it be for services performed or goods purchased for the project*".

  If you need to bill services then you should use tasks billed in units. This allows you to track time for yourself.

  If you need to bill other goods then you should use products.

- **Quantity**

  According to the iBiz manual, this type is used to *"track sales of goods and bill on a per-item basis"*.

  Fanurio uses products to record the same information. If your projects don't have a products list, go to Business » My Business Details+Projects to enable them.

- **Mileage**

  According to the iBiz manual, this type is used to *"track distance traveled in a vehicle and bill on a per-mile/ kilometer basis*".

  Fanurio uses trips to record and bill the distance you travel with a vehicle. If your projects don't have a trips list, go to Business » My Business Details+Projects to enable them.

- **Expense**

  According to the iBiz manual, this type indicates that *"the job event represents a cost that you paid on behalf of the client, which you are passing on to the client as part of their bill*".

  Fanurio uses expenses to record money that you spend for a project. If your projects don't have an expenses list, go to Business » My Business Details+Projects to enable them. Fanurio also allows you to mark up an expense.

- **Non-billable**

  According to the iBiz manual, this type indicates that "*the job event should not be included on estimates and invoices*" if you "*want to track expenses without passing them on to the client, or for logging extra time that is not meant to be billed*".

  Tasks, expenses, and trips can be marked as non-billable in Fanurio. Products are always billable and can't be marked as non-billable. If a project is not billable, you can mark it as non-billable and all its elements will be non-billable.

- **Deduction**

  According to the iBiz manual, this type is "*applied automatically when a job event is created with a negative rate*".

  If you want offer a discount for your services then the best solution is to apply a discount to your tasks. For any other deductions, you can create a product with a negative price.

3.13.2.4. Job event groups

Fanurio doesn't have an equivalent for job event groups. It's not possible to group tasks, expenses, trips, nor products but you can use tags to organize tasks, expenses and trips. You can also use categories to organize tasks and expenses.

However, if you used job event groups to define major activities of a project and timed job events as the specific times when you worked on those activities then you can get the same result if you use tasks (instead of job event groups) and time entries (instead of job events). For instance, if you have a job event group called Brochure with three job events called Day 1, Day 2, and Day 3 then you can create a task called Brochure and add three time entries (Day 1, Day 2, and Day 3).

3.13.2.5. Custom job events

In iBiz, custom job events allow you to create job events with the same settings faster.

In Fanurio, you can create tasks faster if you define one or more task categories. The list of task categories can be managed from Business » My Business Details+Projects+Tasks. Also, you can create products faster if you define one or more catalog items. The list of catalog items can be managed from Business » My Business Details+Billing+Catalog.

3.13.2.6. To Do items

iBiz uses To Do items to track activities that don't need to be tracked as job events. Compared to job events, To Do items have some specific fields like due date, priority and they can be marked as completed. To Do items can also be synced with iCal.

To track the same type of activities in Fanurio, you should use non-billable tasks. Tasks allow you to set a due date and they can be marked as completed when they're done. Just like iBiz shows completed To Do items in gray so does Fanurio.

If you need to see your tasks in Apple Reminders, you can export your projects to the iCalendar format.

3.13.2.7. Timers

iBiz has two types of timers: job event timers and quick timers. A job event timer can only be started for an existing timed job event and its time is added to the job event as the timer is running. A quick timer on the other hand doesn't require a job event and time is added to a job event when the timer is stopped.

Fanurio has only one type of timers which is similar to the iBiz quick timers. When you start a timer in Fanurio you can start it for a specific task or for no specific task. When the timer is stopped, Fanurio creates a time entry and adds it to the task. Time entries are similar to iBiz timer logs except that they can also be added manually to a task. They only way you can record time on a task is by adding one or more time entries to it.

Fanurio allows you to access its timer from the menu bar, the menu bar icon and from the toolbar.

Unlike iBiz, Fanurio doesn't allow multiple timers to run simultaneously. There can be only one timer running at a time. iBiz works the same way if the "Starting a timer stops all other timers" setting is enabled from Preferences.

In Fanurio, time rounding is not a timer feature but a feature available only on billable tasks. Fanurio can handle complex time rounding rules like "round time to the nearest 15 minutes interval but bill at least 30 minutes".

Fanurio can detect idle time just like iBiz can but it also has a few other reminders to help you start, stop and resume timers.

To learn more about how you can record time in Fanurio (manually or using a timer), see the time entries section.

3.13.2.8. File Cabinet

Fanurio doesn't have an equivalent for File Cabinet.

3.13.2.9. Estimates

Fanurio doesn't have built-in support for estimates but there is something you can do to create one until this feature is implemented.

1. Create two projects, one for the estimate and the other for the actual work.
2. Add one or more products to the estimate project.
3. Click the estimate project to display the contextual menu and select New Invoice.
4. Add all the products to the invoice.
5. Instead of clicking Create to create a new Invoice, click Preview. This allows you to export the temporary invoice and use it as an estimate. You will also need a slightly modified invoice template that says Estimate instead of Invoice to export this invoice.

3.13.2.10. Invoices

Fanurio can create project invoices like iBiz but it can also create regular non-project invoices. Project invoices can bill one or more projects and they can include project elements from a specific date range.

Fanurio can export invoices to HTML, PDF, Microsoft Word 2007, OpenDocument and other formats so they can be printed or e-mailed. Invoice templates can be created manually, with a visual editor (Adobe Dreamweaver, Microsoft Word or LibreOffice) or with the built-in template editor.

Fanurio doesn't post the due date of an invoice to iCal. Overdue invoices are displayed in red in the Invoices view.

3.13.2.11. Invoice Reminders

Fanurio doesn't have invoice reminders. Overdue invoices are displayed in red in the Invoices view. You can export an invoice again at any time if you need to resend it to your client.

3.13.2.12. Statements

iBiz uses statements to show the details of a billing account (invoices sent, payments received, and any remaining balance due). Statements can also have late fees (flat or percentage-based) associated with them.

Fanurio doesn't have a separate concept for statements but you can include one with every invoice that you send to your customers. The invoice template editor allows you to show a statement at the bottom of any invoice. If you need to charge for late fees, you need to create a regular non-project invoice with one item for the late fee.

3.13.2.13. Taxes

iBiz can use up to two taxes on its job events and each job event can have its own taxes. iBiz also allows you to configure how taxes are calculated: on each job event or on their sum. There's also an option of compounding taxes.

Fanurio can handle taxes as well but it does it a little differently. First of all, taxes are optional and have to be enabled at business level. This saves you from seeing tax information if you're not using taxes. But if you need to use taxes, Fanurio can use three or more taxes (compounded or not).

In Fanurio, taxes are not specified at item level but only at invoice level. If you have items that need to be taxed differently, you need to create more than one invoice. The tax total is calculated by summing up the tax total of each item.

3.13.2.14. Payments

iBiz allows you to record payments in your clients' billing accounts. You can also split a payment across several invoices and/or payments.

Fanurio records payments per invoice and they can't be split. Also, all billable clients have a deposits account. You can use this account to record money you receive in advance for your work.

In order to make Fanurio work like iBiz, you should record your payments as deposits. Then you can use money from the deposit account to pay the invoices.

3.13.2.15. Templates

iBiz can handle two file formats for its templates: RTF and HTML. Fanurio doesn't support RTF but it supports HTML and many other formats like Microsoft Word and OpenDocument Text (LibreOffice, OpenOffice).

Fanurio doesn't migrate iBiz templates but if you are an expert user, you can convert them relatively easy. The template syntax is similar. A list of all placeholders supported by Fanurio (or tags in iBiz terminology) is documented in this manual. To learn more about templates, see the templates guide.

Fanurio comes with a default invoice template that can be configured using the template editor.

3.13.2.16. Reports

There are two ways you can create reports in Fanurio:

1. Since most views are table-based, you can use the filters above the table from each view to see only certain things. For instance, in the Timesheet view you can use the Date filter to see what you did last month. Then you can ctrl-click the table to display the contextual menu and select the Export action to export the table to CSV or Microsoft Excel.

2. You can also create template-based reports for projects, time, tasks, expenses, trips and sales.

3.13.2.17. Document Monitor

Fanurio doesn't have an equivalent for the iBiz document monitor.

3.13.2.18. The workday pane

Fanurio doesn't have an equivalent for the workday pane

3.13.2.19. iBank integration

Fanurio doesn't integrate with iBank.

3.13.2.20. Applescript and Automator integration

Fanurio can't be used with Applescript or Automator.

3.13.2.21. Networking

Fanurio can't handle multiple users yet. A multi-user version is currently under development.

If you are just one user who needs to use Fanurio on multiple computers, see this section for a few solutions (we recommend solution B. Shared folder).

3.13.2.22. Platforms

iBiz 4 requires Mac OS X 10.5 or higher.

Fanurio runs on Mac OS X 10.5 or higher, Windows and Linux.

3.14. Keyboard shortcuts

The following keyboard shortcuts can only be used from within the application. For a list of keyboard shortcuts that can be used from within **any running application**, please see the global hotkeys section.

Table 3.9. Keyboard shortcuts for the Windows version

Key	Action
F1	Open help
Alt-Enter	Edit the selected record from a table
Delete	Delete the selected record from a table
F4	Edit the active timer
F5	Start new timer immediately
Shift-F5	Start new timer
F6	Pause/Resume timer immediately
Shift-F6	Pause/Resume timer
F7	Stop timer
F9	Synchronize (available in client mode)
Ctrl-C	Create a new client
Ctrl-P	Create a new project
Ctrl-K	Create a new task
Ctrl-T	Create a new time entry
Ctrl-E	Create a new expense
Ctrl-R	Create a new trip
Ctrl-U	Create a new product
Ctrl-V	Create a new invoice
Ctrl-A	Add a new payment
Ctrl-Shift-M	Switch between the iTunes-like mini timer and the main window
Ctrl-Shift-P	Switch to Projects View
Ctrl-Shift-K	Switch to Tasks View
Ctrl-Shift-T	Switch to Timesheet View
Ctrl-Shift-E	Switch to Expenses View
Ctrl-Shift-R	Switch to Trips View
Ctrl-Shift-V	Switch to Invoices View
Ctrl-Shift-A	Switch to Payments View

Table 3.10. Keyboard shortcuts for the Mac OS X version

Key	Action
Command-I	Edit the selected record from a table
Delete	Delete the selected record from a table
F4	Edit the active timer
F5	Start new timer immediately
Shift-F5	Start new timer

Key	Action
F6	Pause/Resume timer
F7	Stop timer
F9	Synchronize (available in client mode)
Command-C	Create a new client
Command-P	Create a new project
Command-K	Create a new task
Command-T	Create a new time entry
Command-E	Create a new expense
Command-R	Create a new trip
Command-U	Create a new product
Command-V	Create a new invoice
Command-A	Add a new payment
Command-M	Minimize window
Command-Q	Quit the application
Command-Shift-M	Switch between the iTunes-like mini timer and the main window
Command-Shift-P	Switch to Projects View
Command-Shift-K	Switch to Tasks View
Command-Shift-T	Switch to Timesheet View
Command-Shift-E	Switch to Expenses View
Command-Shift-R	Switch to Trips View
Command-Shift-V	Switch to Invoices View
Command-Shift-A	Switch to Payments View

Table 3.11. Keyboard shortcuts for the Linux version

Key	Action
F1	Open help
Alt-Enter	Edit the selected record from a table
Delete	Delete the selected record from a table
F4	Edit the active timer
F5	Start new timer immediately
Shift-F5	Start new timer
F6	Pause/Resume timer
F7	Stop timer
F9	Synchronize (available in client mode)
Ctrl-C	Create a new client
Ctrl-P	Create a new project
Ctrl-K	Create a new task
Ctrl-T	Create a new time entry
Ctrl-E	Create a new expense
Ctrl-R	Create a new trip

Key	Action
Ctrl-U	Create a new product
Ctrl-V	Create a new invoice
Ctrl-A	Add a new payment
Ctrl-Shift-M	Switch between the iTunes-like mini timer and the main window
Ctrl-Shift-P	Switch to Projects View
Ctrl-Shift-K	Switch to Tasks View
Ctrl-Shift-T	Switch to Timesheet View
Ctrl-Shift-E	Switch to Expenses View
Ctrl-Shift-R	Switch to Trips View
Ctrl-Shift-V	Switch to Invoices View
Ctrl-Shift-A	Switch to Payments View
Ctrl-Q	Quit the application

Chapter 4. Templates guide: How to create and edit templates for Fanurio

This guide shows you how to create or edit templates that you can use to export documents like invoices.

4.1. Getting started

This guide shows you how to create a template from scratch but in most cases, you won't have to do that. Almost any template can be created a lot easier using the template editor. Even our support team prefers to use the editor instead of coding templates by hand.

Fanurio can handle templates in several file formats like HTML, Microsoft Word 2007 or OpenDocument Text. However, HTML is the recommended format for templates in Fanurio. The template editor also creates HTML templates. You can find a list of all supported file formats with comments on each one of them at the end of this guide.

If you create templates in HTML or any other format, it's important that you read the template language section and the section on how to use directives in tables.

4.1.1. Creating an invoice template in 10 minutes or less

The most recommended way of creating an invoice template is to use the template editor. Even power users can do a lot of tweaks just from the editor.

Here's how to create your first template:

1. Go to File » Template Editor to open the template editor.

2. Go through all the sections of the editor and check the fields you want to include in your template. You can configure things like your logo, the visible columns, the totals or the names of the columns.

   To see how it looks like after you make some changes, click Update or press F5 (Windows), Ctrl +R (Linux) or Cmd+R (Mac OS X).

3. Click Save to save the template to the templates folder when you're done. Later, when you'll want to export an invoice to HTML, you can use this template.

If you want to make further adjustments to the template, use the template editor and click the Open button to open it. To learn more about the template editor, make sure you read the section that describes it.

If you have problems customizing or creating a template, you can always contact us for help.

4.1.2. Installing a template

Installing a template is just a matter of copying one or more files to a certain folder. Go to File » Open Templates Folder to open the folder where you want to install the template. For invoice templates, use File » Open Templates Folder » Invoices.

If you are using an HTML template that uses specific fonts and images, make sure you also copy those files to the templates folder.

4.1.3. About Freemarker

Fanurio uses the FreeMarker template language [http://freemarker.sourceforge.net/] for its templates. Freemarker is a powerful and mature template language that can be used to create both simple and complex templates. FreeMarker can be used to generate any kind of text: HTML, XML, RTF, etc.

This guide doesn't cover all FreeMarker concepts but only those that are absolutely necessary. For a detailed guide on the FreeMarker language, go to http://freemarker.sourceforge.net/docs/dgui.html.

4.2. Using the template editor

The template editor is recommended for both beginners and advanced users who want to create an HTML invoice template. Our support team prefers to use the editor rather than to code templates by hand.

The template editor can be used to create new templates or to edit existing ones.

- Go to File » Template Editor to open the template editor.
- If you've already created a template, go to the Invoices view and double-click an invoice to view it. You can then click the Edit link next to the Templates drop-down list to edit the currently used template.

  The nice thing about opening the template editor from the View Invoice window is that you can adjust the template and see how that specific invoice will look like.

The editor works with HTML templates but you don't have to know HTML to use it, although this might help if you want to adjust even the smallest details. The templates created by the template editor should not be edited outside Fanurio unless you don't want to open them again in the template editor.

Note: The template editor can only open templates it created. It cannot import old HTML templates or templates created manually. If you have an old template, it should be relatively easy to recreate it with this editor.

4.2.1. Basic settings

Most template settings can be easily changed by checking some boxes or by typing text in some fields. That's how you can specify the page format, the logo, the columns of the invoice and their names or the totals.

Headers and footers on the other hand are more complex and they may contain HTML code and placeholders for specific invoice fields.

Example 1: Changing a simple setting (the logo)

1. Go to the Page section,
2. Check the Logo box,
3. Click the link next to the Logo box to specify the logo image,
4. Click the Update button to see how it looks like and
5. Optionally, adjust the Height and click Update again to see how the look changes.

Example 2: Using placeholders (showing the invoice notes at the bottom of the invoice)

1. Go to the Invoice page,
2. Scroll down to the Footer text area,
3. Type or paste the following text:

   ```
   ${invoice.notes}
   ```

4. Click the Update button to see the notes at the bottom of the invoice.

You need to use ${invoice.notes} in order to access the notes of the invoice. Just like **invoice.notes**, there are many other placeholders for the fields of an invoice, your business and your client's business.

The complete list of placeholders can be found here. If you want to know more about placeholders, we have a separate section that explains them.

Example 3: Using HTML to do simple formatting (showing a thank you note at the bottom of the invoice)

1. Go to the Invoice page,
2. Scroll down to the Footer text area,
3. Type or paste the following text:

```
<p>${invoice.notes}</p>
<br/>
<p><center>Thank you for your business!</center></p>
```

4. Click the Update button to see the notes at the bottom of the invoice.

This footer contains HTML code to format the text. The text between ** ** and ** ** represents a paragraph, **
** represents a line-break while the text between ** ** and ** ** will be displayed centered. Another tag that you may want to use to format text as bold is **** ****.

See this guide [http://www.w3schools.com/html/html_intro.asp] if you want to learn HTML at basic level.

4.2.2. Adjust the look and format using CSS (for advanced users only!)

The template editor has a CSS field in the Style section where users can enter custom CSS code to adjust the look and format of the final document. This feature is meant to be used only by people who know CSS.

Here are a few snippets that you can use to customize the invoice template.

Snippet 1: Display totals only on the last page

```
.table tfoot {
  display: table-row-group;
}
```

If the invoice has multiple pages, totals are displayed on each page. This code changes the table to display the totals only on the last page.

Snippet 2: Force the invoice table to a certain height

```
.table {
  height: 12cm;
}
```

The invoice table height is variable and depends on what's in the table. This code forces it to have a specific height.

Snippet 3: Paint a line above the table footer (totals)

```
.table tfoot tr:first-child td {
  border-top: 1px solid black;
}
```

Use this code if you want to separate the totals from the rest of the table by drawing a line.

4.3. The anatomy of a simple template

This section introduces all the concepts you must know to create or edit templates. You can use the ideas presented here to build any type of templates: Microsoft Word 2007, OpenDocument, HTML, XML, etc. To keep things simple, the templates from this section are plain text files.

When dealing with templates, there are three important concepts you need to know:

- **placeholders** help you access fields like invoice number or invoice total,
- **built-ins** can give more details about certain fields like the text representation for a date field and
- **directives** which allow you to perform operations on fields like deciding what to do if an invoice has or doesn't have expenses.

Although templates can be used for several types of documents, we'll only focus on invoices here. We'll build an invoice template step by step to show all the concepts. The final template is in the file text-invoice.txt [files/templates/txt/text-invoice.txt].

4.3.1. Placeholders or how to access fields

Let's suppose you create an invoice that has the number INV-23 and a total of USD 100. A possible layout for the invoice document could be:

```
INVOICE

Number: INV-23
Total: USD 100

Thank you for your business!
```

The number and total are emphasized to show that they are different for each invoice.

Fanurio can help you create invoices that have the same layout but it needs to know which parts of the document change and which don't. It does this with the help of a template.

Here's how the template for the above invoice looks like. It is similar to the actual invoice but the changing parts **INV-23** and **USD 100** have been replaced by **${invoice.number}** and **${invoice.total}**.

```
INVOICE

Number: ${invoice.number}
Total: ${invoice.total}

Thank you for your business!
```

When Fanurio exports an invoice using a template, it identifies all texts of the form **${...}** as changing parts and replaces them with actual values. If the invoice number is a changing part, you use **invoice.number** inside **${...}**. The text **invoice.number** is a placeholder for the actual invoice number.

Just like **invoice.number**, there are many other placeholders for the fields of an invoice, your business and your client's business. The complete list of placeholders can be found here.

This template is saved in the file text-invoice-01.txt [files/templates/txt/text-invoice-01.txt].

Extra: You may also want to try other placeholders like your business contact details:

```
Supplier: ${business.name}
Address: ${business.address}
```

```
INVOICE

Number: ${invoice.number}
Total: ${invoice.total}

Thank you for your business!
```

4.3.2. Built-ins or how to get more details about a field

Usually, when you want to get the value of a placeholder, you put it inside **${..}**. For some placeholders, you can also use built-ins to get additional details. A list of all built-ins can be found here [http://freemarker.sourceforge.net/docs/ref_builtins.html].

For instance, date placeholders can use the **?date** built-in to display them using the default format. The **invoice.date** placeholder represents the date when the invoice was issued. To access its value, you should use **${invoice.date?date}**.

If we want to display the invoice date, the previous template changes to:

```
INVOICE

Number: ${invoice.number}
Date: ${invoice.date?date}

Total: ${invoice.total}

Thank you for your business!
```

This template is saved in the file text-invoice-02.txt [files/templates/txt/text-invoice-02.txt].

Extra: Another built-in that can be used with dates is **?string**. The following code displays the date in a user-specified format:

```
${invoice.date?string("yyyy-MM-dd")}
```

where **yyyy** is the year, **MM** is the month and **dd** is the day. Of course, you can change their order however you like.

4.3.3. Directives or how to perform operations on fields

Placeholders are usually not enough if you want to create an invoice template. You will also need to perform various operations on placeholders. We call these operations directives.

Some common directives are:

- list
- if

The *list* directive allows you to access placeholders with multiple values while *if* lets you make decisions based on the value of a placeholder. A list of all directives can be found here [http://freemarker.sourceforge.net/docs/ref_directives.html].

The list directive

The placeholders we've seen so far (invoice.number, invoice.total, invoice.date) are single-valued but there are placeholders that contain multiple values. The values of these placeholders cannot be simply accessed by surrounding them with **${...}**, you have to access them element by element.

Let's suppose you want to display all the items of an invoice like this:

```
INVOICE

Number: INV-23
Date: Jan 01, 2001

Service1 5 x $10 = $50
Service2 2 x $5 = $10
Product1 1 x $10 = $10
Expense1 1 x $40 = $30

Total: USD 100

Thank you for your business!
```

where each emphasized line represents an item as:

```
name quantity x unit price = total
```

The following template shows how you can use the list directive to display each invoice item on a separate line.

```
INVOICE

Number: ${invoice.number}
Date: ${invoice.date?date}

[#list invoice.items as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]

Total: ${invoice.total}

Thank you for your business!
```

The **invoice.items** placeholder contains a list with all the items of an invoice. To access each item from this list you need to use the list directive. The list directive is always of the form:

```
[#list multi-value-placeholder as element]
access the element
[/#list]
```

In our example, the multi-value placeholder is *invoice.items*.

This template is saved in the file text-invoice-03.txt [files/templates/txt/text-invoice-03.txt].

The if directive

Suppose that instead of displaying all items together, you want to group them by their type. One group for service items, one for product items and the another for expense items. The previous list:

```
Service1 5 x $10 = $50
Service2 2 x $5 = $10
Expense1 1 x $10 = $10
Expense1 1 x $30 = $30
```

becomes:

```
Services
```

```
Service1 5 x $10 = $50
Service2 2 x $5 = $10

Products
Product1 1 x $10 = $10

Expenses
Expense1 1 x $30 = $30
```

The template that makes this possible looks like this:

```
Services
[#list invoice.serviceItems as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]

Products
[#list invoice.productItems as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]

Expenses
[#list invoice.expenseItems as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]
```

As you can see, we use not one but three list directives. One for the *invoice.serviceItems* placeholder, another for the *invoice.productItems* placeholder and the another one for the *invoice.expenseItems* placeholder. These three placeholders help us access items by their type (service, product or expense).

Let's take this example even further and assume that the list of expense items is empty. In that case, the invoice is:

```
Services
Service1 5 x $10 = $50
Service2 2 x $5 = $10

Products
Product1 1 x $10 = $10

Expenses
```

Since there are no expense items, you'll probably want to hide the "Expenses" text too. In other words, you want to display it only if there is at least one expense item. This can be done using the if directive as follows:

```
Services
[#list invoice.serviceItems as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]

Products
[#list invoice.productItems as item]
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]

[#if invoice.expenseItems?size != 0]
Expenses
[#list invoice.expenseItems as item]
```

```
${item.name} ${item.quantity} x ${item.price} = ${item.total}
[/#list]
[/#if]
```

The if directive wraps the text that is displayed conditionally. The text **invoice.expenseItems?size != 0** is the condition that is interpreted as "number of expense items is different from zero". This condition evaluates to:

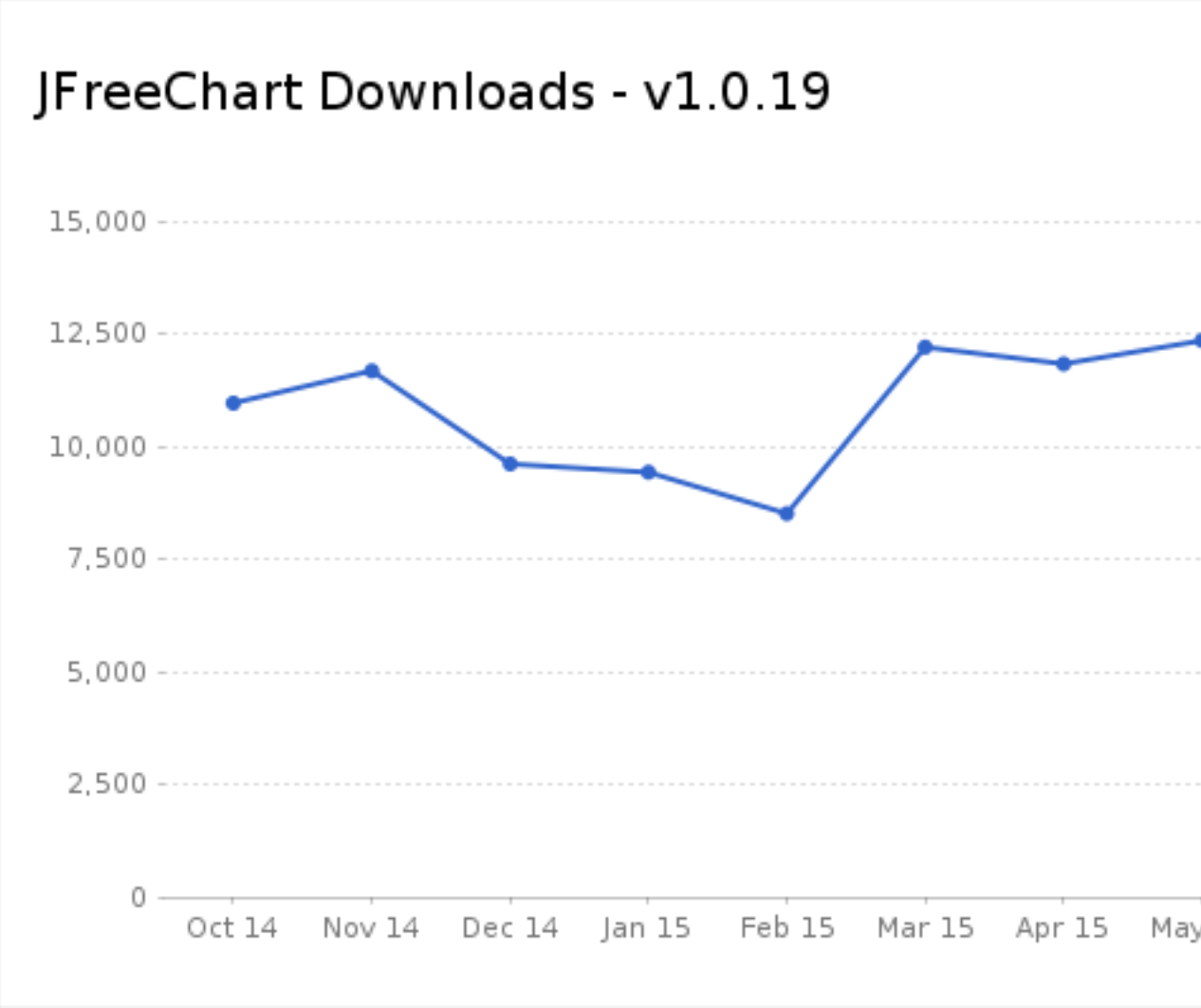
- **true** if there is at least one expense item. In this case, the Expenses text together with all expense items are displayed.
- **false** if there are no expense items. In this case, nothing is displayed.

Notice the **?size** built-in. It applies to multi-value placeholders and retrieves the number of elements.

This template is saved in the file text-invoice-04.txt [files/templates/txt/text-invoice-04.txt].

4.4. Charts (for advanced users)

Fanurio uses the JFreeChart [http://www.jfree.org/jfreechart/] library for its charts. JFreeChart can create professional quality charts and it's the most widely used chart library for Java.



Although this guide has lots of information and examples to help you create great looking charts using Freemarker, if you need to design pixel-perfect charts then you should be familiar with the JFreeChart API. This requires at least basic programming skills. The JFreeChart Developer Guide [http://www.jfree.org/jfreechart/devguide.html] is the reference document for the JFreeChart API [http://www.jfree.org/jfreechart/api.html].

4.4.1. The jfreechart directive

Fanurio has a custom Freemarker directive that allows you to create JFreeChart charts. The jfreechart directive generates a chart as an inline PNG image [https://en.wikipedia.org/wiki/Data_URI_scheme].

Note: Although the jfreechart directive can be used in any Freemarker template, it's very likely you will be using it in templates that create HTML documents. One way to include inline images in an HTML document is to add them in the **src** attribute of an **img** tag as shown in the following examples.

The jfreechart directive has the following parameters that allow you to specify the content and the looks of the chart it creates.

- **Type**

The type parameter is mandatory and it specifies the kind of chart you want to create. The following example uses the 'ring' type to create a ring chart but other types are supported as well. Ring charts (also called doughnut or donut charts) are a special type of pie charts. All chart types are documented in the next section.

```
<img src="[@jfreechart type='ring' /]"/>
```

- **Dataset**

The dataset parameter allows you to specify the data that will be represented by the chart. Different types of charts use different formats for their data. For ring charts, the dataset parameter is a list of two-element sublists.

The following example uses a dataset to represent the time recorded on a project and the estimated remaining time.

```
<img src="[@jfreechart type='ring' dataset=[['Time',  39], ['Remaining', 11]]
```

This dataset has only two values but datasets with multiple values can be quite long. In such cases, datasets can be defined separately as shown in the following example. Please note that the last element in the list is not followed by a comma.

```
[#assign dataSet = [
    ['Time',  39],
    ['Remaining', 11]
] /]

<img src="[@jfreechart type='ring' dataset=dataSet /]"/>
```

Note: Charts that can represent data on multiple axes need to use multiple datasets instead of a single dataset. For these charts, data is specified using the datasets parameter instead of dataset. The datasets parameter is documented for each chart that supports it. See this chart [111] for an example of how to use multiple datasets.

- **Properties**

The properties parameter allows you to control the looks of the chart. Properties must be represented as a list of name and value pairs. All chart properties are documented in the next section.

The following example shows a few properties that can be used with a ring chart.

```
[#assign pieDataSet = [
    ['Time',  39],
    ['Remaining', 11]
] /]

<img src="[@jfreechart type='ring' dataset=dataSet properties={} /]"/>
```

Just like the dataset, the list of properties can be very long so it's better if you define it separately.

```
[#assign dataSet = [
    ['Time',  39],
    ['Remaining', 11]
] /]

[#assign chartProperties = {
    'legend.visible': false,

    'plot.backgroundPaint': '',
```

```
    'plot.outlineVisible': false,
    'plot.shadowPaint': '',

    'plot.sectionOutlineStroke': [{'width': 2.0}, {'width': 0.0}],
    'plot.sectionOutlinePaint': ['#87b80e'],
    'plot.sectionPaint':['#a6cb4a', '#d7d7d7'],

    'plot.labelGenerator': '',

    'plot.centerTextMode': 'fixed',
    'plot.centerText': '39 / 50',
    'plot.centerTextFont': {'size': 36},
    'plot.centerTextColor': '#555555',

    'plot.sectionDepth': 0.08,
    'plot.separatorsVisible': false
} /]

<img src="[@jfreechart type='ring' dataset=dataSet properties=chartProperties
```

- **Width and height**

  Finally, the last two parameters you can use with the jfreechart directive are width and height. These parameters specify the size of the chart image. By default, the width and height are 400 pixels.

```
[#assign dataSet = [
    ['Time',  39],
    ['Remaining', 11]
] /]

[#assign chartProperties = {
    'legend.visible': false,

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.shadowPaint': '',
```

```
    'plot.sectionOutlineStroke': [{'width': 2.0}, {'width': 0.0}],
    'plot.sectionOutlinePaint': ['#87b80e'],
    'plot.sectionPaint':['#a6cb4a', '#d7d7d7'],

    'plot.labelGenerator': '',

    'plot.centerTextMode': 'fixed',
    'plot.centerText': '39 / 50',
    'plot.centerTextFont': {'size': 36},
    'plot.centerTextColor': '#555555',

    'plot.sectionDepth': 0.08,
    'plot.separatorsVisible': false
} /]

<img src="[@jfreechart type='ring' dataset=dataSet properties=chartProperties
```

4.4.2. Chart types

Although JFreeChart can create many types of charts, only the types documented in this section are supported by the jfreechart directive. More types may be supported in the future.

If you haven't used charts before, these resources explain them in more detail:

- Data 101: The Basics You Need to Know [http://visage.co/data-101-refresher-course/]
- How to Use 6 Basic Charts to Create Effective Reports [https://fluidsurveys.com/university/use-different-chart-types/]
- Practical Rules for Using Color in Charts [http://www.perceptualedge.com/articles/visual_business_intelligence/rules_for_using_color.pdf]
- Best Practices: Maximum Elements For Different Visualization Types [http://blog.visual.ly/maximum-elements-for-visualization-types/]
- A Periodic Table of Visualization Methods [http://www.visual-literacy.org/periodic_table/periodic_table.html]

4.4.2.1. Area charts

An area chart is a line chart with the areas below the lines filled with colors to indicate volume. Area charts are most often used to visualize data that changes over time. They are perfect for communicating the overall trend.

If you haven't used area charts before, these resources explain them in more detail:

- Data Visualization 101: Area Charts [http://visage.co/data-visualization-101-area-charts/]
- The Fine Line In a Gray Area: When to Use Line vs Area Charts [http://blog.visual.ly/line-vs-area-charts/]
- I hate stacked area charts [http://www.leancrew.com/all-this/2011/11/i-hate-stacked-area-charts/]

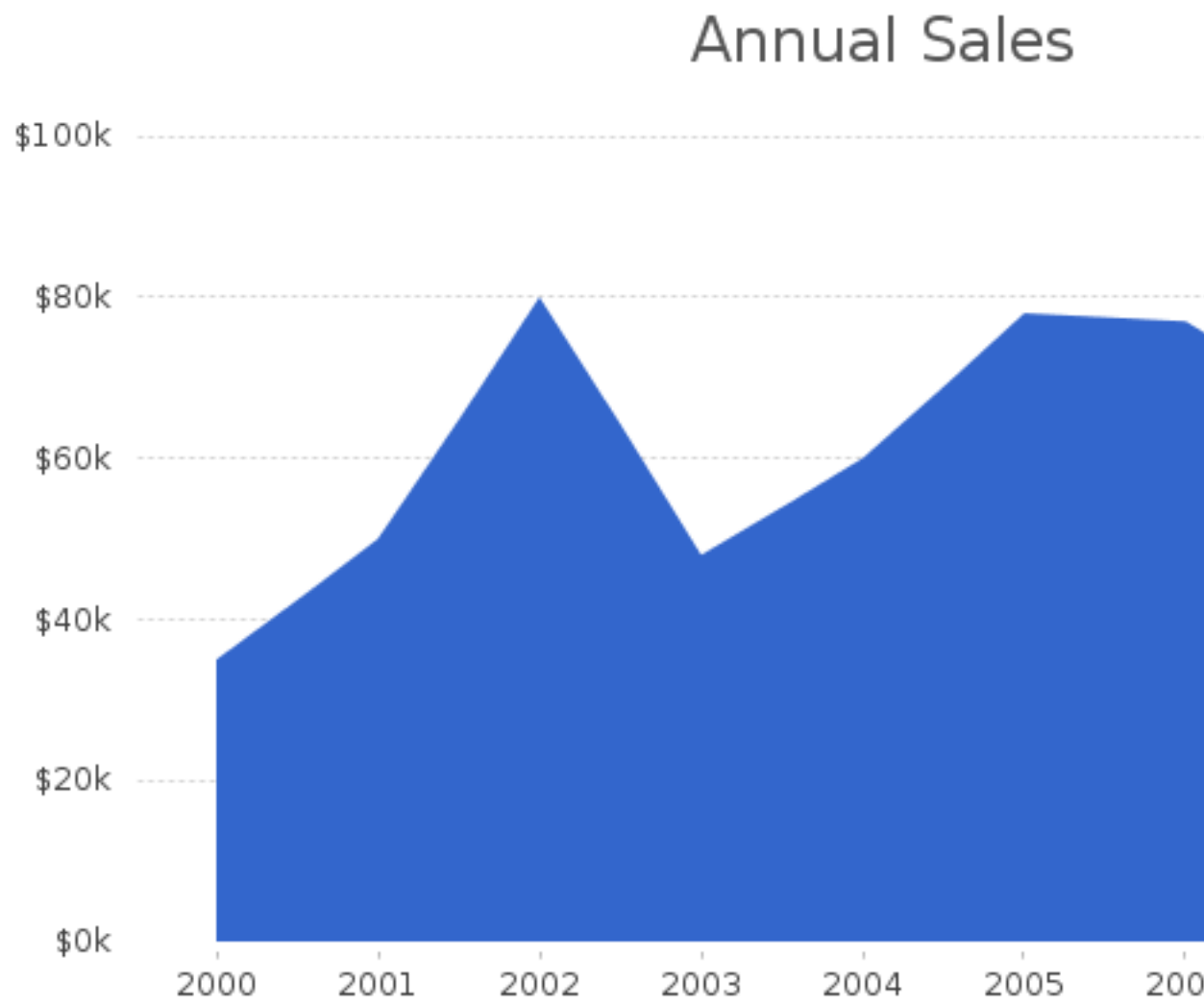
4.4.2.1.1. Category area charts

In order to create an area chart that uses texts (categories) on the domain axis, the jfreechart directive must be used with the following parameters:

Table 4.1. The jfreechart directive parameters for category area charts

Parameter	Value
type	area
dataset	A sequence of the following form: [[**Heading , Series$_1$, ... Series$_n$**], [**xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$**], ... [**xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$**]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a series. • xValue$_i$ is a text that represents a value (category) on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset$_1$, dataset$_2$, ... dataset$_n$**] where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, category plot • renderer: renderer, category item renderer, area renderer • domain axis: axis, category axis • range axis: axis, value axis, number axis

The following example creates an area chart that shows the annual sales of a fictitious company over several years.




```
[#assign dataSet = [
  ['Year', 'Annual Sales'],
  ['2000', 35],
  ['2001', 50],
  ['2002', 80],
  ['2003', 48],
  ['2004', 60],
  ['2005', 78],
  ['2006', 77],
  ['2007', 65],
  ['2008', 50],
  ['2009', 25]
] /]

[#assign chartProperties = {
    'title': 'Annual Sales',
    'title.horizontalAlignment': 'center',
    'title.font': {'size': 24},
    'title.padding': [5, 0, 15, 0],
    'title.paint': '#555555',

    'legend.visible': false,
```

```
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,

    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#aaaaaa',
    'plot.domainGridlinesVisible': false,

    'plot.domainAxis.axisLineVisible': false,
    'plot.domainAxis.lowerMargin': 0.00,
    'plot.domainAxis.upperMargin': 0.00,

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.numberFormatOverride': '${"$"}#0k',
    'plot.rangeAxis.lowerMargin': 0.00,
    'plot.rangeAxis.upperMargin': 0.00,
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 20},
    'plot.rangeAxis.range': [0, 100],

    'plot.renderer.endType': 'truncate'
} /]

<img src="[@jfreechart type='area' dataset=dataSet properties=chartProperties w
```

4.4.2.1.2. XY area charts

In order to create an area chart that uses numbers on the domain axis, the jfreechart directive must be used with the following parameters:

Table 4.2. The jfreechart directive parameters for xy area charts

Parameter	Value
type	xyarea
dataset	A sequence of the following form: [[**Heading** , **Series$_1$** , ... **Series$_n$**], [**xValue$_1$** , **yValue$_{11}$**, ... **yValue$_{n1}$**], ... [**xValue$_k$** , **yValue$_{1k}$**, ... **yValue$_{nk}$**]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a series. • xValue$_i$ is a number that represents a value on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset$_1$**, **dataset$_2$**, ... **dataset$_n$**]

Parameter	Value
	where $dataset_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, xy plot • renderer: renderer, xy item renderer, xy area renderer • domain axis: axis, value axis, number axis • range axis: axis, value axis, number axis

The following example shows the course elevation chart of the Boston Marathon [http://www.boston.com/sports/marathon/course/elevations/]. The Boston Marathon [https://en.wikipedia.org/wiki/Boston_Marathon] is an annual marathon hosted by several cities in greater Boston in eastern Massachusetts, United States.

```
[#-- http://www.boston.com/sports/marathon/course/elevations/ --]
[#assign dataSet = [
  ['Mile', 'Elevation in feet'],
  [ 0.0, 490],
  [ 1.0, 360],
  [ 2.0, 320],
  [ 3.0, 265],
```

```
  [ 4.0, 180],
  [ 5.0, 205],
  [ 6.0, 180],
  [ 7.0, 155],
  [ 8.0, 180],
  [ 9.0, 150],
  [10.0, 170],
  [11.0, 180],
  [12.0, 165],
  [13.0, 145],
  [14.0, 130],
  [15.0, 160],
  [16.0,  60],
  [17.0, 115],
  [18.0, 145],
  [19.0, 130],
  [20.0, 150],
  [21.0, 230],
  [22.0, 150],
  [23.0,  95],
  [24.0,  60],
  [25.0,  15],
  [26.0,  10],
  [26.2,  10]
] /]

[#assign chartProperties = {
    'title': 'Boston Marathon Course Elevations',
    'title.horizontalAlignment': 'center',
    'title.font': {'size': 24},
    'title.padding': [5, 0, 15, 0],
    'title.paint': '#555555',

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.rangeGridlinesVisible': false,
    'plot.domainGridlinesVisible': false,
    'plot.foregroundAlpha': 1.0,

    'plot.renderer.seriesPaint': ['rgba(166, 203, 74, 0.75)'],
    'plot.renderer.seriesOutlinePaint': ['#87b80e'],
    'plot.renderer.outline': true,

    'plot.domainAxis.axisLineVisible': false,
    'plot.domainAxis.label': 'Distance (mi)',
    'plot.domainAxis.labelPaint': '#555555',
    'plot.domainAxis.labelFont': {'size': 12},
    'plot.domainAxis.tickLabelFont': {'size': 12},
    'plot.domainAxis.tickLabelPaint': '#555555',
    'plot.domainAxis.tickUnit': {'size': 2},

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.label': 'Elevation (ft)',
    'plot.rangeAxis.labelPaint': '#555555',
    'plot.rangeAxis.labelFont': {'size': 12},
    'plot.rangeAxis.tickLabelFont': {'size': 12},
    'plot.rangeAxis.tickLabelPaint': '#555555',
```

```
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 100},

    'legend.visible': false
} /]

<img src="[@jfreechart type='xyarea' dataset=dataSet properties=chartProperties
```

4.4.2.1.3. Time series area charts

In order to create an area chart that uses dates/times on the domain axis, the jfreechart directive must be used with the following parameters:

Table 4.3. The jfreechart directive parameters for time series area charts

Parameter	Value
type	timeseriesarea

dataset	A sequence of the following form:
	
	[
	[**Heading , Series$_1$, ... Series$_n$**],
	[**xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$**],
	...
	[**xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$**]
	]
	
	where:
	
	• Heading can be any text that describes the values on the domain axis.
	
	• Series$_i$ is a text that represents the name of a series.
	
	• xValue$_i$ is a unit of time that represents a value on the domain axis.
	
	JFreeChart represents units of time as RegularTimePeriod [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/time/RegularTimePeriod.html] objects (FixedMillisecond, Millisecond, Minute, Hour, Second, Day, Week, Month, Quarter, Year). To create such objects in Freemarker, you need to use the new built-in [http://freemarker.org/docs/ref_builtins_expert.html#ref_builtin_new] and instantiate specific wrappers. Here are a few examples:
	
	`${'com.fanurio.common.freemarker.SimpleYear'`
	`${'com.fanurio.common.freemarker.SimpleMonth`
	
	However, if you specify a unit of time as a Freemarker date object, it will be interpreted as a Day unit of time. Any other date object will be interpreted as a Millisecond unit of time. For instance:

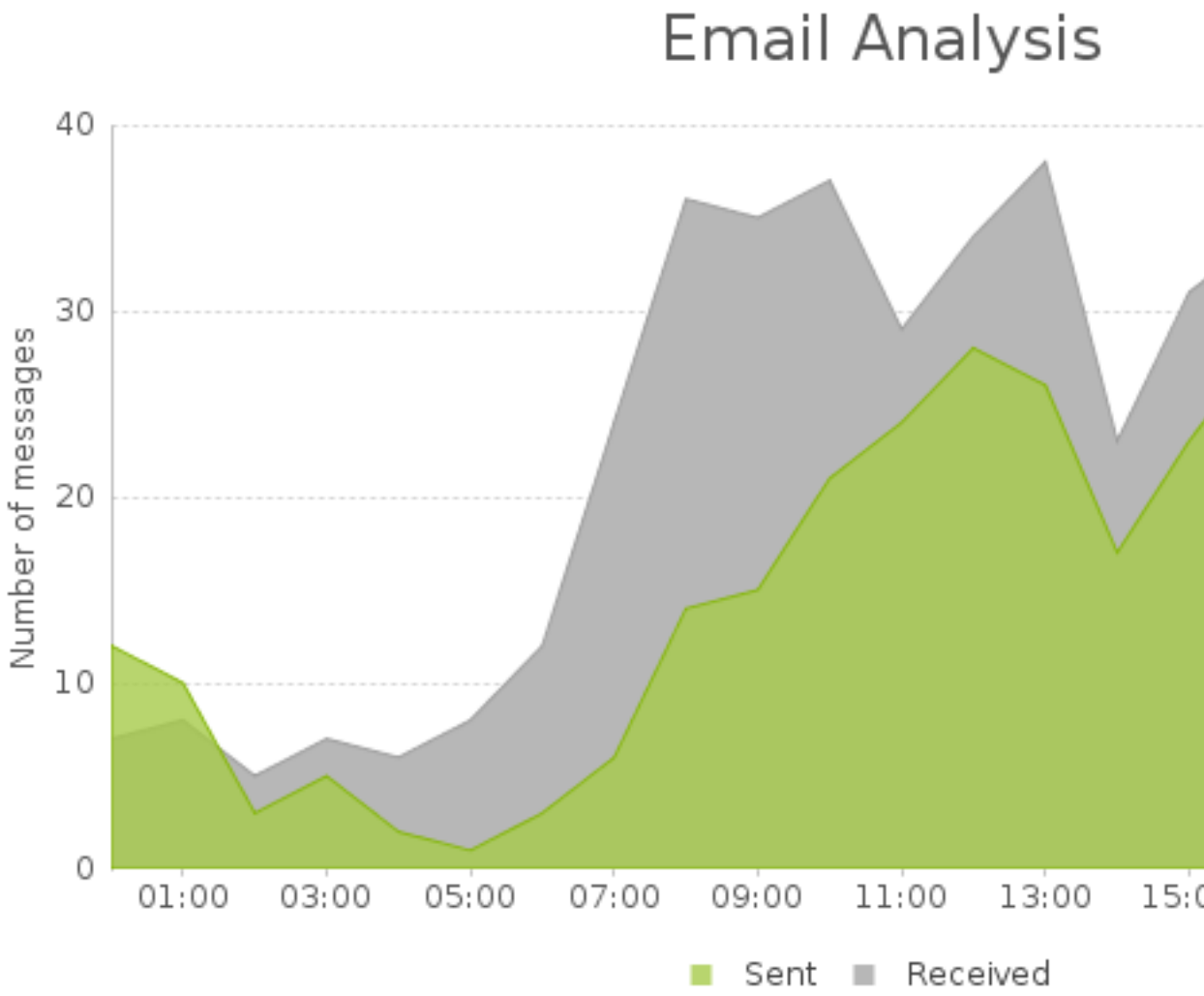
Parameter	Value

	`${'Oct 25, 1995 03:05:00 PM'?datetime}`
	`${'Oct 25, 1995'?date}`
	
	• $yValue_{ij}$ is a number that represents a value on the range axis for a series.

datasets	A sequence of the following form: $[dataset_1, dataset_2, \ldots dataset_n]$ where $dataset_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, xy plot • renderer: renderer, xy item renderer, xy area renderer • domain axis: axis, value axis, date axis • range axis: axis, value axis, number axis

The following example creates an area chart that compares the number of email messages sent and received by a fictitious email account.

This chart sets the 'plot.foregroundAlpha' property to 1.0 (the default is 0.5) and uses a semi-transparent color for the first series. The color is specified using the rgba format, rgba(166, 203, 74, 0.8) which means that the color has an opacity of 80%.




```
[#assign dataSet = [
  ['Date', 'Sent', 'Received'],
  ['2013-01-01T00:00'?datetime.iso, 12,  7],
  ['2013-01-01T01:00'?datetime.iso, 10,  8],
  ['2013-01-01T02:00'?datetime.iso,  3,  5],
  ['2013-01-01T03:00'?datetime.iso,  5,  7],
  ['2013-01-01T04:00'?datetime.iso,  2,  6],
  ['2013-01-01T05:00'?datetime.iso,  1,  8],
  ['2013-01-01T06:00'?datetime.iso,  3, 12],
  ['2013-01-01T07:00'?datetime.iso,  6, 24],
  ['2013-01-01T08:00'?datetime.iso, 14, 36],
  ['2013-01-01T09:00'?datetime.iso, 15, 35],
  ['2013-01-01T10:00'?datetime.iso, 21, 37],
  ['2013-01-01T11:00'?datetime.iso, 24, 29],
  ['2013-01-01T12:00'?datetime.iso, 28, 34],
  ['2013-01-01T13:00'?datetime.iso, 26, 38],
  ['2013-01-01T14:00'?datetime.iso, 17, 23],
  ['2013-01-01T15:00'?datetime.iso, 23, 31],
  ['2013-01-01T16:00'?datetime.iso, 28, 34],
  ['2013-01-01T17:00'?datetime.iso, 22, 29],
  ['2013-01-01T18:00'?datetime.iso, 10, 14],
  ['2013-01-01T19:00'?datetime.iso,  9, 12],
  ['2013-01-01T20:00'?datetime.iso,  6, 10],
```

```
  ['2013-01-01T21:00'?datetime.iso,  4,  8],
  ['2013-01-01T22:00'?datetime.iso, 12, 13],
  ['2013-01-01T23:00'?datetime.iso, 14, 11]
] /]

[#assign chartProperties = {
    'title': 'Email Analysis',
    'title.horizontalAlignment': 'center',
    'title.font': {'size': 24},
    'title.padding': [5, 0, 15, 0],
    'title.paint': '#555555',

    'legend.visible': true,
    'legend.frame': {"insets": [0, 0, 0, 0]},
    'legend.itemLabelPadding': [10, 10, 10, 10],

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#aaaaaa',
    'plot.domainGridlinesVisible': false,
    'plot.foregroundAlpha': 1.0,

    'plot.domainAxis.axisLineVisible': false,
    'plot.domainAxis.labelPaint': '#555555',
    'plot.domainAxis.labelFont': {'size': 12},
    'plot.domainAxis.tickLabelFont': {'size': 12},
    'plot.domainAxis.tickLabelPaint': '#555555',

    'plot.rangeAxis.axisLineVisible': true,
    'plot.rangeAxis.label': 'Number of messages',
    'plot.rangeAxis.labelPaint': '#555555',
    'plot.rangeAxis.labelFont': {'size': 12},
    'plot.rangeAxis.tickLabelFont': {'size': 12},
    'plot.rangeAxis.tickLabelPaint': '#555555',
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 10},
    'plot.rangeAxis.range': [0, 40],

    'plot.renderer.seriesPaint': ['rgba(166, 203, 74, 0.8)', 'rgb(183, 183, 183
    'plot.renderer.seriesOutlinePaint': ['rgb(135, 184, 14)', 'rgb(158, 158, 158
    'plot.renderer.legendArea': {'shape': 'rectangle', 'x': 0, 'y': 0, 'width':
    'plot.renderer.outline': true
} /]

<img src="[@jfreechart type='timeseriesarea' dataset=dataSet properties=chartPr
```

4.4.2.2. Bar charts

A bar chart is a two-axis chart with rectangular bars that can be either vertical or horizontal. Bar charts can help you:

- show change over time,
- compare values of different categories, and
- compare parts of a whole.

If you haven't used bar charts before, these resources explain them in more detail:

- An introduction to bar charts and how they are used to illustrate data [http://www2.le.ac.uk/offices/ld/resources/numerical-data/bar-charts]
- Bar chart [https://en.wikipedia.org/wiki/Bar_chart]
- A Histogram is NOT a Bar Chart [http://www.forbes.com/sites/naomirobbins/2012/01/04/a-histogram-is-not-a-bar-chart/]

4.4.2.2.1. Bar charts

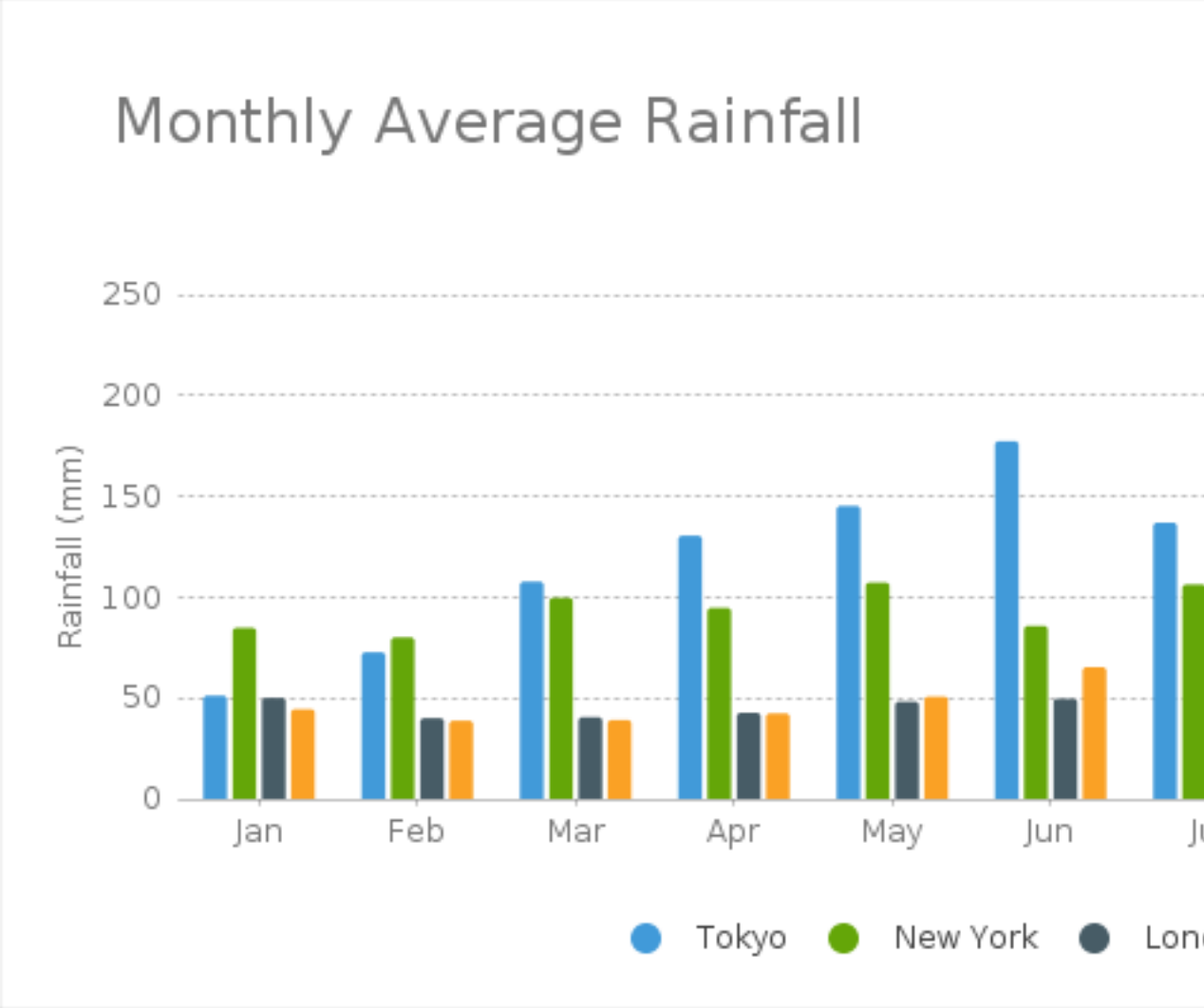
In order to create a bar chart, the jfreechart directive must be used with the following parameters:

Table 4.4. The jfreechart directive parameters for bar charts

Parameter	Value
type	bar
dataset	A sequence of the following form: [[**Heading , Series$_1$, ... Series$_n$**], [**xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$**], ... [**xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$**]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a series. • xValue$_i$ is a text that represents a value (category) on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset$_1$, dataset$_2$, ... dataset$_n$**] where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, category plot

Parameter	Value
	• renderer: renderer, category item renderer, bar renderer • domain axis: axis, category axis • range axis: axis, value axis, number axis

The following example creates a bar chart that shows the monthly average rainfall in four major cities.




```
[#-- http://www.worldclimate.com/ --]
[#assign categoryDataSet = [
  ['Month', 'Tokyo', 'New York', 'London', 'Berlin'],
  [  'Jan',    49.9,       83.6,     48.9,     43.2],
  [  'Feb',    71.5,       78.8,     38.8,     37.6],
  [  'Mar',   106.4,       98.5,     39.3,     37.8],
  [  'Apr',   129.2,       93.4,     41.4,     41.1],
  [  'May',   144.0,      106.0,     47.0,     49.4],
  [  'Jun',   176.0,       84.5,     48.3,     64.1],
  [  'Jul',   135.6,      105.0,     59.0,     71.1],
  [  'Aug',   148.5,      104.3,     59.6,     62.1],
  [  'Sep',   216.4,       91.2,     52.4,     44.1],
  [  'Oct',   194.1,       83.5,     65.2,     44.3],
  [  'Nov',    95.6,      106.6,     59.3,     45.5],
```

```
  [  'Dec',    54.4,       92.3,     51.2,     47.9]
] /]

[#assign chartProperties = {
    'borderPaint': '#777777',
    'borderVisible': true,
    'padding': [20, 10, 20, 10],
    'borderStroke': {'width': 0.2},

    'title': 'Monthly Average Rainfall',
    'title.font': {'size': 24},
    'title.horizontalAlignment': 'left',
    'title.padding': [15, 35, 50, 5],
    'title.paint': '#777777',

    'legend.position': 'bottom',
    'legend.frame': {"insets": [0, 0, 0, 0]},
    'legend.itemLabelPadding': [20, 15, 0, 15],
    'legend.legendItemGraphicPadding': [20, 0, 0, 0],
    'plot.renderer.baseLegendShape': {'shape': 'ellipse', 'width': 10, 'height'

    'plot.renderer.seriesPaint': ['#419ad9', '#64a608', '#475c66', '#faa125'],

    'plot.renderer.drawBarOutline': true,
    'plot.renderer.baseOutlineStroke': {'width': 2.0, 'cap': 2, 'join': 2},
    'plot.renderer.seriesOutlinePaint': ['#419ad9', '#64a608', '#475c66', '#faa

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#777777',

    'plot.domainAxis.categoryMargin': 0.3,
    'plot.domainAxis.lowerMargin': 0.015,
    'plot.domainAxis.upperMargin': 0.015,
    'plot.domainAxis.tickLabelPaint': '#777777',

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.label': 'Rainfall (mm)',
    'plot.rangeAxis.labelFont': {'size': 12},
    'plot.rangeAxis.labelPaint': '#777777',
    'plot.rangeAxis.tickLabelPaint': '#777777',
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 50},
    'plot.rangeAxis.range': [0, 250],

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
} /]

<img src="[@jfreechart type='bar' dataset=categoryDataSet properties=chartPrope
```

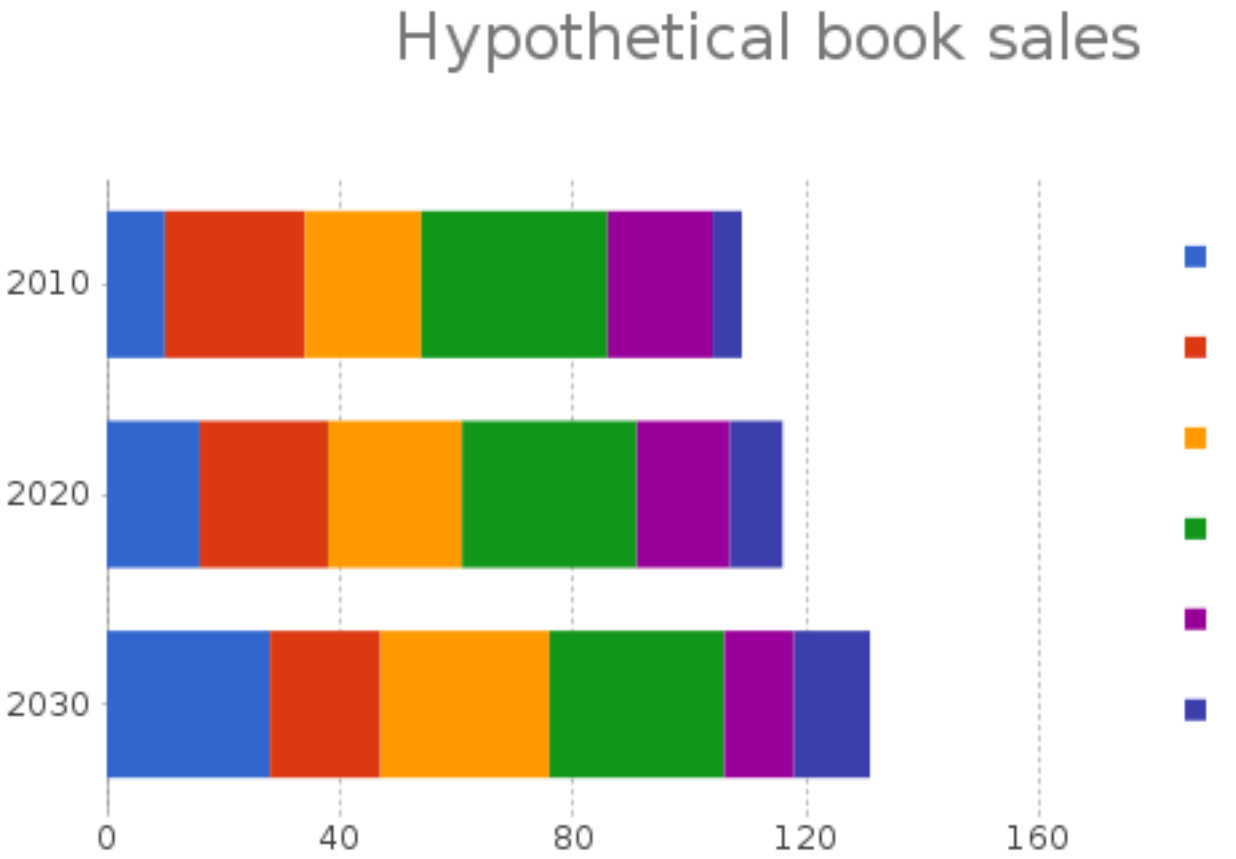
4.4.2.2.2. Stacked bar charts

In order to create a stacked bar chart, the jfreechart directive must be used with the following parameters:

Table 4.5. The jfreechart directive parameters for stacked bar charts

Parameter	Value
type	stackedbar
dataset	A sequence of the following form: [[**Heading , Series$_1$, ... Series$_n$**], [**xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$**], ... [**xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$**]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a series. • xValue$_i$ is a text that represents a value (category) on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset$_1$, dataset$_2$, ... dataset$_n$**] where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, category plot • renderer: renderer, category item renderer, bar renderer, stacked bar renderer • domain axis: axis, category axis • range axis: axis, value axis, number axis

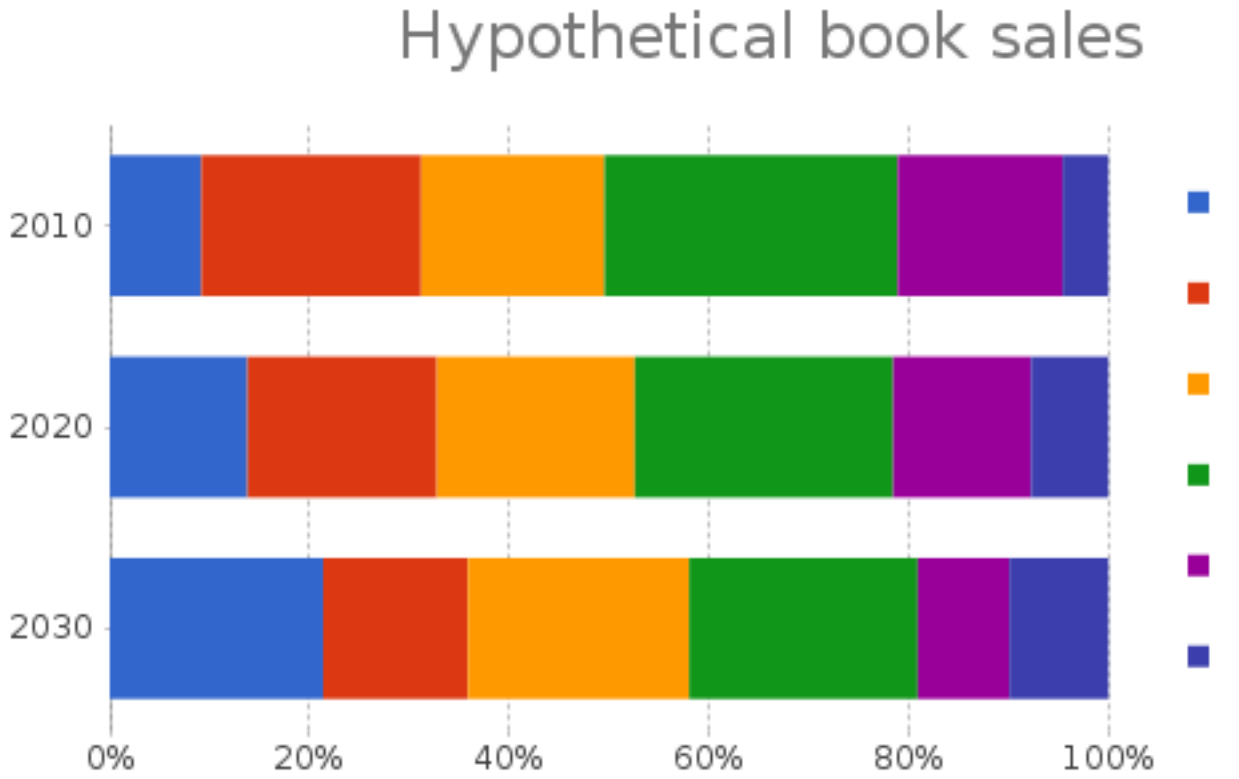
The following example creates a stacked bar chart that shows hypothetical book sales, divided by genre and compared across time.




```
[#assign categoryDataSet = [
  ['Genre', 'Fantasy and Sci Fi', 'Romance', 'Mystery/Crime', 'General', 'Weste
  ['2010', 10, 24, 20, 32, 18, 5],
  ['2020', 16, 22, 23, 30, 16, 9],
  ['2030', 28, 19, 29, 30, 12, 13]
] /]

[#assign chartProperties = {
    'title': 'Hypothetical book sales',
    'title.font': {'size': 24},
    'title.padding': [5, 5, 15, 5],
    'title.paint': '#777777',

    'legend.position': 'right',
    'legend.verticalAlignment': 'top',
    'legend.frame': {"insets": [0, 0, 0, 0]},
    'legend.margin': [15, 0, 0, 0],
    'legend.itemLabelPadding': [10, 10, 10, 10],

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeAxisLocation': 'bottom or right',
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#777777',

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 40},
    'plot.rangeAxis.range': [0, 180],
```

```
    'plot.orientation': 'horizontal',
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
} /]

<img src="[@jfreechart type='stackedbar' dataset=categoryDataSet properties=cha
```

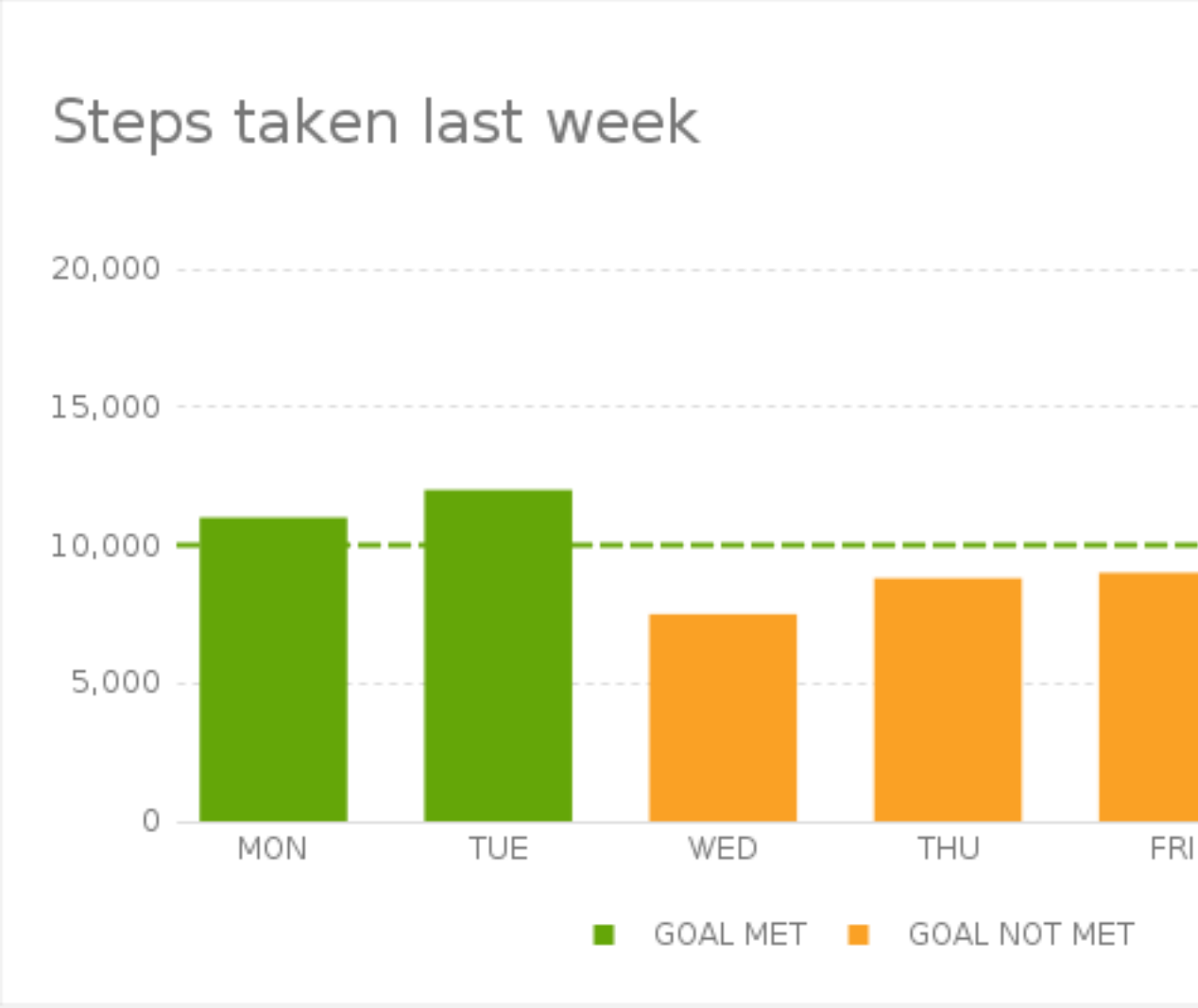
This chart can be configured to show percentages if the 'plot.renderer.renderAsPercentages' property is used. The range axis also needs to use a percentage format ('plot.renderer.renderAsPercentages') to show percentages. The following chart emphasizes these two properties.




```
[#assign categoryDataSet = [
  ['Genre', 'Fantasy and Sci Fi', 'Romance', 'Mystery/Crime', 'General', 'Weste
  ['2010', 10, 24, 20, 32, 18, 5],
  ['2020', 16, 22, 23, 30, 16, 9],
  ['2030', 28, 19, 29, 30, 12, 13]
] /]

[#assign chartProperties = {
    'title': 'Hypothetical book sales',
    'title.font': {'size': 24},
    'title.padding': [5, 5, 15, 5],
    'title.paint': '#777777',

    'legend.position': 'right',
    'legend.verticalAlignment': 'top',
    'legend.frame': {"insets": [0, 0, 0, 0]},
    'legend.margin': [15, 0, 0, 0],
    'legend.itemLabelPadding': [10, 10, 10, 10],

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeAxisLocation': 'bottom or right',
```

```
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#777777',

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.numberFormatOverride': '0%',
    'plot.renderer.renderAsPercentages': true,

    'plot.orientation': 'horizontal',
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
} /]

<img src="[@jfreechart type='stackedbar' dataset=categoryDataSet properties=cha
```

Here's another example that uses a stacked bar chart to show whether a goal was met or not.




```
[#assign categoryDataSet = [
  ['DAY', 'GOAL MET', 'GOAL NOT MET'],
  ['MON', 11000,    0],
  ['TUE', 12000,    0],
  ['WED',     0, 7500],
  ['THU',     0, 8800],
  ['FRI',     0, 9000],
  ['SAT', 13000,    0],
  ['SUN',     0, 5500]
```

```
] /]

[#assign chartProperties = {
    'borderPaint': '#777777',
    'borderVisible': true,
    'padding': [20, 10, 20, 10],
    'borderStroke': {'width': 0.2},

    'title': 'Steps taken last week',
    'title.font': {'size': 24},
    'title.horizontalAlignment': 'left',
    'title.padding': [15, 10, 40, 5],
    'title.paint': '#777777',

    'legend.visible': true,
    'legend.frame': {'insets': [0, 0, 0, 0]},
    'legend.position': 'bottom',
    'legend.verticalAlignment': 'top',
    'legend.itemPaint': '#777777',
    'legend.itemLabelPadding': [10, 15, 0, 15],
    'legend.legendItemGraphicPadding': [10, 0, 0, 0],
    'legend.backgroundPaint': '',

    'plot.renderer.seriesPaint': ['#64a608', '#faa125'],

    'plot.axisOffset': [0, 0, 0, 0],

    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#777777',
    'plot.rangeGridlineStroke': {'width': 0.1, 'cap': 0, 'join': 2, 'dash': [3.

    'plot.rangeCrosshairValue': 10000,
    'plot.rangeCrosshairVisible': true,
    'plot.rangeCrosshairPaint': '#64a608',
    'plot.rangeCrosshairStroke': {'width': 2.0, 'cap': 0, 'join': 2, 'dash': [9

    'plot.domainAxis.axisLinePaint': '#bbbbbb',
    'plot.domainAxis.categoryMargin': 0.3,
    'plot.domainAxis.lowerMargin': 0.015,
    'plot.domainAxis.upperMargin': 0.015,
    'plot.domainAxis.tickLabelPaint': '#777777',
    'plot.domainAxis.tickMarksVisible': false,

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.tickLabelPaint': '#777777',
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 5000},
    'plot.rangeAxis.range': [0, 20000],

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
} /]

<img src="[@jfreechart type='stackedbar' dataset=categoryDataSet properties=cha
```

4.4.2.3. Line charts

A line chart is a type of two-axis chart that presents data as a series of points connected by straight lines. Line charts are most often used to visualize data that changes over time. Line charts can help you:

- See overall patterns, such as trends, fluctuations, cycles and rates of change so you can make predictions about data not yet recorded.
- Analyze and compare multiple data sets so you can see if there is any correlation between them. Some experts recommend no more than 4 lines on a single graph; any more than that and it becomes difficult to interpret.

If you haven't used line charts before, these resources explain them in more detail:

- Data Visualization 101: Line Charts [http://visage.co/data-visualization-101-line-charts/]
- Charts and Graphs [https://www.mindtools.com/pages/article/Charts_and_Diagrams.htm]
- Line Charts Are Not Always the Best Way to Show Time Series [http://www.forbes.com/sites/naomirobbins/2012/12/27/line-charts-are-not-always-the-best-way-to-show-time-series/]

4.4.2.3.1. Category line charts

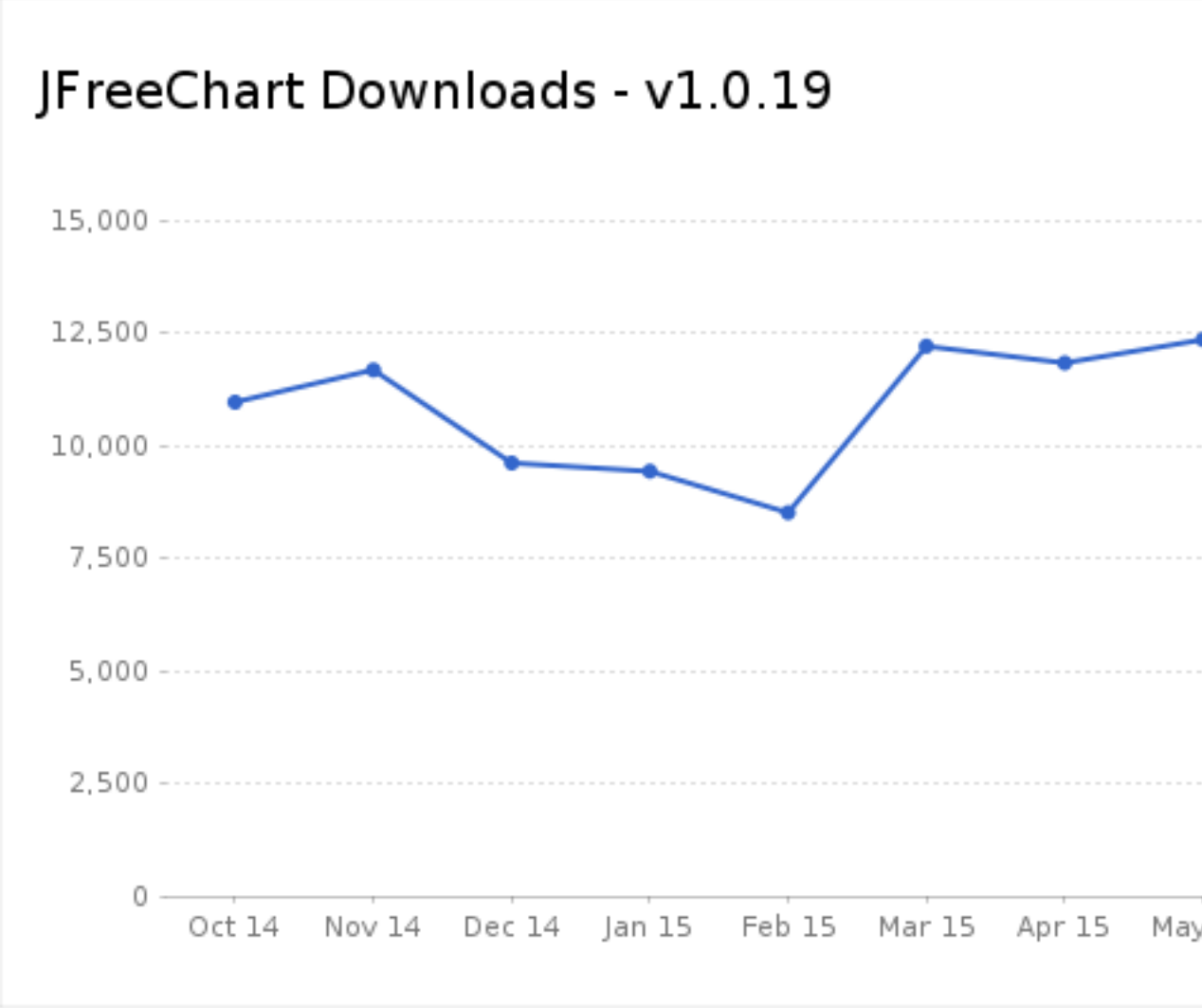
In order to create a line chart that uses texts (categories) on the domain axis, the jfreechart directive must be used with the following parameters:

Table 4.6. The jfreechart directive parameters for category line charts

Parameter	Value
type	line
dataset	A sequence of the following form: [[**Heading** , **Series**$_1$, ... **Series**$_n$], [**xValue**$_1$, **yValue**$_{11}$, ... **yValue**$_{n1}$], ... [**xValue**$_k$, **yValue**$_{1k}$, ... **yValue**$_{nk}$]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a line. • xValue$_i$ is a text that represents a value (category) on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset**$_1$, **dataset**$_2$, ... **dataset**$_n$] where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the

Parameter	Value
	datasets parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, category plot • renderer: renderer, category item renderer, line and shape renderer • domain axis: axis, category axis • range axis: axis, value axis, number axis

The following example creates a line chart with just one line that shows the downloads for JFreeChart version 1.0.19 between October 2014 and September 2015. The downloads from each month are represented as a circle.

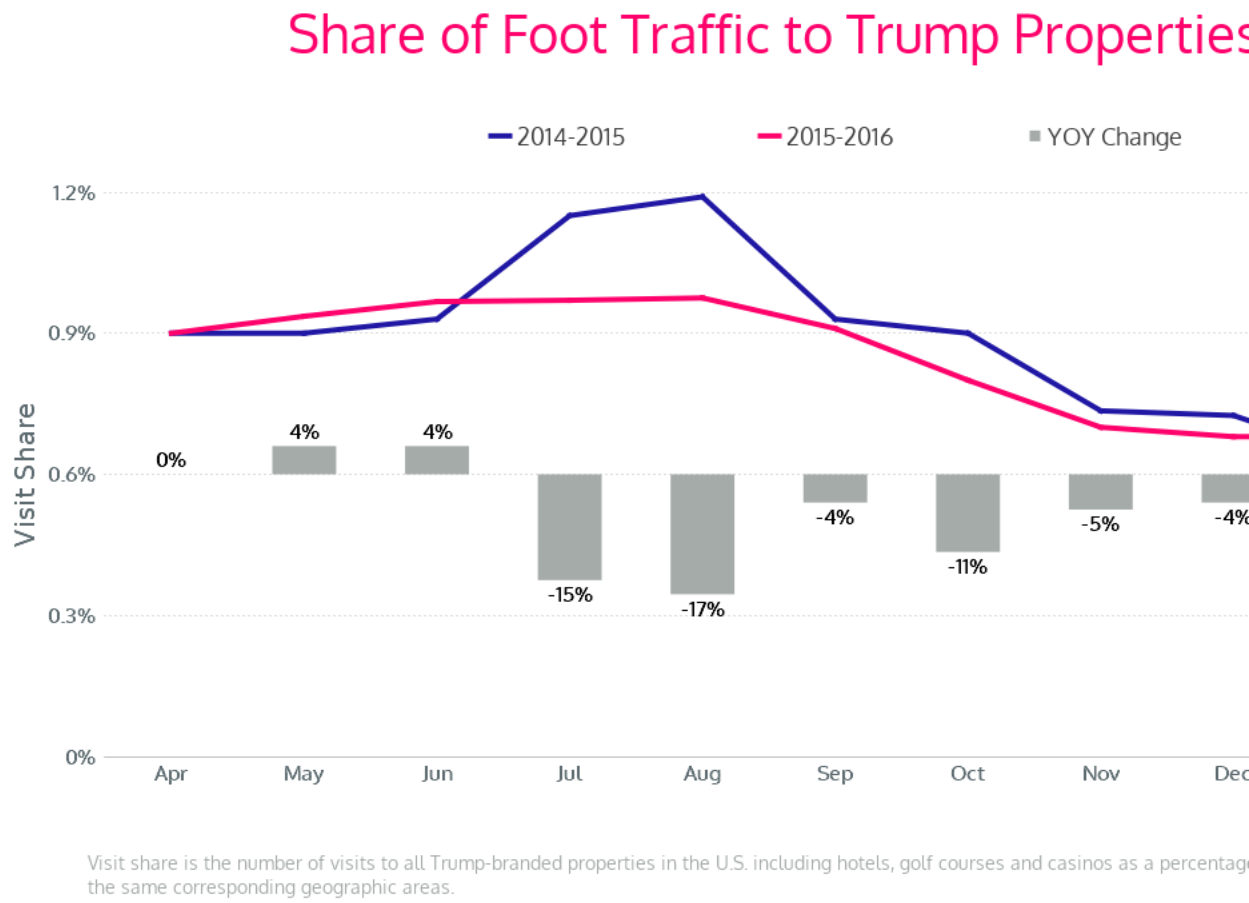



```
[#-- http://sf.net/projects/jfreechart/files/1.%20JFreeChart/1.0.19/jfreechart-
```

```
[#assign dataSet = [
  ['Month', 'Downloads'],
  ['Oct 14', 10958],
  ['Nov 14', 11673],
  ['Dec 14', 9606],
  ['Jan 15', 9421],
  ['Feb 15', 8504],
  ['Mar 15', 12197],
  ['Apr 15', 11827],
  ['May 15', 12343],
  ['Jun 15', 10652],
  ['Jul 15', 8428],
  ['Aug 15', 7931],
  ['Sep 15', 8677]
] /]

[#assign chartProperties = {
    'borderPaint': '#777777',
    'borderVisible': true,
    'padding': [20, 10, 20, 10],
    'borderStroke': {'width': 0.2},

    'title': 'JFreeChart Downloads - v1.0.19',
    'title.font': {'size': 20},
    'title.padding': [5, 5, 35, 5],
    'title.horizontalAlignment': 'left',

    'legend.visible': false,

    'plot.renderer.baseFillPaint': '#3366cc',
    'plot.renderer.drawOutlines': true,
    'plot.renderer.baseShapesVisible': true,
    'plot.renderer.baseLinesVisible': true,
    'plot.renderer.baseShapesFilled': true,
    'plot.renderer.useFillPaint': true,
    'plot.renderer.seriesShape': [{'shape': 'ellipse', 'x': -2, 'y': -2, 'width
    'plot.renderer.seriesStroke': [{'width': 2.0}],
    'plot.renderer.seriesOutlineStroke': [{'width': 2.0}],

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#bbbbbb',
    'plot.rangeGridlineStroke': {'width': 1.0, 'cap': 0, 'join': 2, 'miterlimit

    'plot.domainAxis.lowerMargin': 0.01,
    'plot.domainAxis.upperMargin': 0.01,
    'plot.domainAxis.tickLabelPaint': '#777777',
    'plot.domainAxis.tickLabelFont': {'size': 11},

    'plot.rangeAxis.tickLabelPaint': '#777777',
    'plot.rangeAxis.tickLabelFont': {'size': 11},
    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 2500},
    'plot.rangeAxis.range': [0, 15000],

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
```

```
} /]

<img src="[@jfreechart type='line' dataset=dataSet properties=chartProperties w
```

The following chart uses two datasets and two axes to represent them:




```
[#-- Note: the following datasets are not accurate, they approximate the vales
[#-- https://medium.com/foursquare-direct/how-the-trump-presidential-campaign-i

[#assign dataSet1 = [
  ['Month', '2014-2015', '2015-2016'],
  ['Apr', 0.009, 0.009],
  ['May', 0.009, 0.00936],
  ['Jun', 0.00930, 0.009672],
  ['Jul', 0.0115, 0.0097],
  ['Aug', 0.0119, 0.00975],
  ['Sep', 0.00930, 0.00910],
  ['Oct', 0.009, 0.008],
  ['Nov', 0.00735, 0.007],
  ['Dec', 0.00725, 0.0068],
  ['Jan', 0.0063, 0.0068],
  ['Feb', 0.0065, 0.0070],
  ['Mar', 0.0095, 0.0078]
] /]

[#assign dataSet2 = [
```

```
  ['Month', 'YOY Change'],
  ['Apr', 0.00],
  ['May', 0.04],
  ['Jun', 0.04],
  ['Jul', -0.15],
  ['Aug', -0.17],
  ['Sep', -0.04],
  ['Oct', -0.11],
  ['Nov', -0.05],
  ['Dec', -0.04],
  ['Jan', 0.04],
  ['Feb', 0.05],
  ['Mar', -0.17]
] /]

[#assign chartProperties = {
    'padding': [0, 10, 0, 10],
    'textAntiAlias': true,

    'title': 'Share of Foot Traffic to Trump Properties in the U.S.',
    'title.font': {'size': 40, 'file': '~/Documents/Oxygen/Oxygen-Regular.ttf'},
    'title.paint': '#ff026c',
    'title.padding': [5, 5, 35, 5],
    'title.horizontalAlignment': 'center',

    'subtitles': ['FOURSQUARE', 'Visit share is the number of visits to all Trum

    'subtitle[0].horizontalAlignment': 'right',
    'subtitle[0].textAlignment': 'right',
    'subtitle[0].font': {'size': 42, 'weight': 'bold', 'file': '~/Documents/Oxyg
    'subtitle[0].padding': [40, 10, 0, 10],
    'subtitle[0].paint': '#2018a4',
    'subtitle[0].position': 'bottom',

    'subtitle[1].horizontalAlignment': 'left',
    'subtitle[1].textAlignment': 'left',
    'subtitle[1].font': {'size': 14, 'file': '~/Documents/Oxygen/Oxygen-Regular
    'subtitle[1].padding': [40, 70, 0, 70],
    'subtitle[1].paint': '#a5aba9',
    'subtitle[1].position': 'bottom',

    'legend.visible': true,
    'legend.position': 'top',
    'legend.verticalAlignment': 'top',
    'legend.frame': {'insets': [0, 0, 0, 0]},
    'legend.margin': [0, 0, 20, 0],
    'legend.itemLabelPadding': [5, 5, 5, 100],
    'legend.legendItemGraphicPadding': [0, 0, 0, 0],
    'legend.itemFont': {'size': 18, 'file': '~/Documents/Oxygen/Oxygen-Regular.t

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#bbbbbb',
    'plot.rangeGridlineStroke': {'width': 1.0, 'cap': 0, 'join': 2, 'miterlimit

    'plot.domainAxis.lowerMargin': 0.01,
    'plot.domainAxis.upperMargin': 0.01,
    'plot.domainAxis.tickLabelPaint': '#5e6c71',
```

```
        'plot.domainAxis.tickLabelFont': {'size': 14, 'weight': 'bold', 'file': '~/
        'plot.domainAxis.tickMarksVisible': false,

        'plot.rangeAxis': ['org.jfree.chart.axis.NumberAxis', 'org.jfree.chart.axis
        'plot.mapDatasetToRangeAxis':[0, 1],

        'plot.rangeAxis[0].axisLineVisible': false,
        'plot.rangeAxis[0].tickLabelPaint': '#5e6c71',
        'plot.rangeAxis[0].tickLabelFont': {'size': 14, 'weight': 'bold', 'file': '
        'plot.rangeAxis[0].tickMarksVisible': false,
        'plot.rangeAxis[0].range': [0, 0.012],
        'plot.rangeAxis[0].tickUnit': {'size': 0.003},
        'plot.rangeAxis[0].numberFormatOverride': '0.#%',
        'plot.rangeAxis[0].label': 'Visit Share',
        'plot.rangeAxis[0].labelPaint': '#5e6c71',
        'plot.rangeAxis[0].labelFont': {'size': 20},

        'plot.rangeAxis[1].tickLabelFont': {'size': 14, 'weight': 'bold', 'file': '
        'plot.rangeAxis[1].tickLabelPaint': '#5e6c71',
        'plot.rangeAxis[1].tickMarksVisible': false,
        'plot.rangeAxis[1].range': [-0.40, 0.40],
        'plot.rangeAxis[1].tickUnit': {'size': 0.10},
        'plot.rangeAxis[1].numberFormatOverride': '#.#%',
        'plot.rangeAxis[1].label': 'YOY Change in Visit Share',
        'plot.rangeAxis[1].labelPaint': '#5e6c71',
        'plot.rangeAxis[1].labelFont': {'size': 20},

        'plot.renderer': ['org.jfree.chart.renderer.category.LineAndShapeRenderer',

        'plot.renderer[0].baseFillPaint': '#3366cc',
        'plot.renderer[0].drawOutlines': false,
        'plot.renderer[0].baseShapesVisible': false,
        'plot.renderer[0].baseLinesVisible': true,
        'plot.renderer[0].baseShapesFilled': false,
        'plot.renderer[0].useFillPaint': true,
        'plot.renderer[0].seriesStroke': [{'width': 4.0}, {'width': 4.0}],
        'plot.renderer[0].seriesPaint': ['#2018a4', '#ff026c'],

        'plot.renderer[1].seriesItemLabelGenerator': [{'labelFormat':'{2}', 'number
        'plot.renderer[1].seriesItemLabelFont': [{'size': 14, 'weight': 'bold', 'fi
        'plot.renderer[1].seriesItemLabelsVisible': [true],
        'plot.renderer[1].seriesPaint': ['#a5aba9'],
        'plot.renderer[1].maximumBarWidth': 0.04,
        'plot.renderer[1].drawBarOutline': false,
        'plot.renderer[1].shadowVisible': false,

        'plot.backgroundPaint': '',
        'plot.outlineVisible': false
    } /]

    <img src="[@jfreechart type='line' datasets=[dataSet1, dataSet2] properties=cha
```

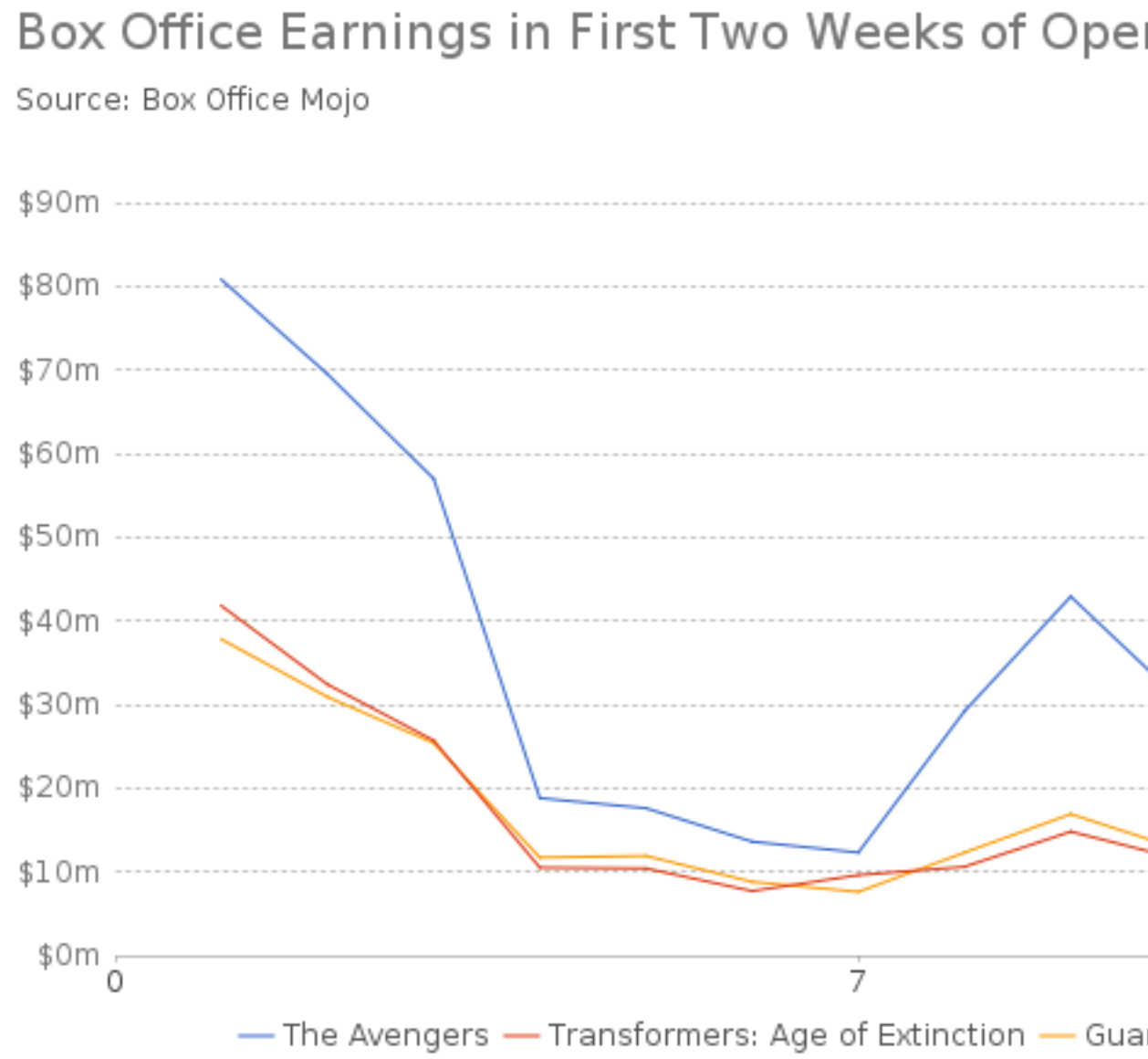
4.4.2.3.2. XY line charts

In order to create a line chart that uses numbers on the domain axis, the jfreechart directive must be used with the following parameters:

Table 4.7. The jfreechart directive parameters for xy line charts

Parameter	Value
type	xyline
dataset	A sequence of the following form: [[**Heading , Series$_1$, ... Series$_n$**], [**xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$**], ... [**xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$**]] where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a line. • xValue$_i$ is a number that represents a value on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: [**dataset$_1$, dataset$_2$, ... dataset$_n$**] where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, xy plot • renderer: renderer, xy item renderer, xy line and shape renderer • domain axis: axis, value axis, number axis • range axis: axis, value axis, number axis

The following example creates a line chart to represent the box office earnings in first two weeks of opening for three movies.




```
[#--
http://www.boxofficemojo.com/movies/?page=daily&view=chart&id=avengers11.htm
http://www.boxofficemojo.com/movies/?page=daily&view=chart&id=transformers4.htm
http://www.boxofficemojo.com/movies/?page=daily&view=chart&id=marvel2014a.htm
--]

[#assign dataSet = [
 ['Day', 'The Avengers', 'Transformers: Age of Extinction',  'Guardians of the
 [ 1, 80.8, 41.8, 37.8],
 [ 2, 69.5, 32.4, 30.9],
 [ 3,   57, 25.7, 25.4],
 [ 4, 18.8, 10.5, 11.7],
 [ 5, 17.6, 10.4, 11.9],
 [ 6, 13.6,  7.7,  8.8],
 [ 7, 12.3,  9.6,  7.6],
 [ 8, 29.2, 10.6, 12.3],
 [ 9, 42.9, 14.8, 16.9],
 [10, 30.9, 11.6, 12.8],
 [11,  7.9,  4.7,  5.3],
```

```
 [12,  8.4,  5.2,  6.6],
 [13,  6.3,  3.6,  4.8],
 [14,  6.2,  3.4,  4.2]
]/]

[#assign chartProperties = {
    'title': 'Box Office Earnings in First Two Weeks of Opening',
    'title.font': {'size': 20},
    'title.horizontalAlignment': 'left',
    'title.padding': [15, 10, 10, 5],
    'title.paint': '#777777',

    'subtitles': ['Source: Box Office Mojo'],
    'subtitle[0].horizontalAlignment': 'left',
    'subtitle[0].padding': [0, 10, 30, 10],
    'subtitle[0].font': {'size': 12},
    'subtitle[0].paint': '#555555',

    'legend.frame': {'insets': [0, 0, 0, 0]},
    'legend.visible': true,
    'legend.position': 'bottom',
    'legend.verticalAlignment': 'top',
    'legend.itemPaint': '#555555',

    'plot.axisOffset': [0, 0, 0, 0],
    'plot.rangeGridlinesVisible': true,
    'plot.rangeGridlinePaint': '#777777',

    'plot.domainAxis.tickUnit': {'size': 7},

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.numberFormatOverride': '${"$"}#0m',
    'plot.rangeAxis.labelFont': {'size': 12},
    'plot.rangeAxis.labelPaint': '#777777',
    'plot.rangeAxis.tickLabelPaint': '#777777',
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 10},
    'plot.rangeAxis.range': [0, 90],

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false
} /]

<img src="[@jfreechart type='xyline' dataset=dataSet properties=chartProperties
```

4.4.2.3.3. Time series line charts

In order to create a line chart that uses dates/times on the domain axis, the jfreechart directive must be used with the following parameters:

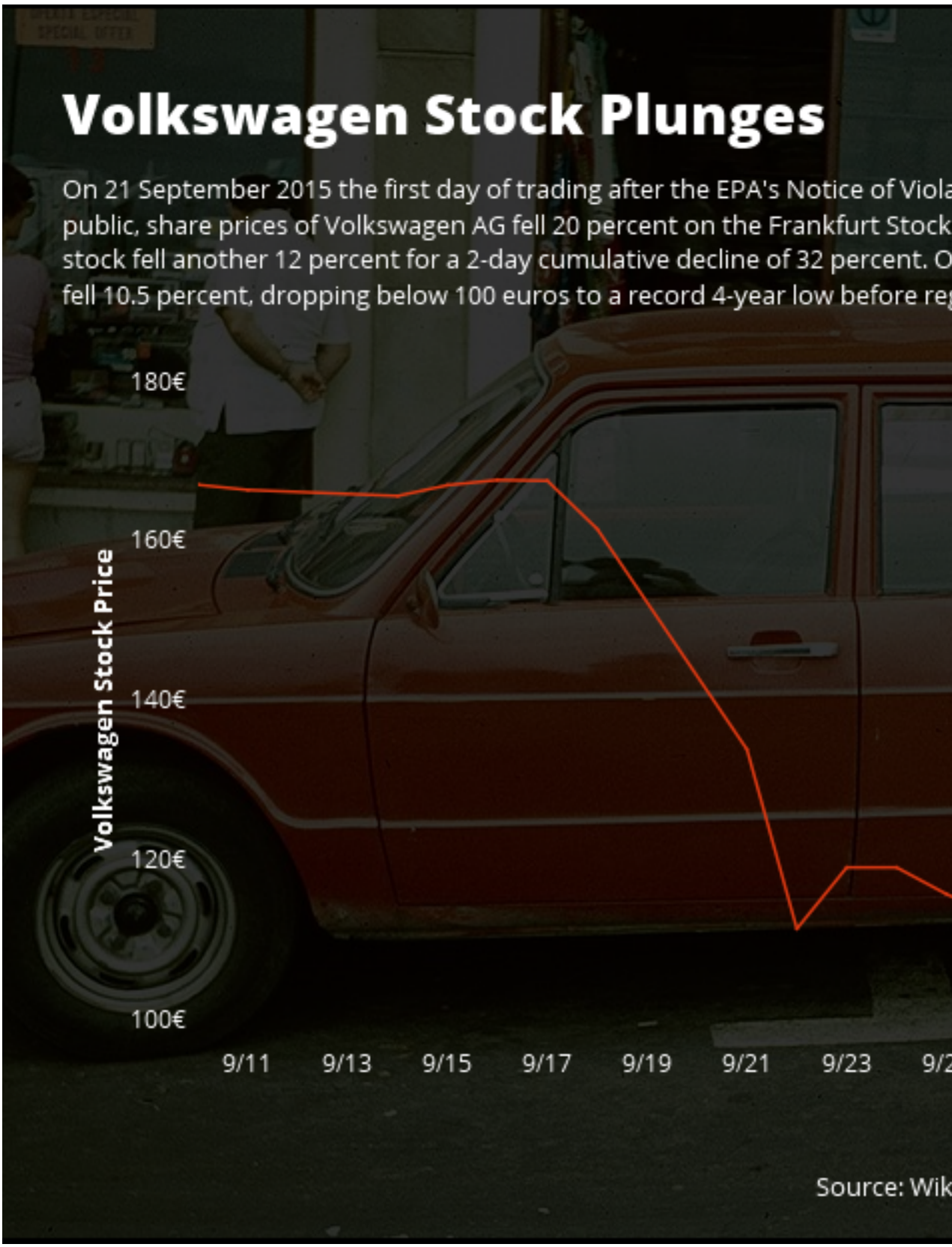
Table 4.8. The jfreechart directive parameters for time series line charts

Parameter	Value
type	timeseriesline
dataset	A sequence of the following form: [

Parameter	Value
	[Heading , Series$_1$, ... Series$_n$], **[xValue$_1$, yValue$_{11}$, ... yValue$_{n1}$],** ... **[xValue$_k$, yValue$_{1k}$, ... yValue$_{nk}$]** **]** where: • Heading can be any text that describes the values on the domain axis. • Series$_i$ is a text that represents the name of a line. • xValue$_i$ is a date/time that represents a value on the domain axis. • yValue$_{ij}$ is a number that represents a value on the range axis for a series.
datasets	A sequence of the following form: **[dataset$_1$, dataset$_2$, ... dataset$_n$]** where dataset$_i$ is a dataset defined according to the above format. This chart can use one dataset if the **dataset** parameter is used or multiple datasets if the **datasets** parameter is used. It doesn't accept datasets specified using both parameters at the same time. The datasets parameter is used by charts with multiple axes.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, xy plot • renderer: renderer, xy item renderer, xy line and shape renderer • domain axis: axis, value axis, date axis • range axis: axis, value axis, number axis

The following example creates a line chart that shows the evolution of the Volkswagen AG stock price after it admitted to cheating on U.S. air pollution tests for years.

This chart uses the Open Sans [https://www.google.com/fonts/specimen/Open+Sans] font and it expects to find its files in the folder **~/Documents/Open_Sans/**.




```
[#--
https://en.wikipedia.org/wiki/Volkswagen_emissions_scandal#Stock_value
http://www.boerse-frankfurt.de/en/equities/volkswagen+ag+st+DE0007664005/price+
--]
```

```
[#assign dataSet = [
  ['Date', 'Price'],
  ['2015-09-01'?date.iso, 161.95],
  ['2015-09-02'?date.iso, 159.50],
  ['2015-09-03'?date.iso, 164.40],
  ['2015-09-04'?date.iso, 159.95],
  ['2015-09-07'?date.iso, 161.15],
  ['2015-09-08'?date.iso, 165.40],
  ['2015-09-09'?date.iso, 169.60],
  ['2015-09-10'?date.iso, 166.90],
  ['2015-09-11'?date.iso, 166.25],
  ['2015-09-14'?date.iso, 165.50],
  ['2015-09-15'?date.iso, 166.85],
  ['2015-09-16'?date.iso, 167.50],
  ['2015-09-17'?date.iso, 167.40],
  ['2015-09-18'?date.iso, 161.35],
  ['2015-09-21'?date.iso, 133.70],
  ['2015-09-22'?date.iso, 111.20],
  ['2015-09-23'?date.iso, 118.90],
  ['2015-09-24'?date.iso, 118.90],
  ['2015-09-25'?date.iso, 115.55],
  ['2015-09-28'?date.iso, 107.10],
  ['2015-09-29'?date.iso, 103.30],
  ['2015-09-30'?date.iso, 104.95],
  ['2015-10-01'?date.iso, 105.05],
  ['2015-10-02'?date.iso, 101.15]
] /]

[#assign chartProperties = {
    'backgroundImage': 'https://upload.wikimedia.org/wikipedia/commons/thumb/2/
    'backgroundImageAlignment': 0,
    'backgroundImageAlpha': 0.2,
    'backgroundPaint': '#000000',

    'borderPaint': '',
    'borderVisible': true,
    'padding': [30, 30, 20, 30],
    'borderStroke': {'width': 0.2},

    'title': 'Volkswagen Stock Plunges',
    'title.horizontalAlignment': 'left',
    'title.font': {'size': 28, 'file': '~/Documents/Open_Sans/OpenSans-ExtraBol
    'title.padding': [5, 0, 10, 0],
    'title.paint': '#ffffff',

    'subtitles': ['On 21 September 2015 the first day of trading after the EPA\

    'subtitle[0].horizontalAlignment': 'left',
    'subtitle[0].font': {'size': 13, 'file': '~/Documents/Open_Sans/OpenSans-Re
    'subtitle[0].textAlignment': 'left',
    'subtitle[0].padding': [0, 0, 30, 0],
    'subtitle[0].paint': '#ffffff',

    'subtitle[1].horizontalAlignment': 'right',
    'subtitle[1].font': {'size': 13, 'file': '~/Documents/Open_Sans/OpenSans-Re
    'subtitle[1].padding': [40, 10, 0, 10],
    'subtitle[1].paint': '#ffffff',
    'subtitle[1].position': 'bottom',
```

```
    'legend.visible': false,

    'plot.axisOffset': [0, 0, 10, 0],
    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.rangeGridlinesVisible': false,
    'plot.domainGridlinesVisible': false,

    'plot.renderer.baseLinesVisible': true,
    'plot.renderer.seriesStroke': [{'width': 1.5}],
    'plot.renderer.seriesPaint': ['#dc3912'],

    'plot.domainAxis.axisLineVisible': false,
    'plot.domainAxis.lowerMargin': 0.01,
    'plot.domainAxis.upperMargin': 0.01,
    'plot.domainAxis.tickMarksVisible': false,
    'plot.domainAxis.tickLabelPaint': '#ffffff',
    'plot.domainAxis.tickLabelFont': {'size': 12, 'file': '~/Documents/Open_Sans
    'plot.domainAxis.tickUnit': {'unitType': 'day', 'multiple': 2},
    'plot.domainAxis.dateFormatOverride': 'M/d',
    'plot.domainAxis.range': ['2015-09-10'?date.iso, '2015-10-03'?date.iso],

    'plot.rangeAxis.axisLineVisible': false,
    'plot.rangeAxis.label': 'Volkswagen Stock Price',
    'plot.rangeAxis.labelPaint': '#ffffff',
    'plot.rangeAxis.labelFont': {'size': 13, 'file': '~/Documents/Open_Sans/Oper
    'plot.rangeAxis.tickLabelFont': {'size': 12, 'file': '~/Documents/Open_Sans/
    'plot.rangeAxis.tickLabelPaint': '#ffffff',
    'plot.rangeAxis.tickMarksVisible': false,
    'plot.rangeAxis.tickUnit': {'size': 20},
    'plot.rangeAxis.numberFormatOverride': '0€',
    'plot.rangeAxis.range': [100, 180]
} /]

<img src="[@jfreechart type='timeseriesline' dataset=dataSet properties=chartPr
```

4.4.2.4. Pie charts

A pie chart is a type of chart in which a circle is divided into sectors that each represent a proportion of the whole. Pie charts are used to make part-to-whole comparisons however they should be used only when a rough overview is enough because it's hard to make exact comparisons between sectors of similar sizes.

If you haven't used pie charts before, these resources explain them in more detail:

- Understanding Pie Charts [https://eagereyes.org/techniques/pie-charts]
- Save the Pies for Dessert [http://www.perceptualedge.com/articles/08-21-07.pdf]
- Donut charts are worse than pie charts: Have no fear! There are many alternatives. [http://vizwiz.blogspot.com/2012/06/donut-charts-are-worse-than-pie-charts.html]

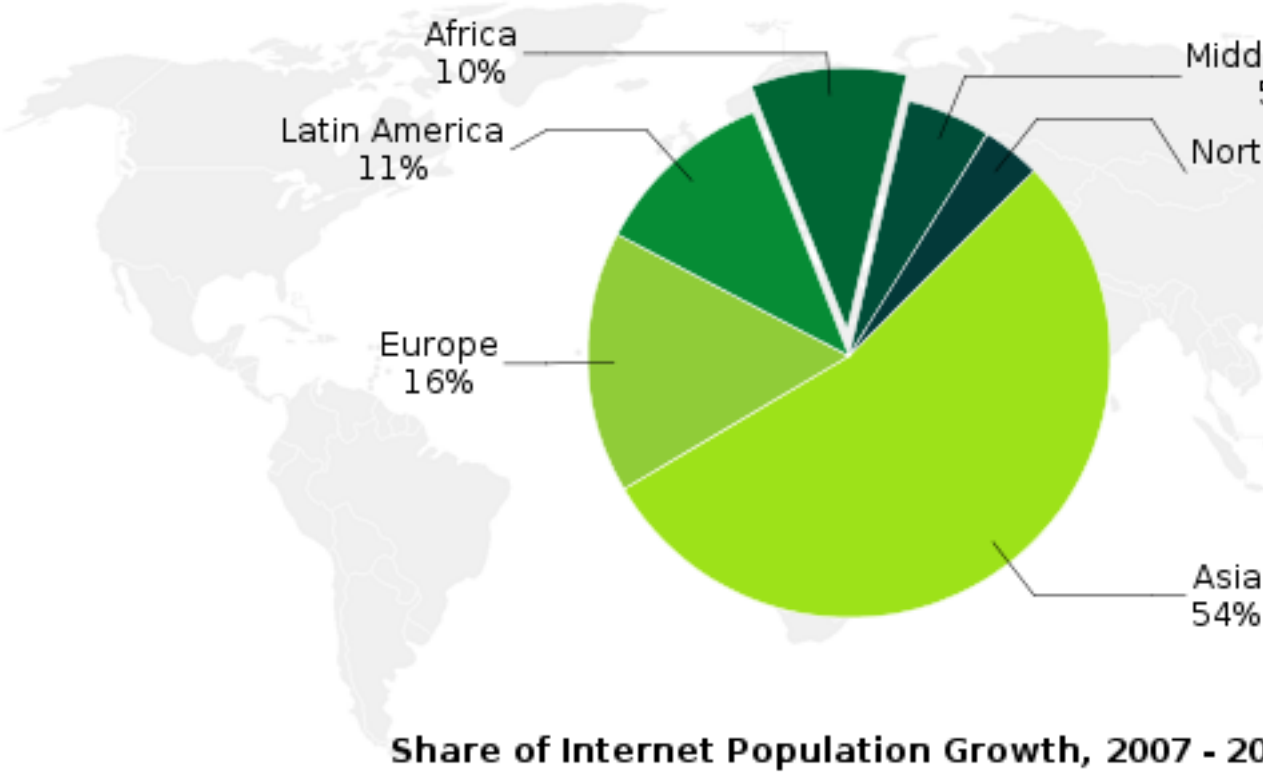
4.4.2.4.1. Pie charts

In order to create a pie chart, the jfreechart directive must be used with the following parameters:

Table 4.9. The jfreechart directive parameters for pie charts

Parameter	Value
type	pie
dataset	A sequence of the following form: [[**Section$_1$, Value$_1$**], ... [**Section$_n$, Value$_n$**]] where: • Section$_i$ is a text that represents the section name and • Value$_i$ is a number that represents the value associated with the section.
properties	The following properties are accepted: • general: chart, title, legend • plot: plot, pie plot

The following example creates a pie chart that shows the growth of internet population between 2007 and 2012.




```
[#assign pieDataSet = [
    ['Asia',          53.8],
    ['Europe',        16.1],
    ['Latin America', 11.3],
    ['Africa',         9.6],
```

```
    ['Middle East',    5.2],
    ['North America',  3.6]
] /]

[#assign chartProperties = {
    'title': 'Share of Internet Population Growth, 2007 - 2012',
    'backgroundImage': 'https://upload.wikimedia.org/wikipedia/commons/thumb/f/

    'title.position': 'bottom',
    'title.font': {'size': 12, 'weight': 'bold'},

    'legend.visible': false,

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.shadowPaint': '',

    'plot.startAngle': 45,
    'plot.baseSectionOutlinePaint': '#ffffff',
    'plot.sectionPaint':['#9de219', '#90cc38', '#068c35', '#006634', '#004d38',
    'plot.explodePercent': [0, 0, 0, 0.1, 0, 0],

    'plot.labelGenerator': '{0}\n{2}',
    'plot.labelBackgroundPaint': '',
    'plot.labelShadowPaint': '',
    'plot.labelOutlinePaint': '',
    'plot.labelLinkStyle': 'standard'
} /]

<img src="[@jfreechart type='pie' dataset=pieDataSet properties=chartProperties
```

4.4.2.4.2. Ring charts

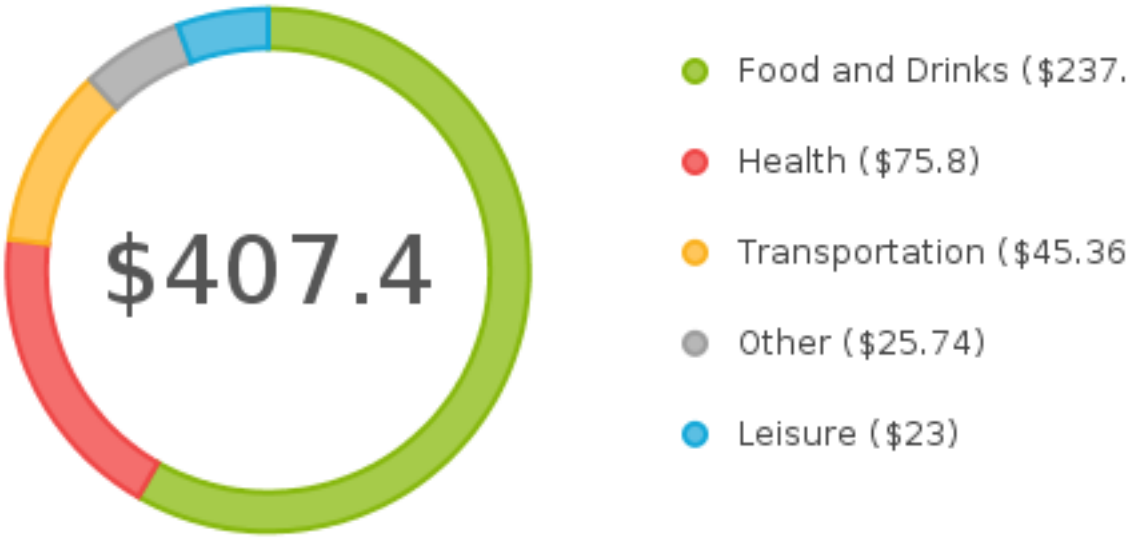
A ring chart is a pie chart with a hole in the center. In order to create a ring chart, the jfreechart directive must be used with the following parameters:

Table 4.10. The jfreechart directive parameters for ring charts

Parameter	Value
type	ring
dataset	A sequence of the following form: [[**Section$_1$, Value$_1$**], ... [**Section$_n$, Value$_n$**]] where: • Section$_i$ is a text that represents the section name and • Value$_i$ is a number that represents the value associated with the section.
properties	The following properties are accepted: • general: chart, title, legend

Parameter	Value
	• plot: plot, pie plot, ring plot

The following example creates a ring chart that shows the monthly expenses of a fictitious person.




```
[#assign pieDataSet = [
    ['Food and Drinks', 237.5],
    ['Health',           75.8],
    ['Transportation',  45.36],
    ['Other',           25.74],
    ['Leisure',          23.0]
] /]

[#assign chartProperties = {
    'title.visible': false,

    'legend.visible': true,
    'legend.position': 'right',
    'legend.verticalAlignment': 'top',
    'legend.frame': {"insets": [0, 0, 0, 0]},
    'legend.margin': [30, 0, 0, 0],
    'legend.itemLabelPadding': [10, 10, 10, 10],
    'plot.legendLabelGenerator': '{0} (${"$"}{1})',

    'plot.backgroundPaint': '',
    'plot.outlineVisible': false,
    'plot.shadowPaint': '',

    'plot.baseSectionOutlineStroke': {'width': 2.0},
    'plot.sectionOutlinePaint': ['#87b80e', '#ef4747', '#fab220', '#9e9e9e', '#
    'plot.sectionPaint':['#a6cb4a', '#f46e6d', '#ffc65b', '#b7b7b7', '#5bbfe3'],

    'plot.labelGenerator': '',

    'plot.centerTextMode': 'fixed',
    'plot.centerText': '$407.4',
```

```
    'plot.centerTextFont': {'size': 36},
    'plot.centerTextColor': '#555555',

    'plot.sectionDepth': 0.15,
    'plot.separatorsVisible': false
} /]

<img src="[@jfreechart type='ring' dataset=pieDataSet properties=chartProperties
```

4.4.3. Chart properties

Chart properties control the visual elements of a chart. All properties supported by the jfreechart directive are documented in this section. They are grouped by the major components of a chart (title, legend, plot, renderer, domain axis, range axis).

Properties have a name and a type. Their name is linked to the corresponding JFreeChart API method so you can read about the features they control. Here are a few properties and their corresponding JFreeChart methods:

- The 'title' property maps to the JFreeChart.setTitle(String) [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setTitle-java.lang.String-] method
- The 'legend.visible' property maps to the JFreeChart.getLegend().setVisible(boolean) [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setVisible-boolean-] method
- The 'plot.backgroundImage' property maps to the JFreeChart.getPlot().setBackgroundImage(Image) [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setBackgroundImage-java.awt.Image-] method

The type of a property defines the allowed values of a property. All types are documented in the following table. Property types are linked to the corresponding classes.

Property types that are followed by two brackets represent a sequence of elements of the specified type. For instance, **boolean[]** represents a sequence of boolean elements like **[true, true, false, true]**, while **Paint[]** represents a sequence of colors like **['#ff0000', '#ffffff', '#00ff00']**. These sequences are useful for configuring things like the colors of a pie chart (plot.sectionPaint).

Table 4.11. Property types

Type	Values	Example
AreaRendererEndType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AreaRendererEndType.html]	'level', 'taper' or 'truncate'	'plot.renderer.endType': 'level'
AxisLabelLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/AxisLabelLocation.html]	'low end', 'middle' or 'high end'	'plot.domainAxis.labelLocation': 'middle'
AxisLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/AxisLocation.html]	'top or left', 'top or right', 'bottom or left', 'bottom or right'	'plot.domainAxisLocation': 'top or left'
BlockFrame [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/LineBorder.html]	a hash with one or more of the following keys: • paint - a Paint property • stroke - a Stroke property	'legend.frame': {'paint': '#000000', 'insets': [10, 10, 10, 10]}

Type	Values	Example
	• insets - a RectangleInsets property	
boolean	true or false	'title.visible': false
CategoryAnchor [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAnchor.html]	'start', 'middle' or 'end'	'plot.domainGridlinePosition': 'middle'

CategoryItemLabelGenerator [http://www.jfree.org/jfreechart/api/gjdoc/org/jfree/chart/labels/StandardCategoryItemLabelGenerator.html]	a hash with none or some of the following keys:
	• labelFormat - a text that can use one or more of the following placeholders for a specific generator:
	  • {0} for series name,
	  • {1} for category value, and
	  • {2} for data value.
	• numberFormat - See DecimalFormat [http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html] for symbols that can be used to define the pattern.
	• dateFormat - See SimpleDateFormat [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html] for symbols that can be used to define the pattern.
	  dateFormat can't be used together with percentFormat and numberFormat.
	• percentFormat - See DecimalFormat [http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html] for symbols that can be used to define the pattern.
	  percentFormat can't be used tgether with dateFormat.
		'plot.renderer.baseItemLabelGenerator': {'labelFormat':'{2}', 'numberFormat':'#%'}

CategoryItemRenderer [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/CategoryItemRenderer.html]	a text with the fully qualified class name of the renderer	'plot.renderer': 'org.jfree.chart.renderer.category.LineAndShapeRe

Type	Values	Example
CategorySeriesLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/labels/StandardCategorySeriesLabelGenerator.html]	a text that should use the {0} placeholder	'plot.renderer.legendItemLabelGenerator': '{0}'
CenterTextMode [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CenterTextMode.html]	'fixed', 'none' or 'value'	'plot.centerTextMode': 'none'
Color	Colors can be specified as texts in rgb, rgba or hexadecimal formats. If the text is empty it means no color.	'plot.centerTextColor': '#ff0000' 'plot.centerTextColor': 'rgb(255, 0, 0)' 'plot.centerTextColor': 'rgba(255, 0, 0, 1.0)' 'plot.centerTextColor': ''
DatasetRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/DatasetRenderingOrder.html]	'forward' or 'reverse'	'plot.datasetRenderingOrder': 'forward'
Date [http://docs.oracle.com/javase/7/docs/api/java/util/Date.html]	A date or time value. See the Freemarker docs for more details on how to convert a text to a date [http://freemarker.org/docs/ref_builtins_string.html#ref_builtin_string_date].	'plot.domainAxis.maximumDate': '2015-01-01'?date.iso
DateFormat [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html]	See SimpleDateFormat [http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html] for symbols that can be used to define the pattern.	'plot.domainAxis.dateFormatOverride': 'MMM dd'
DateTickMarkPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateTickMarkPosition.html]	'start', 'middle' or 'end'	'plot.domainAxis.tickMarkPosition': 'middle'
DateTickUnit [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateTickUnit.html]	a hash with the following keys: • unitType - DateTickUnitType (mandatory) • multiple - a number > 0 (mandatory) • rollUnitType - DateTickUnitType (optional) • rollMultiple - a number > 0 (optional) • format - DateFormat (optional)	'plot.domainAxis.tickUnit': {'unitType': 'hour', 'multiple': 2}

Type	Values	Example
DateTickUnitType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateTickUnitType.html]	'year', 'month', 'day', 'hour', 'minute', 'second', 'millisecond'	'plot.domainAxis.tickUnit': {'unitType': 'hour', 'multiple': 2}
double	a number	'title.width': 30
HorizontalAlignment [http://www.jfree.org/jcommon/api/org/jfree/ui/HorizontalAlignment.html]	'left', 'right' or 'center'	'legend.horizontalAlignment': 'center'
Image	• file://pathtofile/file.png • http://www.somedomain.com/image.png • data URI [https://en.wikipedia.org/wiki/Data_URI_scheme]	'backgroundImage': 'file://pathtofile/file.png' 'backgroundImage': 'http://www.somedomain.com/image.png' 'backgroundImage': '...'
int	a number	'title.maximumLinesToDisplay': 1
ItemLabelAnchor [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/labels/ItemLabelAnchor.html]	'center', 'inside1', 'inside2', 'inside3', 'inside4', 'inside5', 'inside6', 'inside7', 'inside8', 'inside9', 'inside10', 'inside11', 'inside12', 'outside1', 'outside2', 'outside3', 'outside4', 'outside5', 'outside6', 'outside7', 'outside8', 'outside9', 'outside10', 'outside11', or 'outside12' See BarRenderer.calculateLabelAnchorPoint [http://www.jfree.org/jfreechart/api/javadoc/src-html/org/jfree/chart/renderer/AbstractRenderer.html#line.2758] to understand how each anchor is used to calculate the coordinates.	'plot.renderer.baseNegativeItemLabelPosition': ['inside9', 'center', 'center', 0]
ItemLabelPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/labels/ItemLabelPosition.html]	a sequence with four elements: [itemLabelAnchor, textAnchor, rotationAnchor, angle] where: • itemLabelAnchor - ItemLabelAnchor • textAnchor - TextAnchor • rotationAnchor - TextAnchor • angle - double	'plot.renderer.baseNegativeItemLabelPosition': ['inside9', 'center', 'center', 0]
float	a number	'backgroundImageAlpha': 0.9

Type	Values	Example
Font [http://docs.oracle.com/javase/7/docs/api/java/awt/Font.html]	a hash with one or more of the following keys: • name - a text • file - path to a ttf font file. path or name are mutually exclusive, only one of them can be used in a font definition. • size - a number • style - 'normal' or 'italic' • weight - 'normal' or 'bold'	'title.font': {'size': 20} 'title.font': {'name': 'sans-serif', 'size': 20, 'style': 'normal', 'weight': 'bold'} 'title.font': {'file': '/usr/share/fonts/truetype/freefont/FreeSans.ttf', 'size': 20}
Format [http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html]	See DecimalFormat [http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html] for symbols that can be used to define the pattern.	'plot.centerTextFormatter': '0.000'
Locale [http://docs.oracle.com/javase/7/docs/api/java/util/Locale.html]	See Locale [http://docs.oracle.com/javase/7/docs/api/java/util/Locale.html] for symbols that can be used to define the pattern.	'plot.domainAxis.locale': 'fr_FR'
NumberFormat [http://docs.oracle.com/javase/7/docs/api/java/text/NumberFormat.html]	See DecimalFormat [http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html] for symbols that can be used to define the pattern.	'plot.rangeAxis.numberFormatOverride': '0.000'
NumberTickUnit [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberTickUnit.html]	a hash with one or more of the following keys: • size - a number • format - a text • minorTickCount - a number	'plot.rangeAxis.tickUnit': {'size': 50} 'plot.rangeAxis.tickUnit': {'size': 50, 'format': '#0.00', 'minorTickCount': 5}
Paint	see Color	'title.paint': '#000000'
PieLabelLinkStyle [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PieLabelLinkStyle.html]	'standard', 'cubic curve' or 'quad curve'	'plot.labelLinkStyle': 'standard'
PieSectionLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/labels/StandardPieSectionLabelGenerator.html]	• " for no generator (hides labels) • a text that can use one or more of the following placeholders for a specific generator: • {0} for name,	'plot.labelGenerator': '{0}'

Type	Values	Example
	• {1} for actual value, • {2} for percentage, and • {3} for total value.	
PlotOrientation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PlotOrientation.html]	'vertical' or 'horizontal'	'plot.orientation': 'horizontal'
Point2D [http://docs.oracle.com/javase/7/docs/api/java/awt/Point.html]	[x, y]	'plot.quadrantOrigin': [0, 0]
Range [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/Range.html]	a sequence with two numbers or two dates: [lower, upper]	'plot.rangeAxis.defaultAutoRange': [0, 10] 'plot.domainAxis.range': ['10:30'?time.iso, '12:35'?time.iso]
RangeType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/RangeType.html]	'full', 'positive' or 'negative'	'plot.rangeAxis.rangeType': 'full'
RectangleAnchor [http://www.jfree.org/jcommon/api/org/jfree/ui/RectangleAnchor.html]	'top', 'top left', 'top right', 'center', 'left', 'right', 'bottom', 'bottom left', or 'bottom right'	'legend.legendItemGraphicLocation': 'center'
RectangleEdge [http://www.jfree.org/jcommon/api/org/jfree/ui/RectangleEdge.html]	'top', 'left', 'bottom', or 'right'	'legend.position': 'bottom'
RectangleInsets [http://www.jfree.org/jcommon/api/org/jfree/ui/RectangleInsets.html]	a sequence with four numbers: [top, left, bottom, right]	'padding': [0, 0 , 0, 0]
Rotation [http://www.jfree.org/jcommon/api/org/jfree/util/Rotation.html]	'clockwise' or 'anticlockwise'	'plot.direction': 'anticlockwise'
SeriesRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/SeriesRenderingOrder.html]	'forward' or 'reverse'	'plot.seriesRenderingOrder': 'forward'
Shape	One of the following shapes (see examples for parameters): 'up-triangle', 'down-triangle', 'left-triangle', 'right-triangle', 'diamond', 'regular-cross', 'diagonal-cross', 'rectangle', 'ellipse'	'plot.rangeAxis.upArrow': {'shape': 'up-triangle', 'size': 3} 'plot.rangeAxis.downArrow': {'shape': 'down-triangle', 'size': 3} 'plot.rangeAxis.leftArrow': {'shape': 'left-triangle', 'size': 3} 'plot.rangeAxis.rightArrow': {'shape': 'right-triangle', 'size': 3}

Type	Values	Example
		'plot.legendItemShape': {'shape':'diamond', 'size': 5} 'plot.legendItemShape': {'shape': 'regular-cross', 'length': 5, 'thickness': 1} 'plot.legendItemShape': {'shape': 'diagonal-cross', 'length': 5, 'thickness': 1} 'plot.legendItemShape': {'shape': 'rectangle', 'x': 0, 'y': 0, 'width': 5, 'height': 10} 'plot.legendItemShape': {'shape': 'ellipse', 'x': 0, 'y': 0, 'width': 10, 'height': 10},
SortOrder [http://www.jfree.org/jcommon/api/org/jfree/util/SortOrder.html]	'ascending' or 'descending'	'legend.sortOrder': 'ascending'
String	a text	'title': 'Title'
Stroke [http://docs.oracle.com/javase/7/docs/api/java/awt/BasicStroke.html]	a hash with one or more of the following keys (click the type link for more details): • width - a number • cap - a number (0, 1, or 2) • join - a number (0, 1, or 2) • miterlimit - a number • dash[] - a sequence of numbers • dash_phase - a number	'borderStroke': {'width': 1.0} 'borderStroke': {'width': 1.0, 'cap': 0, 'join': 2, 'miterlimit': 0.0, 'dash': [2.0, 2.0], 'dash_phase': 0.0}
TextAnchor [http://www.jfree.org/jcommon/api/org/jfree/ui/TextAnchor.html]	'top left', 'top center', 'top right', 'half ascent left', 'half ascent center', 'half ascent right', 'center left', 'center', 'center right', 'baseline left', 'baseline center', 'baseline right', 'bottom left', 'bottom center', or 'bottom right'	'plot.renderer.baseNegativeItemLabelPosition': ['inside9', 'center', 'center', 0]
TimeZone [http://docs.oracle.com/javase/7/docs/api/java/util/TimeZone.html]	See TimeZone [http://docs.oracle.com/javase/7/docs/api/java/util/TimeZone.html#getTimeZone(java.lang.String)] for codes that can be used to define the timezone.	'plot.domainAxis.timezone': 'GMT'
ValueAxis [http://www.jfree.org/jfreechart/api/	a text with the fully qualified class name of the axis	'plot.rangeAxis': 'org.jfree.chart.axis.NumberAxis'

Type	Values	Example
javadoc/org/jfree/chart/axis/ ValueAxis.html]		
VerticalAlignment [http:// www.jfree.org/jcommon/api/ org/jfree/ui/ VerticalAlignment.html]	'top', 'center' or 'bottom'	'legend.verticalAlignment': 'center'
XYItemLabelGenerator [http:// www.jfree.org/jfreechart/api/ javadoc/org/jfree/chart/labels/ StandardXYItemLabelGenerator.html]	• " for no generator (hides labels) • a text that can use one or more of the following placeholders for a specific generator: • {0} for series name, • {1} for domain value, and • {2} for range value.	'plot.renderer.baseItemLabelGenerator': '{2}'
XYItemRenderer [http:// www.jfree.org/jfreechart/api/ javadoc/org/jfree/chart/ renderer/xy/ XYItemRenderer.html]	a text with the fully qualified class name of the renderer	'plot.renderer': 'org.jfree.chart.renderer.xy.XYLineAndShapeRend
XYSeriesLabelGenerator [http://www.jfree.org/jfreechart/ api/javadoc/org/jfree/chart/ labels/ StandardXYSeriesLabelGenerator.html]	a text that should use the {0} placeholder	'plot.renderer.legendItemLabelGenerator': '{0}'

4.4.3.1. Chart

The following properties are available for all types of charts.

Table 4.12. JFreeChart properties

Name	Type	Example
antiAlias [http://www.jfree.org/ jfreechart/api/javadoc/org/jfree/ chart/ JFreeChart.html#setAntiAlias-boolean-]	boolean	'antiAlias': true
backgroundImage [http:// www.jfree.org/jfreechart/api/ javadoc/org/jfree/chart/ JFreeChart.html#setBackgroundImage-java.awt.Image-]	Image	'backgroundImage': 'http:// www.somedomain.com/ image.png'
backgroundImageAlignment [http://www.jfree.org/jfreechart/ api/javadoc/org/jfree/chart/ JFreeChart.html#setBackgroundImageAlignment-int-]	int	The number must be the value of one of the align constants. The value zero [http://www.jfree.org/ jcommon/api/constant-values.html#org.jfree.ui.Align.CENTER] stands for center alignment.

Name	Type	Example
		• CENTER 0, LEFT 4, RIGHT 8 • TOP 1, TOP_LEFT 5, TOP_RIGHT 9 • BOTTOM 2, BOTTOM_LEFT 6, BOTTOM_RIGHT 10 • FIT 15, FIT_HORIZONTAL 12, FIT_VERTICAL 3 'backgroundImageAlignment': 0
backgroundImageAlpha [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setBackgroundImageAlpha-float-]	float	'backgroundImageAlpha': 0.9
backgroundPaint [hhttp://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setBackgroundPaint-java.awt.Paint-]	Paint	'backgroundPaint': ''
borderPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setBorderPaint-java.awt.Paint-]	Paint	'borderPaint': '#ff0000'
borderStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setBorderStroke-java.awt.Stroke-]	Stroke	'borderStroke': {'width': 1.0}
borderVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setBorderVisible-boolean-]	boolean	'borderVisible': true
padding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setPadding-org.jfree.ui.RectangleInsets-]	RectangleInsets	'padding': [0, 0 , 0, 0]
subtitles [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setSubtitles-java.util.List-]	List	'subtitles': ['Source: Box Office Mojo'] See the title section below for details on how to configure the properties of a subtitle. For instance, to change the font of the first subtitle use the 'subtitle[0].font' property.

Name	Type	Example
textAntiAlias [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setTextAntiAlias-boolean-]	boolean	'textAntiAlias': true
title [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/JFreeChart.html#setTitle-java.lang.String-]	String	'title': 'Title'

4.4.3.2. Title and subtitles

The following properties configure the title and the subtitles of all types of charts.

To access the properties of a subtitle, use subtitle[index] (where index starts at zero). For instance, to access the font property of the first subtitle use 'subtitle[0].font'.

Table 4.13. Title properties

Name	Type	Example
title.backgroundPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setBackgroundPaint-java.awt.Paint-]	Paint	'title.backgroundPaint': ''
title.expandToFitSpace [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setExpandToFitSpace-boolean-]	boolean	'title.expandToFitSpace': true
title.font [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setFont-java.awt.Font-]	Font	'title.font': {'size': 20}
title.frame [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setFrame-org.jfree.chart.block.BlockFrame-]	BlockFrame	'title.frame': {"insets": [10, 10, 10, 10]}
title.height [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setHeight-double-]	double	'title.height': 30
title.horizontalAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setHorizontalAlignment-org.jfree.ui.HorizontalAlignment-]	HorizontalAlignment	'title.horizontalAlignment': 'center'
title.margin [http://www.jfree.org/jfreechart/api/	RectangleInsets	'title.margin': [0, 0, 0, 0]

Name	Type	Example
javadoc/org/jfree/chart/block/AbstractBlock.html#setMargin-double-double-double-double-]		
title.maximumLinesToDisplay [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setMaximumLinesToDisplay-int-]	int	'title.maximumLinesToDisplay': 1
title.padding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setPadding-double-double-double-double-]	RectangleInsets	'title.padding': [10, 10, 10, 10]
title.paint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setPaint-java.awt.Paint-]	Paint	'title.paint': '#000000'
title.position [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setPosition-org.jfree.ui.RectangleEdge-]	RectangleEdge	'title.position': 'top'
title.text [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setText-java.lang.String-]	String	'title.text': 'Title' same as 'title': 'Title'.
title.textAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/TextTitle.html#setTextAlignment-org.jfree.ui.HorizontalAlignment-]	HorizontalAlignment	'title.textAlignment': 'left'
title.verticalAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setVerticalAlignment-org.jfree.ui.VerticalAlignment-]	VerticalAlignment	'title.verticalAlignment': 'center'
title.visible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setVisible-boolean-]	boolean	'title.visible': false
title.width [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setWidth-double-]	double	'title.width': 30

4.4.3.3. Legend

The following properties configure the legend of a chart and they are available for all types of charts.

Table 4.14. LegendTitle properties

Name	Type	Example
legend.backgroundPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setBackgroundPaint-java.awt.Paint-]	Paint	'legend.backgroundPaint': ''
legend.frame [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setFrame-org.jfree.chart.block.BlockFrame-]	BlockFrame	'legend.frame': {"insets": [10, 10, 10, 10]}
legend.height [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setHeight-double-]	double	'legend.height': 30
legend.horizontalAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setHorizontalAlignment-org.jfree.ui.HorizontalAlignment-]	HorizontalAlignment	'legend.horizontalAlignment': 'center'
legend.itemFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setItemFont-java.awt.Font-]	Font	'legend.itemFont': {'size': 20}
legend.itemLabelPadding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setItemLabelPadding-org.jfree.ui.RectangleInsets-]	RectangleInsets	'legend.itemLabelPadding': [10, 10, 10, 10]
legend.itemPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setItemPaint-java.awt.Paint-]	Paint	'legend.itemPaint': '#000000'
legend.legendItemGraphicAnchor [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setLegendItemGraphicAnchor-org.jfree.ui.RectangleAnchor-]	RectangleAnchor	'legend.legendItemGraphicAnchor': 'center'
legend.legendItemGraphicEdge [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setLegendItemGraphicEdge-org.jfree.ui.RectangleEdge-]	RectangleEdge	'legend.legendItemGraphicEdge': 'left'
legend.legendItemGraphicLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/	RectangleAnchor	'legend.legendItemGraphicLocation': 'center'

Name	Type	Example
LegendTitle.html#setLegendItemGraphicLocation-org.jfree.ui.RectangleAnchor-]		
legend.legendItemGraphicPadding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setLegendItemGraphicPadding-org.jfree.ui.RectangleInsets-]	RectangleInsets	'legend.legendItemGraphicPadding': [0, 0, 0, 0]
legend.margin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setMargin-double-double-double-double-]	RectangleInsets	'legend.margin': [0, 0, 0, 0]
legend.padding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setPadding-double-double-double-double-]	RectangleInsets	'legend.padding': [10, 10, 10, 10]
legend.position [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setPosition-org.jfree.ui.RectangleEdge-]	RectangleEdge	'legend.position': 'bottom'
legend.sortOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/LegendTitle.html#setSortOrder-org.jfree.util.SortOrder-]	SortOrder	'legend.sortOrder': 'ascending'
legend.verticalAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setVerticalAlignment-org.jfree.ui.VerticalAlignment-]	VerticalAlignment	'legend.verticalAlignment': 'center'
legend.visible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/title/Title.html#setVisible-boolean-]	boolean	'legend.visible': false
legend.width [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/block/AbstractBlock.html#setWidth-double-]	double	'legend.width': 30

4.4.3.4. Plot

The following properties are common to all charts. Each chart uses a specialized plot that has specific properties. For instance, pie charts use a PiePlot that has these basic properties and a few more (documented below).

Table 4.15. Plot properties

Name	Type	Example
plot.backgroundAlpha [http://www.jfree.org/jfreechart/api/	float	'plot.backgroundAlpha': 0.9

Name	Type	Example
javadoc/org/jfree/chart/plot/Plot.html#setBackgroundAlpha-float-]		
plot.backgroundImage [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setBackgroundImage-java.awt.Image-]	Image	'plot.backgroundImage': 'http://www.somedomain.com/image.png'
plot.backgroundImageAlignment [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setBackgroundImageAlignment-int-]	int	The number must be the value of one of the align constants. The value zero [http://www.jfree.org/jcommon/api/constant-values.html#org.jfree.ui.Align.CENTER] stands for center alignment. • CENTER 0, LEFT 4, RIGHT 8 • TOP 1, TOP_LEFT 5, TOP_RIGHT 9 • BOTTOM 2, BOTTOM_LEFT 6, BOTTOM_RIGHT 10 • FIT 15, FIT_HORIZONTAL 12, FIT_VERTICAL 3 'plot.backgroundImageAlignment': 0
plot.backgroundImageAlpha [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setBackgroundImageAlpha-float-]	float	'plot.backgroundImageAlpha': 0.9
plot.backgroundPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setBackgroundPaint-java.awt.Paint-]	Paint	'plot.backgroundPaint': '#ff0000'
plot.foregroundAlpha [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setForegroundAlpha-float-]	float	'plot.foregroundAlpha': 1.0
plot.noDataMessage [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setNoDataMessage-java.lang.String-]	String	'plot.noDataMessage': 'No data available'
plot.noDataMessageFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/	Font	'plot.noDataMessageFont': {'size': 10}

Name	Type	Example
Plot.html#setNoDataMessageFont-java.awt.Font-]		
plot.noDataMessagePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setNoDataMessagePaint-java.awt.Paint-]	Paint	'plot.noDataMessagePaint': '#ff0000'
plot.outlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setOutlinePaint-java.awt.Paint-]	Paint	'plot.outlinePaint': '#ffffff'
plot.outlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setOutlineStroke-java.awt.Stroke-]	Stroke	'plot.outlineStroke': {'width': 1.0}
plot.outlineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/Plot.html#setOutlineVisible-boolean-]	boolean	'plot.outlineVisible': true

The following properties are common to pie charts and ring charts.

Table 4.16. PiePlot properties

Name	Type	Example
plot.autoPopulateSectionOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setAutoPopulateSectionOutlinePaint-boolean-]	boolean	'plot.autoPopulateSectionOutlinePaint': true
plot.autoPopulateSectionOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setAutoPopulateSectionOutlineStroke-boolean-]	boolean	'plot.autoPopulateSectionOutlineStroke': true
plot.autoPopulateSectionPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setAutoPopulateSectionPaint-boolean-]	boolean	'plot.autoPopulateSectionPaint': true
plot.baseSectionOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setBaseSectionOutlinePaint-java.awt.Paint-]	Paint	'plot.baseSectionOutlinePaint': '#ffffff'
plot.baseSectionOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setBaseSectionOutlineStroke-java.awt.Stroke-]	Stroke	'plot.baseSectionOutlineStroke': {'width': 1.0}

Name	Type	Example
plot.baseSectionPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setBaseSectionPaint-java.awt.Paint-]	Paint	'plot.baseSectionPaint': '#ffffff'
plot.circular [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setCircular-boolean-]	boolean	'plot.circular': true
plot.direction [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setDirection-org.jfree.util.Rotation-]	Rotation	'plot.direction': 'anticlockwise'
plot.explodePercent [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setExplodePercent-int-double-]	double[]	'plot.explodePercent': [0.10,0,0]
plot.ignoreNullValues [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setIgnoreNullValues-boolean-]	boolean	'plot.ignoreNullValues': true
plot.ignoreZeroValues [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setIgnoreZeroValues-boolean-]	boolean	'plot.ignoreZeroValues': true
plot.interiorGap [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setInteriorGap-double-]	double	'plot.interiorGap': 0.1
plot.labelBackgroundPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelBackgroundPaint-java.awt.Paint-]	Paint	'plot.labelBackgroundPaint': '#ffffff'
plot.labelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelFont-java.awt.Font-]	Font	'plot.labelFont': {'size': 12}
plot.labelGap [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelGap-double-]	double	'plot.labelGap': 0.1
plot.labelGenerator [http://www.jfree.org/jfreechart/api/	PieSectionLabelGenerator	'plot.labelGenerator': '{0}'

Name	Type	Example
javadoc/org/jfree/chart/plot/PiePlot.html#setLabelGenerator-org.jfree.chart.labels.PieSectionLabelGenerator-]		
plot.labelLinkMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelLinkMargin-double-]	double	'plot.labelLinkMargin': 5
plot.labelLinkPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelLinkPaint-java.awt.Paint-]	Paint	'plot.labelLinkPaint': '#ffffff'
plot.labelLinkStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelLinkStroke-java.awt.Stroke-]	Stroke	'plot.labelLinkStroke': {'width': 1.0}
plot.labelLinkStyle [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelLinkStyle-org.jfree.chart.plot.PieLabelLinkStyle-]	PieLabelLinkStyle	'plot.labelLinkStyle': 'standard'
plot.labelLinksVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelLinksVisible-boolean-]	boolean	'plot.labelLinksVisible': true
plot.labelOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelOutlinePaint-java.awt.Paint-]	Paint	'plot.labelOutlinePaint': '#ffffff'
plot.labelOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelOutlineStroke-java.awt.Stroke-]	Stroke	'plot.labelOutlineStroke': {'width': 1.0}
plot.labelPadding [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelPadding-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.labelPadding': [10, 10, 10, 10]
plot.labelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLabelPaint-java.awt.Paint-]	Paint	'plot.labelPaint': '#ffffff'
plot.labelShadowPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/	Paint	'plot.labelShadowPaint': '#000000'

Name	Type	Example
PiePlot.html#setLabelShadowPaint-java.awt.Paint-]		
plot.legendItemShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLegendItemShape-java.awt.Shape-]	Shape	'plot.legendItemShape': {'shape':'diamond', 'size': 5}
plot.legendLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setLegendLabelGenerator-org.jfree.chart.labels.PieSectionLabelGenerator-]	PieSectionLabelGenerator	'plot.legendLabelGenerator': '{0}'
plot.maximumLabelWidth [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setMaximumLabelWidth-double-]	double	'plot.maximumLabelWidth': 0.20
plot.minimumArcAngleToDraw [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setMinimumArcAngleToDraw-double-]	double	'plot.minimumArcAngleToDraw': 5
plot.sectionOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSectionOutlinePaint-int-java.awt.Paint-]	Paint[]	'plot.sectionOutlinePaint': ['#ffffff','#ffffff']
plot.sectionOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSectionOutlineStroke-int-java.awt.Stroke-]	Stroke[]	'plot.sectionOutlineStroke': [{'width': 1.0},{'width': 1.0}]
plot.sectionOutlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSectionOutlinesVisible-boolean-]	boolean	'plot.sectionOutlinesVisible': true
plot.sectionPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSectionPaint-int-java.awt.Paint-]	Paint[]	'plot.sectionPaint': ['#ffffff','#ffffff']
plot.shadowPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setShadowPaint-java.awt.Paint-]	Paint	'plot.shadowPaint': '#000000'
plot.shadowXOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/	double	'plot.shadowXOffset': 2

Name	Type	Example
PiePlot.html#setShadowXOffset-double-]		
plot.shadowYOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setShadowYOffset-double-]	double	'plot.shadowYOffset': 2
plot.simpleLabelOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSimpleLabelOffset-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.simpleLabelOffset': [1, 1, 1, 1]
plot.simpleLabels [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setSimpleLabels-boolean-]	boolean	'plot.simpleLabels': true
plot.startAngle [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/PiePlot.html#setStartAngle-double-]	double	Angle value is expressed in degrees. 'plot.startAngle': 90

The following properties are specific to ring charts.

Table 4.17. RingPlot properties

Name	Type	Example
plot.centerText [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setCenterText-java.lang.String-]	String	Use this if 'plot.centerTextMode' is set to 'fixed'. 'plot.centerTextMode': 'fixed' 'plot.centerText': 'Text'
plot.centerTextColor [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setCenterTextColor-java.awt.Color-]	Color	'plot.centerTextColor': '#222222'
plot.centerTextFormatter [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setCenterTextFormatter-java.text.Format-]	Format	Use this if 'plot.centerTextMode' is set to 'value'. It formats the value of the first item from the dataset. 'plot.centerTextMode': 'value' 'plot.centerTextFormatter': '0.000'
plot.centerTextFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setCenterTextFont-java.awt.Font-]	Font	'plot.centerTextFont': {'size': 20}

Name	Type	Example
plot.centerTextMode [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setCenterTextMode-org.jfree.chart.plot.CenterTextMode-]	CenterTextMode	'plot.centerTextMode': 'none' 'plot.centerTextMode': 'value' 'plot.centerTextMode': 'fixed'
plot.innerSeparatorExtension [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setInnerSeparatorExtension-double-]	double	'plot.innerSeparatorExtension': 0.50
plot.outerSeparatorExtension [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setOuterSeparatorExtension-double-]	double	'plot.outerSeparatorExtension': 0.50
plot.sectionDepth [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setSectionDepth-double-]	double	'plot.sectionDepth': 0.45
plot.separatorPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setSeparatorPaint-java.awt.Paint-]	Paint	'plot.separatorPaint': '#ff0000'
plot.separatorStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setSeparatorStroke-java.awt.Stroke-]	Stroke	'plot.separatorStroke': ''
plot.separatorsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/RingPlot.html#setSeparatorsVisible-boolean-]	boolean	'plot.separatorsVisible': true

The following properties are common to charts that use a category dataset (with texts on the domain axis).

Table 4.18. CategoryPlot properties

Name	Type	Example
plot.anchorValue [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setAnchorValue-double-]	double	'plot.anchorValue': 10
plot.axisOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setAxisOffset-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.axisOffset': [0, 0, 0, 0]

Name	Type	Example
plot.columnRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setColumnRenderingOrder-org.jfree.util.SortOrder-]	SortOrder	'plot.columnRenderingOrder': 'ascending'
plot.crosshairDatasetIndex [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setCrosshairDatasetIndex-int-]	int	'plot.crosshairDatasetIndex': 0
plot.datasetRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDatasetRenderingOrder-org.jfree.chart.plot.DatasetRenderingOrder-]	DatasetRenderingOrder	'plot.datasetRenderingOrder': 'forward'
plot.domainAxisLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainAxisLocation-org.jfree.chart.axis.AxisLocation-]	AxisLocation	'plot.domainAxisLocation': 'bottom or right'
plot.domainCrosshairPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainCrosshairPaint-java.awt.Paint-]	Paint	'plot.domainCrosshairPaint': '#ff0000'
plot.domainCrosshairStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainCrosshairStroke-java.awt.Stroke-]	Stroke	'plot.domainCrosshairStroke': {'width': 1.0}
plot.domainCrosshairVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainCrosshairVisible-boolean-]	boolean	'plot.domainCrosshairVisible': true
plot.domainGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainGridlinePaint-java.awt.Paint-]	Paint	'plot.domainGridlinePaint': '#ffffff'
plot.domainGridlinePosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainGridlinePosition-org.jfree.chart.axis.CategoryAnchor-]	CategoryAnchor	'plot.domainGridlinePosition': 'middle'
plot.domainGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainGridlineStroke-java.awt.Stroke-]	Stroke	'plot.domainGridlineStroke': {'width': 1.0}

Name	Type	Example
plot.domainGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDomainGridlinesVisible-boolean-]	boolean	'plot.domainGridlinesVisible': true
plot.drawSharedDomainAxis [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setDrawSharedDomainAxis-boolean-]	boolean	'plot.drawSharedDomainAxis': true
plot.mapDatasetToRangeAxis [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#mapDatasetToRangeAxis-int-int-]	int[]	The number of parameters depends on the number of datasets set on the chart. The following example is for a chart with two datasets and two range axes. 'plot.mapDatasetToRangeAxis': [0, 1]
plot.orientation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setOrientation-org.jfree.chart.plot.PlotOrientation-]	PlotOrientation	'plot.orientation': 'horizontal'
plot.renderer [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRenderer-org.jfree.chart.renderer.category.CategoryItemRenderer-]	CategoryItemRenderer	'plot.renderer': 'org.jfree.chart.renderer.category.LineAndShapeRe
plot.rangeAxis [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeAxis-org.jfree.chart.axis.ValueAxis-]	ValueAxis	'plot.rangeAxis': 'org.jfree.chart.axis.NumberAxis'
plot.rangeAxisLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeAxisLocation-org.jfree.chart.axis.AxisLocation-]	AxisLocation	'plot.rangeAxisLocation': 'top or left'
plot.rangeCrosshairLockedOnData [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeCrosshairLockedOnData-boolean-]	boolean	'plot.rangeCrosshairLockedOnData': true
plot.rangeCrosshairPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeCrosshairPaint-java.awt.Paint-]	Paint	'plot.rangeCrosshairPaint': '#ffffff'

Name	Type	Example
plot.rangeCrosshairStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeCrosshairStroke-java.awt.Stroke-]	Stroke	'plot.rangeCrosshairStroke': {'width': 1.0}
plot.rangeCrosshairValue [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeCrosshairValue-double-]	double	'plot.rangeCrosshairValue': 5
plot.rangeCrosshairVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeCrosshairVisible-boolean-]	boolean	'plot.rangeCrosshairVisible': true
plot.rangeGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeGridlinePaint-java.awt.Paint-]	Paint	'plot.rangeGridlinePaint': '#ffffff'
plot.rangeGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeGridlineStroke-java.awt.Stroke-]	Stroke	'plot.rangeGridlineStroke': {'width': 1.0}
plot.rangeGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeGridlinesVisible-boolean-]	boolean	'plot.rangeGridlinesVisible': true
plot.rangeMinorGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeMinorGridlinePaint-java.awt.Paint-]	Paint	'plot.rangeMinorGridlinePaint': '#ffffff'
plot.rangeMinorGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeMinorGridlineStroke-java.awt.Stroke-]	Stroke	'plot.rangeMinorGridlineStroke': {'width': 1.0}
plot.rangeMinorGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeMinorGridlinesVisible-boolean-]	boolean	'plot.rangeMinorGridlinesVisible': true
plot.rangePannable [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangePannable-boolean-]	boolean	'plot.rangePannable': true
plot.rangeZeroBaselinePaint [http://www.jfree.org/jfreechart/	Paint	'plot.rangeZeroBaselinePaint': '#ffffff'

Name	Type	Example
api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeZeroBaselinePaint-java.awt.Paint-]		
plot.rangeZeroBaselineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeZeroBaselineStroke-java.awt.Stroke-]	Stroke	'plot.rangeZeroBaselineStroke': {'width': 1.0}
plot.rangeZeroBaselineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRangeZeroBaselineVisible-boolean-]	boolean	'plot.rangeZeroBaselineVisible': true
plot.rowRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/CategoryPlot.html#setRowRenderingOrder-org.jfree.util.SortOrder-]	SortOrder	'plot.rowRenderingOrder': 'ascending'

The following properties are common to charts that use an xy dataset (with dates or numbers on the domain axis). Please note that range [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#addRangeMarker-int-org.jfree.chart.plot.Marker-org.jfree.ui.Layer-] and domain [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#addDomainMarker-int-org.jfree.chart.plot.Marker-org.jfree.ui.Layer-] markers are not supported.

Table 4.19. XYPlot properties

Name	Type	Example
plot.axisOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setAxisOffset-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.axisOffset': [0, 0, 0, 0]
plot.datasetRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDatasetRenderingOrder-org.jfree.chart.plot.DatasetRenderingOrder-]	DatasetRenderingOrder	'plot.datasetRenderingOrder': 'forward'
plot.domainAxisLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainAxisLocation-org.jfree.chart.axis.AxisLocation-]	AxisLocation	'plot.domainAxisLocation': 'bottom or right'
plot.domainCrosshairLockedOnData [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainCrosshairLockedOnData-boolean-]	boolean	'plot.domainCrosshairLockedOnData': true
plot.domainCrosshairPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/	Paint	'plot.domainCrosshairPaint': '#ff0000'

Name	Type	Example
XYPlot.html#setDomainCrosshairPaint-java.awt.Paint-]		
plot.domainCrosshairStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainCrosshairStroke-java.awt.Stroke-]	Stroke	'plot.domainCrosshairStroke': {'width': 1.0}
plot.domainCrosshairValue [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainCrosshairValue-double-]	double	'plot.domainCrosshairValue': 5
plot.domainCrosshairVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainCrosshairVisible-boolean-]	boolean	'plot.domainCrosshairVisible': true
plot.domainGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainGridlinePaint-java.awt.Paint-]	Paint	'plot.domainGridlinePaint': '#ffffff'
plot.domainGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainGridlineStroke-java.awt.Stroke-]	Stroke	'plot.domainGridlineStroke': {'width': 1.0}
plot.domainGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainGridlinesVisible-boolean-]	boolean	'plot.domainGridlinesVisible': true
plot.domainMinorGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainMinorGridlinePaint-java.awt.Paint-]	Paint	'plot.domainMinorGridlinePaint': '#ffffff'
plot.domainMinorGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainMinorGridlineStroke-java.awt.Stroke-]	Stroke	'plot.domainMinorGridlineStroke': {'width': 1.0}
plot.domainMinorGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainMinorGridlinesVisible-boolean-]	boolean	'plot.domainMinorGridlinesVisible': true
plot.domainPannable [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainPannable-boolean-]	boolean	'plot.domainPannable': true

Name	Type	Example
plot.domainTickBandPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainTickBandPaint-java.awt.Paint-]	Paint	'plot.domainTickBandPaint': '#ff0000'
plot.domainZeroBaselinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainZeroBaselinePaint-java.awt.Paint-]	Paint	'plot.domainZeroBaselinePaint': '#ffffff'
plot.domainZeroBaselineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainZeroBaselineStroke-java.awt.Stroke-]	Stroke	'plot.domainZeroBaselineStroke': {'width': 1.0}
plot.domainZeroBaselineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setDomainZeroBaselineVisible-boolean-]	boolean	'plot.domainZeroBaselineVisible': true
plot.mapDatasetToRangeAxis [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#mapDatasetToRangeAxis-int-int-]	int[]	The number of parameters depends on the number of datasets set on the chart. The following example is for a chart with two datasets and two range axes. 'plot.mapDatasetToRangeAxis': [0, 1]
plot.orientation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setOrientation-org.jfree.chart.plot.PlotOrientation-]	PlotOrientation	'plot.orientation': 'horizontal'
plot.quadrantOrigin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setQuadrantOrigin-java.awt.geom.Point2D-]	Point2D	'plot.quadrantOrigin': [0, 0]
plot.quadrantPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setQuadrantPaint-int-java.awt.Paint-]	Paint[]	'plot.quadrantPaint': ['#ff0000', '#00ff00', '#0000ff']
plot.renderer [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRenderer-org.jfree.chart.renderer.xy.XYItemRenderer-]	XYItemRenderer	'plot.renderer': 'org.jfree.chart.renderer.xy.XYLineAndShapeRend '
plot.rangeAxis [http://www.jfree.org/jfreechart/api/	ValueAxis	'plot.rangeAxis': 'org.jfree.chart.axis.NumberAxis'

Name	Type	Example
javadoc/org/jfree/chart/plot/XYPlot.html#setRangeAxis-org.jfree.chart.axis.ValueAxis-]		
plot.rangeAxisLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeAxisLocation-org.jfree.chart.axis.AxisLocation-]	AxisLocation	'plot.rangeAxisLocation': 'top or left'
plot.rangeCrosshairLockedOnData [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeCrosshairLockedOnData-boolean-]	boolean	'plot.rangeCrosshairLockedOnData': true
plot.rangeCrosshairPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeCrosshairPaint-java.awt.Paint-]	Paint	'plot.rangeCrosshairPaint': '#ffffff'
plot.rangeCrosshairStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeCrosshairStroke-java.awt.Stroke-]	Stroke	'plot.rangeCrosshairStroke': {'width': 1.0}
plot.rangeCrosshairValue [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeCrosshairValue-double-]	double	'plot.rangeCrosshairValue': 5
plot.rangeCrosshairVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeCrosshairVisible-boolean-]	boolean	'plot.rangeCrosshairVisible': true
plot.rangeGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeGridlinePaint-java.awt.Paint-]	Paint	'plot.rangeGridlinePaint': '#ffffff'
plot.rangeGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeGridlineStroke-java.awt.Stroke-]	Stroke	'plot.rangeGridlineStroke': {'width': 1.0}
plot.rangeGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeGridlinesVisible-boolean-]	boolean	'plot.rangeGridlinesVisible': true
plot.rangeMinorGridlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/	Paint	'plot.rangeMinorGridlinePaint': '#ffffff'

Name	Type	Example
XYPlot.html#setRangeMinorGridlinePaint-java.awt.Paint-]		
plot.rangeMinorGridlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeMinorGridlineStroke-java.awt.Stroke-]	Stroke	'plot.rangeMinorGridlineStroke': {'width': 1.0}
plot.rangeMinorGridlinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeMinorGridlinesVisible-boolean-]	boolean	'plot.rangeMinorGridlinesVisible': true
plot.rangePannable [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangePannable-boolean-]	boolean	'plot.rangePannable': true
plot.rangeTickBandPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeTickBandPaint-java.awt.Paint-]	Paint	'plot.rangeTickBandPaint': '#ff0000'
plot.rangeZeroBaselinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeZeroBaselinePaint-java.awt.Paint-]	Paint	'plot.rangeZeroBaselinePaint': '#ffffff'
plot.rangeZeroBaselineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeZeroBaselineStroke-java.awt.Stroke-]	Stroke	'plot.rangeZeroBaselineStroke': {'width': 1.0}
plot.rangeZeroBaselineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setRangeZeroBaselineVisible-boolean-]	boolean	'plot.rangeZeroBaselineVisible': true
plot.seriesRenderingOrder [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/plot/XYPlot.html#setSeriesRenderingOrder-org.jfree.chart.plot.SeriesRenderingOrder-]	SeriesRenderingOrder	'plot.seriesRenderingOrder': 'forward'

4.4.3.5. Renderer

The following properties are common to all charts. Each chart uses a specialized renderer that has specific properties. Renderers control the appearance of a chart.

Table 4.20. AbstractRenderer properties

Name	Type	Example
plot.renderer.autoPopulateSeriesFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesFillPaint-boolean-]	boolean	'plot.renderer.autoPopulateSeriesFillPaint': true
plot.renderer.autoPopulateSeriesOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesOutlinePaint-boolean-]	boolean	'plot.renderer.autoPopulateSeriesOutlinePaint': true
plot.renderer.autoPopulateSeriesOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesOutlineStroke-boolean-]	boolean	'plot.renderer.autoPopulateSeriesOutlineStroke': true
plot.renderer.autoPopulateSeriesPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesPaint-boolean-]	boolean	'plot.renderer.autoPopulateSeriesPaint': true
plot.renderer.autoPopulateSeriesShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesShape-boolean-]	boolean	'plot.renderer.autoPopulateSeriesShape': true
plot.renderer.autoPopulateSeriesStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setAutoPopulateSeriesStroke-boolean-]	boolean	'plot.renderer.autoPopulateSeriesStroke': true
plot.renderer.baseCreateEntities [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseCreateEntities-boolean-]	boolean	'plot.renderer.baseCreateEntities': true
plot.renderer.baseFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseFillPaint-java.awt.Paint-]	Paint	'plot.renderer.baseFillPaint': '#ffffff'
plot.renderer.baseItemLabelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/	Font	'plot.renderer.baseItemLabelFont': {'size': 10}

Name	Type	Example
renderer/AbstractRenderer.html#setBaseItemLabelFont-java.awt.Font-]		
plot.renderer.baseItemLabelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseItemLabelPaint-java.awt.Paint-]	Paint	'plot.renderer.baseItemLabelPaint': '#ffffff'
plot.renderer.baseItemLabelsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseItemLabelsVisible-boolean-]	boolean	'plot.renderer.baseItemLabelsVisible': true
plot.renderer.baseLegendShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseLegendShape-java.awt.Shape-]	Shape	'plot.renderer.baseLegendShape': {'shape': 'regular-cross', 'length': 5, 'thickness': 1}
plot.renderer.baseLegendTextFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseLegendTextFont-java.awt.Font-]	Font	'plot.renderer.baseLegendTextFont': {'size': 10}
plot.renderer.baseLegendTextPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseLegendTextPaint-java.awt.Paint-]	Paint	'plot.renderer.baseLegendTextPaint': '#ffffff'
plot.renderer.baseNegativeItemLabelPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseNegativeItemLabelPosition-org.jfree.chart.labels.ItemLabelPosition-]	ItemLabelPosition	'plot.renderer.baseNegativeItemLabelPosition': ['inside9', 'center', 'center', 0]
plot.renderer.baseOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseOutlinePaint-java.awt.Paint-]	Paint	'plot.renderer.baseOutlinePaint': '#ffffff'
plot.renderer.baseOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseOutlineStroke-java.awt.Stroke-]	Stroke	'plot.renderer.baseOutlineStroke': {'width': 1.0}

Name	Type	Example
plot.renderer.basePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBasePaint-java.awt.Paint-]	Paint	'plot.renderer.basePaint': '#ffffff'
plot.renderer.basePositiveItemLabelPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBasePositiveItemLabelPosition-org.jfree.chart.labels.ItemLabelPosition-]	ItemLabelPosition	'plot.renderer.basePositiveItemLabelPosition': ['inside1', 'center', 'center', 0]
plot.renderer.baseSeriesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseSeriesVisible-boolean-]	boolean	'plot.renderer.baseSeriesVisible': true
plot.renderer.baseSeriesVisibleInLegend [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseSeriesVisibleInLegend-boolean-]	boolean	'plot.renderer.baseSeriesVisibleInLegend': true
plot.renderer.baseShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseShape-java.awt.Shape-]	Shape	'plot.renderer.baseShape': {'shape': 'diagonal-cross', 'length': 5, 'thickness': 1}
plot.renderer.baseStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setBaseStroke-java.awt.Stroke-]	Stroke	'plot.renderer.baseStroke': {'width': 1.0}
plot.renderer.dataBoundsIncludesVisibleSeriesOnly [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setDataBoundsIncludesVisibleSeriesOnly-boolean-]	boolean	'plot.renderer.dataBoundsIncludesVisibleSeriesOnl true
plot.renderer.defaultEntityRadius [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setDefaultEntityRadius-int-]	int	'plot.renderer.defaultEntityRadius': 5
plot.renderer.itemLabelAnchorOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/	float	'plot.renderer.itemLabelAnchorOffset': 10

Name	Type	Example
AbstractRenderer.html#setItemLabelAnchorOffset-double-]		
plot.renderer.itemLabelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setItemLabelFont-java.awt.Font-]	Font	'plot.renderer.itemLabelFont': {'size': 10}
plot.renderer.itemLabelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setItemLabelPaint-java.awt.Paint-]	Paint	'plot.renderer.itemLabelPaint': '#ffffff'
plot.renderer.itemLabelsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setItemLabelsVisible-boolean-]	boolean	'plot.renderer.itemLabelsVisible': true
plot.renderer.legendShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setLegendShape-int-java.awt.Shape-]	Shape[]	'plot.renderer.legendShape': [{'shape': 'regular-cross', 'length': 5, 'thickness': 1},{'shape': 'diagonal-cross', 'length': 5, 'thickness': 1}]
plot.renderer.legendTextFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setLegendTextFont-int-java.awt.Font-]	Font[]	'plot.renderer.legendTextFont': [{'size': 10}]
plot.renderer.legendTextPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setLegendTextPaint-int-java.awt.Paint-]	Paint[]	'plot.renderer.legendTextPaint': ['#ffffff']
plot.renderer.seriesCreateEntities [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesCreateEntities-int-java.lang.Boolean-]	Boolean[]	'plot.renderer.seriesCreateEntities': [true,true,true]
plot.renderer.seriesFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesFillPaint-int-java.awt.Paint-]	Paint[]	'plot.renderer.seriesFillPaint': ['#ffffff']
plot.renderer.seriesItemLabelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/	Font[]	'plot.renderer.seriesItemLabelFont': [{'size': 10}]

Name	Type	Example
renderer/AbstractRenderer.html#setSeriesItemLabelFont-int-java.awt.Font-]		
plot.renderer.seriesItemLabelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesItemLabelPaint-int-java.awt.Paint-]	Paint[]	'plot.renderer.seriesItemLabelPaint': ['#ffffff']
plot.renderer.seriesItemLabelsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesItemLabelsVisible-int-boolean-]	boolean[]	'plot.renderer.seriesItemLabelsVisible': [true,true,true]
plot.renderer.seriesNegativeItemLabelPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesNegativeItemLabelPosition-int-org.jfree.chart.labels.ItemLabelPosition-]	ItemLabelPosition[]	'plot.renderer.seriesNegativeItemLabelPosition': [['inside9', 'center', 'center', 0]]
plot.renderer.seriesOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesOutlinePaint-int-java.awt.Paint-]	Paint[]	'plot.renderer.seriesOutlinePaint': ['#ffffff']
plot.renderer.seriesOutlineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesOutlineStroke-int-java.awt.Stroke-]	Stroke[]	'plot.renderer.seriesOutlineStroke': [{'width': 1.0}]
plot.renderer.seriesPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesPaint-int-java.awt.Paint-]	Paint[]	'plot.renderer.seriesPaint': ['#ffffff']
plot.renderer.seriesPositiveItemLabelPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/AbstractRenderer.html#setSeriesPositiveItemLabelPosition-int-org.jfree.chart.labels.ItemLabelPosition-]	ItemLabelPosition[]	'plot.renderer.seriesPositiveItemLabelPosition': [['inside9', 'center', 'center', 0]]
plot.renderer.seriesShape [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/	Shape[]	'plot.renderer.seriesShape': [{'shape': 'regular-cross', 'length': 5, 'thickness':

Name	Type	Example
AbstractRenderer.html#setSeries	Shape-	1},{'shape': 'diagonal-cross',
int-java.awt.Shape-]		'length': 5, 'thickness': 1}]
plot.renderer.seriesStroke	Stroke[]	'plot.renderer.seriesStroke':
[http://www.jfree.org/jfreechart/		[{'width': 1.0}]
api/javadoc/org/jfree/chart/		
renderer/		
AbstractRenderer.html#setSeries	Stroke-	
int-java.awt.Stroke-]		
plot.renderer.seriesVisible	Boolean[]	'plot.renderer.seriesVisible':
[http://www.jfree.org/jfreechart/		[true,true,true]
api/javadoc/org/jfree/chart/		
renderer/		
AbstractRenderer.html#setSeries	Visible-	
java.lang.Boolean-]		
plot.renderer.seriesVisibleInLegend	Boolean[]	'plot.renderer.seriesVisibleInLegend':
[http://www.jfree.org/jfreechart/		[true,true,true]
api/javadoc/org/jfree/chart/		
renderer/		
AbstractRenderer.html#setSeries	VisibleInLegend-	
java.lang.Boolean-]		

The following properties are specific to charts that use texts on the domain axis.

Table 4.21. AbstractCategoryItemRenderer properties

Name	Type	Example
plot.renderer.baseItemLabelGenerator	CategoryItemLabelGenerator	'plot.renderer.baseItemLabelGenerator':
[http://www.jfree.org/jfreechart/		{'labelFormat':'{2}'}
api/javadoc/org/jfree/chart/		
renderer/category/		
AbstractCategoryItemRenderer.html#setBaseItemLabelGenerator-		
org.jfree.chart.labels.CategoryItemLabelGenerator-		
]		
plot.renderer.legendItemLabelGenerator	CategorySeriesLabelGenerator	'plot.renderer.legendItemLabelGenerator':
[http://www.jfree.org/jfreechart/		'{0}'
api/javadoc/org/jfree/chart/		
renderer/category/		
AbstractCategoryItemRenderer.html#setLegendItemLabelGenerator-		
org.jfree.chart.labels.CategorySeriesLabelGenerator-		
]		
plot.renderer.seriesItemLabelGenerator	CategoryItemLabelGenerator[]	'plot.renderer.seriesItemLabelGenerator':
[http://www.jfree.org/jfreechart/		[{'labelFormat':'{2}'},
api/javadoc/org/jfree/chart/		{'labelFormat':'{2}'}]
renderer/category/		
AbstractCategoryItemRenderer.html#setSeriesItemLabelGenerator-		
int-		
org.jfree.chart.labels.CategoryItemLabelGenerator-		
]		

The following properties are specific to bar charts that use texts on the domain axis.

Table 4.22. BarRenderer properties

Name	Type	Example
plot.renderer.base [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setBase-double-]	double	'plot.renderer.base': 0
plot.renderer.drawBarOutline [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setDrawBarOutline-boolean-]	boolean	'plot.renderer.drawBarOutline': true
plot.renderer.includeBaseInRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setIncludeBaseInRange-boolean-]	boolean	'plot.renderer.includeBaseInRange': true
plot.renderer.itemMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setItemMargin-double-]	double	'plot.renderer.itemMargin': 0.05
plot.renderer.maximumBarWidth [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setMaximumBarWidth-double-]	double	'plot.renderer.maximumBarWidth': 0.35
plot.renderer.minimumBarLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setMinimumBarLength-double-]	double	'plot.renderer.minimumBarLength': 0
plot.renderer.negativeItemLabelPositionFallback [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setNegativeItemLabelPositionFallback-org.jfree.chart.labels.ItemLabelPosition-]	itemLabelPosition	'plot.renderer.negativeItemLabelPositionFallback': ['inside9', 'center', 'center', 0]
plot.renderer.positiveItemLabelPositionFallback [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setPositiveItemLabelPositionFallback-org.jfree.chart.labels.ItemLabelPosition-]	itemLabelPosition	'plot.renderer.positiveItemLabelPositionFallback': ['outside3', 'center', 'center', 0]

Name	Type	Example
plot.renderer.shadowPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setShadowPaint-java.awt.Paint-]	Paint	'plot.renderer.shadowPaint': ''
plot.renderer.shadowVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setShadowVisible-boolean-]	boolean	'plot.renderer.shadowVisible': false
plot.renderer.shadowXOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setShadowXOffset-double-]	double	'plot.renderer.shadowXOffset': 0
plot.renderer.shadowYOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/BarRenderer.html#setShadowYOffset-double-]	double	'plot.renderer.shadowYOffset': 0

The following properties are specific to stacked bar charts that use texts on the domain axis.

Table 4.23. StackedBarRenderer properties

Name	Type	Example
plot.renderer.renderAsPercentages [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/StackedBarRenderer.html#setRenderAsPercentages-boolean-]	boolean	'plot.renderer.renderAsPercentages': true

The following properties are specific to line charts that use texts on the domain axis.

Table 4.24. LineAndShapeRenderer properties

Name	Type	Example
plot.renderer.baseLinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setBaseLinesVisible-boolean-]	boolean	'plot.renderer.baseLinesVisible': true
plot.renderer.baseShapesFilled [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setBaseShapesFilled-boolean-]	boolean	'plot.renderer.baseShapesFilled': true

Name	Type	Example
plot.renderer.baseShapesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setBaseShapesVisible-boolean-]	boolean	'plot.renderer.baseShapesVisible': true
plot.renderer.drawOutlines [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setDrawOutlines-boolean-]	boolean	'plot.renderer.drawOutlines': true
plot.renderer.itemMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setItemMargin-double-]	double	'plot.renderer.itemMargin': 0.5
plot.renderer.seriesLinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setSeriesLinesVisible-int-boolean-]	boolean[]	'plot.renderer.seriesLinesVisible': [true,true,true]
plot.renderer.seriesShapesFilled [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setSeriesShapesFilled-int-boolean-]	boolean[]	'plot.renderer.seriesShapesFilled': [true,true,true]
plot.renderer.seriesShapesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setSeriesShapesVisible-int-boolean-]	boolean[]	'plot.renderer.seriesShapesVisible': [true,true,true]
plot.renderer.useFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setUseFillPaint-boolean-]	boolean	'plot.renderer.useFillPaint': true
plot.renderer.useOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/LineAndShapeRenderer.html#setUseOutlinePaint-boolean-]	boolean	'plot.renderer.useOutlinePaint': true
plot.renderer.useSeriesOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/	boolean	'plot.renderer.useSeriesOffset': true

Name	Type	Example
LineAndShapeRenderer.html#setUseSeriesOffset-boolean-]		

The following properties are specific to area charts that use texts on the domain axis.

Table 4.25. AreaRenderer properties

Name	Type	Example
plot.renderer.endType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/category/AreaRenderer.html#setEndType-org.jfree.chart.renderer.AreaRendererEndType-]	AreaRendererEndType	'plot.renderer.endType': 'level'

The following properties are specific to charts that use dates or numbers on the domain axis.

Table 4.26. AbstractXYItemRenderer properties

Name	Type	Example
plot.renderer.baseItemLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/AbstractXYItemRenderer.html#setBaseItemLabelGenerator-org.jfree.chart.labels.XYItemLabelGenerator-]	XYItemLabelGenerator	'plot.renderer.baseItemLabelGenerator': '{2}'
plot.renderer.legendItemLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/AbstractXYItemRenderer.html#setLegendItemLabelGenerator-org.jfree.chart.labels.XYSeriesLabelGenerator-]	XYSeriesLabelGenerator	'plot.renderer.legendItemLabelGenerator': '{0}'
plot.renderer.seriesItemLabelGenerator [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/AbstractXYItemRenderer.html#setSeriesItemLabelGenerator-int-org.jfree.chart.labels.XYItemLabelGenerator-]	XYItemLabelGenerator[]	'plot.renderer.seriesItemLabelGenerator': ['{2}', '{2}']

The following properties are specific to line charts that use dates or numbers on the domain axis.

Table 4.27. XYLineAndShapeRenderer properties

Name	Type	Example
plot.renderer.baseLinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setBaseLinesVisible-boolean-]	boolean	'plot.renderer.baseLinesVisible': true

Name	Type	Example
plot.renderer.baseShapesFilled [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setBaseShapesFilled-boolean-]	boolean	'plot.renderer.baseShapesFilled': true
plot.renderer.baseShapesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setBaseShapesVisible-boolean-]	boolean	'plot.renderer.baseShapesVisible': true
plot.renderer.drawOutlines [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setDrawOutlines-boolean-]	boolean	'plot.renderer.drawOutlines': true
plot.renderer.drawSeriesLineAsPath [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setDrawSeriesLineAsPath-boolean-]	boolean	'plot.renderer.drawSeriesLineAsPath': true
plot.renderer.seriesLinesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setSeriesLinesVisible-int-boolean-]	boolean[]	'plot.renderer.seriesLinesVisible': [true,true,true]
plot.renderer.seriesShapesFilled [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setSeriesShapesFilled-int-boolean-]	boolean[]	'plot.renderer.seriesShapesFilled': [true,true,true]
plot.renderer.seriesShapesVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setSeriesShapesVisible-int-boolean-]	boolean[]	'plot.renderer.seriesShapesVisible': [true,true,true]
plot.renderer.useFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.html#setUseFillPaint-boolean-]	boolean	'plot.renderer.useFillPaint': true
plot.renderer.useOutlinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/	boolean	'plot.renderer.useOutlinePaint': true

Name	Type	Example
XYLineAndShapeRenderer.html#setUseOutlinePaint-boolean-]		

Table 4.28. XYAreaRenderer properties

Name	Type	Example
plot.renderer.legendArea [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYAreaRenderer.html#setLegendArea-java.awt.Shape-]	Shape	'plot.renderer.legendArea': {'shape': 'rectangle', 'x': 0, 'y': 0, 'width': 5, 'height': 5}
plot.renderer.outline [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYAreaRenderer.html#setOutline-boolean-]	boolean	'plot.renderer.outline': true
plot.renderer.useFillPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/renderer/xy/XYAreaRenderer.html#setUseFillPaint-boolean-]	boolean	'plot.renderer.useFillPaint': true

4.4.3.6. Domain axis

There are two types of domain axes:

- axes that display discrete values (categories) and
- axes that display continuous values (dates or numbers).

The following properties are available for all types of axes.

Table 4.29. Axis properties

Name	Type	Example
plot.domainAxis.axisLinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLinePaint-java.awt.Paint-]	Paint	'plot.domainAxis.axisLinePaint': '#ffffff'
plot.domainAxis.axisLineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLineStroke-java.awt.Stroke-]	Stroke	'plot.domainAxis.axisLineStroke': {'width': 1.0}
plot.domainAxis.axisLineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLineVisible-boolean-]	boolean	'plot.domainAxis.axisLineVisible': true
plot.domainAxis.fixedDimension [http://www.jfree.org/jfreechart/	double	'plot.domainAxis.fixedDimension': 100

Name	Type	Example
api/javadoc/org/jfree/chart/axis/Axis.html#setFixedDimension-double-]		
plot.domainAxis.label [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabel-java.lang.String-]	String	'plot.domainAxis.label': 'Domain'
plot.domainAxis.labelAngle [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelAngle-double-]	double	Angle value is expressed in radians [https://www.google.ro/search?q=1+degrees+in+radians]. 'plot.domainAxis.labelAngle': 1.57
plot.domainAxis.labelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelFont-java.awt.Font-]	Font	'plot.domainAxis.labelFont': {'size': 8}
plot.domainAxis.labelInsets [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelInsets-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.domainAxis.labelInsets': [2, 2, 2, 2]
plot.domainAxis.labelLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelLocation-org.jfree.chart.axis.AxisLabelLocation-]	AxisLabelLocation	'plot.domainAxis.labelLocation': 'middle'
plot.domainAxis.labelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelPaint-java.awt.Paint-]	Paint	'plot.domainAxis.labelPaint': '#ffffff'
plot.domainAxis.minorTickMarkInsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarkInsideLength-float-]	Float	'plot.domainAxis.minorTickMarkInsideLength': 5
plot.domainAxis.minorTickMarkOutsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarkOutsideLength-float-]	Float	'plot.domainAxis.minorTickMarkOutsideLength': 5
plot.domainAxis.minorTickMarksVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarksVisible-boolean-]	Boolean	'plot.domainAxis.minorTickMarksVisible': true
plot.domainAxis.tickLabelFont [http://www.jfree.org/jfreechart/	Font	'plot.domainAxis.tickLabelFont': {'size': 6}

Name	Type	Example
api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelFont-java.awt.Font-]		
plot.domainAxis.tickLabelInsets [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelInsets-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.domainAxis.tickLabelInsets': [2, 2, 2, 2]
plot.domainAxis.tickLabelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelPaint-java.awt.Paint-]	Paint	'plot.domainAxis.tickLabelPaint': '#ffffff'
plot.domainAxis.tickLabelsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelsVisible-boolean-]	boolean	'plot.domainAxis.tickLabelsVisible': true
plot.domainAxis.tickMarkInsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkInsideLength-float-]	float	'plot.domainAxis.tickMarkInsideLength': 5
plot.domainAxis.tickMarkOutsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkOutsideLength-float-]	float	'plot.domainAxis.tickMarkOutsideLength': 5
plot.domainAxis.tickMarkPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkPaint-java.awt.Paint-]	Paint	'plot.domainAxis.tickMarkPaint': '#ff0000'
plot.domainAxis.tickMarkStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkStroke-java.awt.Stroke-]	Stroke	'plot.domainAxis.tickMarkStroke': {'width': 1.0}
plot.domainAxis.tickMarksVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarksVisible-boolean-]	boolean	'plot.domainAxis.tickMarksVisible': true
plot.domainAxis.visible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setVisible-boolean-]	boolean	'plot.domainAxis.visible': true

The following properties are available for axes that display categories.

Table 4.30. CategoryAxis properties

Name	Type	Example
plot.domainAxis.categoryLabelPositionOffset [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setCategoryLabelPositionOffset-int-]	int	'plot.domainAxis.categoryLabelPositionOffset': 5
plot.domainAxis.categoryLabelPositions [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setCategoryLabelPositions-org.jfree.chart.axis.CategoryLabelPositions-]	Not supported	N/A
plot.domainAxis.categoryMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setCategoryMargin-double-]	double	'plot.domainAxis.categoryMargin': 0.01
plot.domainAxis.lowerMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setLowerMargin-double-]	double	'plot.domainAxis.lowerMargin': 0.01
plot.domainAxis.maximumCategoryLabelLines [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setMaximumCategoryLabelLines-int-]	int	'plot.domainAxis.maximumCategoryLabelLines': 1
plot.domainAxis.maximumCategoryLabelWidthRatio [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setMaximumCategoryLabelWidthRatio-float-]	float	'plot.domainAxis.maximumCategoryLabelWidthRa 0.5
plot.domainAxis.upperMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/CategoryAxis.html#setUpperMargin-double-]	double	'plot.domainAxis.upperMargin': 0.01

The following properties are available for axes that display continuous values like numbers and dates.

Table 4.31. ValueAxis properties

Name	Type	Example
plot.domainAxis.autoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setAutoRange-boolean-]	boolean	'plot.domainAxis.autoRange': true
plot.domainAxis.autoRangeMinimumSize [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setAutoRangeMinimumSize-double-]	double	'plot.domainAxis.autoRangeMinimumSize': 1

Name	Type	Example
plot.domainAxis.autoTickUnitSelection [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setAutoTickUnitSelection-boolean-]	boolean	'plot.domainAxis.autoTickUnitSelection': true
plot.domainAxis.defaultAutoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setDefaultAutoRange-org.jfree.data.Range-]	Range	'plot.domainAxis.defaultAutoRange': [0, 10]
plot.domainAxis.downArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setDownArrow-java.awt.Shape-]	Shape	'plot.domainAxis.downArrow': {'shape': 'down-triangle', 'size': 3}
plot.domainAxis.fixedAutoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setFixedAutoRange-double-]	double	'plot.domainAxis.fixedAutoRange': 50
plot.domainAxis.inverted [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setInverted-boolean-]	boolean	'plot.domainAxis.inverted': true
plot.domainAxis.leftArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLeftArrow-java.awt.Shape-]	Shape	'plot.domainAxis.leftArrow': {'shape': 'left-triangle', 'size': 3}
plot.domainAxis.lowerBound [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLowerBound-double-]	double	'plot.domainAxis.lowerBound': 0
plot.domainAxis.lowerMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLowerMargin-double-]	double	'plot.domainAxis.lowerMargin': 0.01
plot.domainAxis.minorTickCount [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setMinorTickCount-int-]	int	'plot.domainAxis.minorTickCount': 10
plot.domainAxis.negativeArrowVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setNegativeArrowVisible-boolean-]	boolean	'plot.domainAxis.negativeArrowVisible': true
plot.domainAxis.positiveArrowVisible [http://www.jfree.org/jfreechart/	boolean	'plot.domainAxis.positiveArrowVisible': true

Name	Type	Example
api/javadoc/org/jfree/chart/axis/ValueAxis.html#setPositiveArrowVisible-boolean-]		
plot.domainAxis.range [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRange-org.jfree.data.Range-]	Range	'plot.domainAxis.range': [2, 20]
plot.domainAxis.rangeWithMargins [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRangeWithMargins-org.jfree.data.Range-]	Range	'plot.domainAxis.rangeWithMargins': [2, 20]
plot.domainAxis.rightArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRightArrow-java.awt.Shape-]	Shape	'plot.domainAxis.rightArrow': {'shape': 'right-triangle', 'size': 3}
plot.domainAxis.upArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpArrow-java.awt.Shape-]	Shape	'plot.domainAxis.upArrow': {'shape': 'up-triangle', 'size': 3}
plot.domainAxis.upperBound [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpperBound-double-]	double	'plot.domainAxis.upperBound': 20
plot.domainAxis.upperMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpperMargin-double-]	double	'plot.domainAxis.upperMargin': 0.01
plot.domainAxis.verticalTickLabels [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setVerticalTickLabels-boolean-]	boolean	'plot.domainAxis.verticalTickLabels': true

The following properties are available for axes that display numbers.

Table 4.32. NumberAxis properties

Name	Type	Example
plot.domainAxis.autoRangeIncludesZero [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setAutoRangeIncludesZero-boolean-]	boolean	'plot.domainAxis.autoRangeIncludesZero': true
plot.domainAxis.autoRangeStickyZero [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/	boolean	'plot.domainAxis.autoRangeStickyZero': true

Name	Type	Example
NumberAxis.html#setAutoRangeStickyZero-boolean-]		
plot.domainAxis.markerBand [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setMarkerBand-org.jfree.chart.axis.MarkerAxisBand-]	Not supported	N/A
plot.domainAxis.numberFormatOverride [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setNumberFormatOverride-java.text.NumberFormat-]	NumberFormat	'plot.domainAxis.numberFormatOverride': '0.000'
plot.domainAxis.rangeType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setRangeType-org.jfree.data.RangeType-]	RangeType	'plot.domainAxis.rangeType': 'full'
plot.domainAxis.tickUnit [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setTickUnit-org.jfree.chart.axis.NumberTickUnit-]	NumberTickUnit	'plot.domainAxis.tickUnit': {'size': 50}

The following properties are available for axes that display dates (including time).

Table 4.33. DateAxis properties

Name	Type	Example
plot.domainAxis.dateFormatOverride [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setDateFormatOverride-java.text.DateFormat-]	DateFormat	'plot.domainAxis.dateFormatOverride': 'MMM dd'
plot.domainAxis.locale [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setLocale-java.util.Locale-]	Locale	'plot.domainAxis.locale': 'fr_FR'
plot.domainAxis.maximumDate [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setMaximumDate-java.util.Date-]	Date	'plot.domainAxis.maximumDate': '2015-01-01'?date.iso
plot.domainAxis.minimumDate [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setMinimumDate-java.util.Date-]	Date	'plot.domainAxis.minimumDate': '2015-01-01T10:30:00'?datetime.iso
plot.domainAxis.range [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/	Range	'plot.domainAxis.range': ['10:30'?time.iso, '12:35'?time.iso]

Name	Type	Example
DateAxis.html#setRange-org.jfree.data.Range-]		
plot.domainAxis.tickMarkPosition [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setTickMarkPosition-org.jfree.chart.axis.DateTickMarkPosition-]	DateTickMarkPosition	'plot.domainAxis.tickMarkPosition': 'middle'
plot.domainAxis.tickUnit [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setTickUnit-org.jfree.chart.axis.DateTickUnit-]	DateTickUnit	'plot.domainAxis.tickUnit': {'unitType': 'hour', 'multiple': 2}
plot.domainAxis.timeline [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setTimeline-org.jfree.chart.axis.Timeline-]	Not supported (may be removed in version 2 [http://www.jfree.org/forum/viewtopic.php?f=10&t=116507])	N/A
plot.domainAxis.timezone [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/DateAxis.html#setTimeZone-java.util.TimeZone-]	TimeZone	'plot.domainAxis.timezone': 'GMT'

4.4.3.7. Range axis

Unlike the domain axis, the range axis displays only continuous values (dates or numbers). The following properties are available for all types of axes.

Table 4.34. Axis properties

Name	Type	Example
plot.rangeAxis.axisLinePaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLinePaint-java.awt.Paint-]	Paint	'plot.rangeAxis.axisLinePaint': '#ffffff'
plot.rangeAxis.axisLineStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLineStroke-java.awt.Stroke-]	Stroke	'plot.rangeAxis.axisLineStroke': {'width': 1.0}
plot.rangeAxis.axisLineVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setAxisLineVisible-boolean-]	boolean	'plot.rangeAxis.axisLineVisible': true
plot.rangeAxis.fixedDimension [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setFixedDimension-double-]	double	'plot.rangeAxis.fixedDimension': 100

Name	Type	Example
plot.rangeAxis.label [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabel-java.lang.String-]	String	'plot.rangeAxis.label': 'Range'
plot.rangeAxis.labelAngle [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelAngle-double-]	double	Angle value is expressed in radians [https://www.google.ro/search?q=1+degrees+in+radians]. 'plot.rangeAxis.labelAngle': 1.57
plot.rangeAxis.labelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelFont-java.awt.Font-]	Font	'plot.rangeAxis.labelFont': {'size': 8}
plot.rangeAxis.labelInsets [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelInsets-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.rangeAxis.labelInsets': [2, 2, 2, 2]
plot.rangeAxis.labelLocation [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelLocation-org.jfree.chart.axis.AxisLabelLocation-]	AxisLabelLocation	'plot.rangeAxis.labelLocation': 'middle'
plot.rangeAxis.labelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setLabelPaint-java.awt.Paint-]	Paint	'plot.rangeAxis.labelPaint': '#ffffff'
plot.rangeAxis.minorTickMarkInsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarkInsideLength-float-]	inside Length	'plot.rangeAxis.minorTickMarkInsideLength': 1
plot.rangeAxis.minorTickMarkOutsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarkOutsideLength-float-]	utside Length	'plot.rangeAxis.minorTickMarkOutsideLength': 1
plot.rangeAxis.minorTickMarksVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setMinorTickMarksVisible-boolean-]	isible an	'plot.rangeAxis.minorTickMarksVisible': true
plot.rangeAxis.tickLabelFont [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelFont-java.awt.Font-]	Font	'plot.rangeAxis.tickLabelFont': {'size': 6}

Name	Type	Example
plot.rangeAxis.tickLabelInsets [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelInsets-org.jfree.ui.RectangleInsets-]	RectangleInsets	'plot.rangeAxis.tickLabelInsets': [2, 2, 2, 2]
plot.rangeAxis.tickLabelPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelPaint-java.awt.Paint-]	Paint	'plot.rangeAxis.tickLabelPaint': '#ff0000'
plot.rangeAxis.tickLabelsVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickLabelsVisible-boolean-]	boolean	'plot.rangeAxis.tickLabelsVisible': true
plot.rangeAxis.tickMarkInsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkInsideLength-float-]	float	'plot.rangeAxis.tickMarkInsideLength': 5
plot.rangeAxis.tickMarkOutsideLength [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkOutsideLength-float-]	float	'plot.rangeAxis.tickMarkOutsideLength': 5
plot.rangeAxis.tickMarkPaint [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkPaint-java.awt.Paint-]	Paint	'plot.rangeAxis.tickMarkPaint': '#ffffff'
plot.rangeAxis.tickMarkStroke [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarkStroke-java.awt.Stroke-]	Stroke	'plot.rangeAxis.tickMarkStroke': {'width': 1.0}
plot.rangeAxis.tickMarksVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setTickMarksVisible-boolean-]	boolean	'plot.rangeAxis.tickMarksVisible': true
plot.rangeAxis.visible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/Axis.html#setVisible-boolean-]	boolean	'plot.rangeAxis.visible': true

The following properties are available for axes that display continuous values like dates or numbers.

Table 4.35. ValueAxis properties

Name	Type	Example
plot.rangeAxis.autoRangeMinimumSize [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/	double	'plot.rangeAxis.autoRangeMinimumSize': 1

Name	Type	Example
ValueAxis.html#setAutoRangeMinimumSize-double-]		
plot.rangeAxis.autoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setAutoRange-boolean-]	boolean	'plot.rangeAxis.autoRange': true
plot.rangeAxis.autoTickUnitSelection [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setAutoTickUnitSelection-boolean-]	boolean	'plot.rangeAxis.autoTickUnitSelection': true
plot.rangeAxis.defaultAutoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setDefaultAutoRange-org.jfree.data.Range-]	Range	'plot.rangeAxis.defaultAutoRange': [0, 10]
plot.rangeAxis.downArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setDownArrow-java.awt.Shape-]	Shape	'plot.rangeAxis.downArrow': {'shape': 'down-triangle', 'size': 3}
plot.rangeAxis.fixedAutoRange [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setFixedAutoRange-double-]	double	'plot.rangeAxis.fixedAutoRange': 50
plot.rangeAxis.inverted [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setInverted-boolean-]	boolean	'plot.rangeAxis.inverted': true
plot.rangeAxis.leftArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLeftArrow-java.awt.Shape-]	Shape	'plot.rangeAxis.leftArrow': {'shape': 'left-triangle', 'size': 3}
plot.rangeAxis.lowerBound [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLowerBound-double-]	double	'plot.rangeAxis.lowerBound': 0
plot.rangeAxis.lowerMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setLowerMargin-double-]	double	'plot.rangeAxis.lowerMargin': 0.01
plot.rangeAxis.minorTickCount [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setMinorTickCount-int-]	int	'plot.rangeAxis.minorTickCount': 10

Name	Type	Example
plot.rangeAxis.negativeArrowVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setNegativeArrowVisible-boolean-]	boolean	'plot.rangeAxis.negativeArrowVisible': true
plot.rangeAxis.positiveArrowVisible [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setPositiveArrowVisible-boolean-]	boolean	'plot.rangeAxis.positiveArrowVisible': true
plot.rangeAxis.range [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRange-org.jfree.data.Range-]	Range	'plot.rangeAxis.range': [2, 20]
plot.rangeAxis.rangeWithMargins [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRangeWithMargins-org.jfree.data.Range-]	Range	'plot.rangeAxis.rangeWithMargins': [2, 20]
plot.rangeAxis.rightArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setRightArrow-java.awt.Shape-]	Shape	'plot.rangeAxis.rightArrow': {'shape': 'right-triangle', 'size': 3}
plot.rangeAxis.upArrow [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpArrow-java.awt.Shape-]	Shape	'plot.rangeAxis.upArrow': {'shape': 'up-triangle', 'size': 3}
plot.rangeAxis.upperBound [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpperBound-double-]	double	'plot.rangeAxis.upperBound': 20
plot.rangeAxis.upperMargin [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setUpperMargin-double-]	double	'plot.rangeAxis.upperMargin': 0.01
plot.rangeAxis.verticalTickLabels [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/ValueAxis.html#setVerticalTickLabels-boolean-]	boolean	'plot.rangeAxis.verticalTickLabels': true

The following properties are available for axes that display numbers.

Table 4.36. NumberAxis properties

Name	Type	Example
plot.rangeAxis.autoRangeIncludesZero [http://www.jfree.org/jfreechart/	boolean	'plot.rangeAxis.autoRangeIncludesZero': true

Name	Type	Example
api/javadoc/org/jfree/chart/axis/NumberAxis.html#setAutoRangeIncludesZero-boolean-]		
plot.rangeAxis.autoRangeStickyZero [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setAutoRangeStickyZero-boolean-]	boolean	'plot.rangeAxis.autoRangeStickyZero': true
plot.rangeAxis.markerBand [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setMarkerBand-org.jfree.chart.axis.MarkerAxisBand-]	Not supported	N/A
plot.rangeAxis.numberFormatOverride [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setNumberFormatOverride-java.text.NumberFormat-]	NumberFormat	'plot.rangeAxis.numberFormatOverride': '0.000'
plot.rangeAxis.rangeType [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setRangeType-org.jfree.data.RangeType-]	RangeType	'plot.rangeAxis.rangeType': 'full'
plot.rangeAxis.tickUnit [http://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/axis/NumberAxis.html#setTickUnit-org.jfree.chart.axis.NumberTickUnit-]	NumberTickUnit	'plot.rangeAxis.tickUnit': {'size': 50}

4.5. Creating templates the easy way

Once you learn the basic notions of building a template, all you have to do is create a document with the right layout.

The previous section introduced the template language using plain text files. However plain text files are just plain. They are easier to understand and create but you can't have fancy layouts with headers, footers, page numbering and the like. One easy way to do this is to use a visual editor like Adobe Dreamweaver, Microsoft Word or LibreOffice Writer.

This section shows how to use placeholders, built-ins and directives with HTML, Microsoft Word or OpenDocument Text documents. Before you read any further, please make sure you read the first section as it tells how to use directives in tables.

4.5.1. Using directives in tables (important!)

If you read the section on directives, you probably noticed that *directives start and end on separate lines*. Since HTML, Microsoft Word or OpenDocument Text documents allow other elements than just paragraphs of text, this translates differently if you use tables.

Let's use a table to represent the services, products and expenses of an invoice:

Table 4.37. Invoice items in a table

Name	**Quantity**	**Price**	**Total**
Services			
Service1	5	$10	$50
Service2	2	$5	$10
Products			
Product1	1	$10	$10
Expenses			
Expense1	1	$30	$30
Total			**$100**

The template that produces this table looks as follows. Please observe that the start and end of a directive are emphasized and placed on separate lines. You can also color these lines in gray to distinguish them easier. They will be removed by Fanurio when it processes the template.

Table 4.38. Template code for invoice items in a table

Name	**Quantity**	**Price**	**Total**
Services			
[#list invoice.serviceItems as item]			
${item.name}	${item.quantity}	${item.price}	${item.total}
[/#list]			

Name	**Quantity**	**Price**	**Total**
[#if invoice.productItems?size != 0]			
Products			
[#list invoice.productItems as item]			
${item.name}	${item.quantity}	${item.price}	${item.total}
[/#list]			
[/#if]			
[#if invoice.expenseItems?size != 0]			
Expenses			
[#list invoice.expenseItems as item]			
${item.name}	${item.quantity}	${item.price}	${item.total}
[/#list]			
[/#if]			
Total			**${invoice.total}**

4.5.2. HyperText Markup Language (.html)

An HTML template [files/templates/html/html-invoice.html] is similar to the plain text template created in the previous section but with specific formatting like header, footer, page numbering and a table for the invoice items.

HTML is a very popular file format that's used especially for web pages. It's also the default file format for templates used in Fanurio. Fanurio can use templates in several file formats but only HTML templates can be used for both viewing and exporting invoices to PDF. A comparison of all file formats is available at the end of this guide.

One easy way to create an HTML template is to use the template editor. Another way (although not recommended) is create it by hand. If you prefer to create it manually, you may want to use a visual editor like Adobe Dreamweaver [http://www.adobe.com/products/dreamweaver/] (commercial) or KompoZer [http://www.kompozer.net] (free).

We also have a separate section that shows how to fine-tune an HTML document.

4.5.3. Microsoft Word (.docx)

A Microsoft Word 2007 template [files/templates/docx/word-invoice.docx] is similar to the plain text template created in the previous section but with specific formatting like header, footer, page numbering and a table for the invoice items.

Please note that Fanurio can only handle Microsoft Word 2007 (.docx) files. Older Microsoft Word formats are not supported. Templates created for Microsoft Word are regular files for which you have to disable a few settings.

Disable random numbers, grammar and spell check

Although this is something you don't see as a user, Word adds various details to a document that prevent Fanurio from using it as a template. To fix this problem, you need to disable a few settings [http://support.microsoft.com/kb/937422].

To Remove RSID (not available in the Mac version)

1. Click the Microsoft Office Button, and then click Word Options
2. Click Trust Center, and then click Trust Center Settings
3. Click on Privacy Options
4. Uncheck "Store random number to improve combine accuracy"

To Remove Grammar and Spell check

1. Click the Microsoft Office Button, and then click Word Options
2. Click Proofing
3. Click to clear the Check spelling as you type check box.
4. Click to clear the Mark grammar as you type check box.

When adding Freemarker code to a Microsoft Word template, you may want to cut the text and paste it back as unformatted text. This ensures that the text is inserted as a block and it's not split in multiple invisible fragments.

4.5.4. OpenDocument Text (.odt)

An OpenDocument Text template [files/templates/odt/odt-invoice.odt] is similar to the plain text template created in the previous section but with specific formatting like header, footer, page numbering and a table for the invoice items.

When adding Freemarker code to a OpenDocument Text template, you may want to cut the text and paste it back as unformatted text. This ensures that the text is inserted as a block and it's not split in multiple invisible fragments.

4.5.5. OpenDocument Spreadsheet (.ods)

An OpenDocument Spreadsheet template [files/templates/ods/ods-invoice.ods] is similar to the plain text template created in the previous section.

When adding Freemarker code to a OpenDocument Spreadsheet template, you may want to cut the text and paste it back as unformatted text. This ensures that the text is inserted as a block and it's not split in multiple invisible fragments.

4.6. Creating templates like a professional (HTML and CSS)

This section only shows what you can do to improve the layout of an HTML document in general. You should read this if you want to change an HTML template manually.

If you are creating an HTML template, we highly recommend that you use the template editor.

4.6.1. Page formatting

If you are using a template to export to PDF, there are a few settings that you could add to the template to make the PDF document look nicer. These settings are actually CSS 3 page rules [http://www.w3.org/TR/css3-page/]. This section covers the following settings:

- page size
- header and footer
- page numbers
- page breaks
- background layer
- document properties

Please note that not all CSS3 page properties are currently supported by Fanurio when converting a document from HTML to PDF.

Page size

The size of a page can be set using the @page rule [http://www.w3.org/TR/css3-page/#page-box-page-rule].

```
@page {
 size: A4 portrait; /* can also use 'landscape' for orientation */
}
```

Header and footer

The @page rule can also be used to specify the header and footer. The following example shows a document that has a 3-inch header and an 1-inch footer.

```
<html>
<head>
 <style type="text/css" media="all">
  @page {
   size: A4 portrait; /* can use also 'landscape' for orientation */
   margin-top: 3.0in;
   margin-bottom: 1.0in;

   @bottom-center {
    content: element(footer);
   }

   @top-center {
    content: element(header);
   }
  }
```

```
  #page-header {
   display: block;
   position: running(header);
  }

  #page-footer {
   display: block;
   position: running(footer);
  }
 </style>
</head>

<body>
 <div id="page-header">
  <p>Header text</p>
 </div>

 <div id="page-footer">
  <p>Footer text</p>
 </div>

 <div id="page-content">
  <p>Body text</p>
 </div>
</body>
</html>
```

Page numbers

Use span tags to insert the current page number and the number of total pages.

```
<p>Page <span class="page-number"/> of <span class="page-count"/></p>
```

The span tags must be styled to contain the actual values.

```
.page-number:before {
 content: counter(page);
}

.page-count:before {
 content: counter(pages);
}
```

Page breaks

There are several page-break properties [http://www.w3.org/TR/css3-page/#page-breaks]. To enter a page break, just use the *page-break-after* property as shown in the following example.

```
<p style="page-break-after:always;"/>
```

Background layer

Fanurio can add a background layer when exporting an HTML document to PDF. To instruct it to do this, you have to add a link tag in the head section as shown in the following example.

```
<link rel="stylesheet" type="application/pdf" href="letterhead.pdf" />
```

The background layer is useful if you have a letterhead and you want to print the contents of the document over it.

Document properties

Title and meta tags from the head section are converted to document properties when an HTML document is exported to PDF. You can add author, subject and keywords meta tags to include additional information in exported PDFs.

```
<title>PDF Test</title>
<meta name="author" content="author"/>
<meta name="subject" content="subject"/>
<meta name="keywords" content="keyword 1, keyword 2"/>
```

Sample document

The following document has two A4 pages with portrait orientation and a 0.25 inch margin. A black border is drawn around the pages to emphasize the content.

```
<html>
<head>
 <style type="text/css" media="all">
  @page {
   size: A4 portrait; /* can use also 'landscape' for orientation */
   margin: 1.0in;
   border: thin solid black;
   padding: 1em;

   @bottom-center {
    content: element(footer);
   }

   @top-center {
    content: element(header);
   }
  }

  #page-header {
   display: block;
   position: running(header);
  }

  #page-footer {
   display: block;
   position: running(footer);
  }

  .page-number:before {
   content: counter(page);
  }

  .page-count:before {
   content: counter(pages);
  }
 </style>
</head>

<body>
 <div id="page-header">
  <p>Header text</p>
 </div>
```

```
 <div id="page-footer">
  <p>Footer text</p>
  <p>Page <span class="page-number"/> of <span class="page-count"/></p>
 </div>

 <div id="page-content">
  <p>Page 1.</p>

  <p style="page-break-after:always;"/>

  <p>Page 2.</p>
 </div>
</body>
</html>
```

4.6.2. Fonts

If you want to use specific fonts for your templates, make sure you copy the font files to the same folder where the template is located and then change the template to use them.

Fanurio can only work with TrueType (*.ttf) and OpenType (*.otf) fonts. If you have other types of fonts, you need to convert them to one of these types. You should copy the font files only if you use the template to export the document to PDF. If you export your documents to HTML, it's the job of the browser to render them correctly and you don't have to copy them anymore.

The following CSS code configures the template to use a specific font:

```
body {
    font-family: "AG Buch Condensed BQ"; /* replace this with your font */
}
```

If the font doesn't render correctly, you also have to specify the character encoding. By default, Fanurio assumes it is UTF-8 (Unicode) but this will not work every time.

The following code configures the template to use a certain character encoding:

```
<head>
 ...
 <!-- Replace Cp1252 with your charset -->
 <meta http-equiv="Content-Type" content="application/pdf; charset=Cp1252"/>
 ...
</head>
```

Cp1252 refers to the West European Latin character encoding. Other encodings are:

- Cp1250 - East European Latin
- Cp1251 - Cyrillic
- Cp1252 - West European Latin
- Cp1253 - Greek
- Cp1254 - Turkish
- Cp1255 - Hebrew
- Cp1256 - Arabic
- Cp1257 - Baltic

- Cp1258 - Vietnamese

4.7. Creating templates like a professional (Plain Text and XML)

The one thing that all template file formats supported by Fanurio have in common is that they all are text-based formats. Technically, Fanurio can handle any text-based format through its Freemarker [http://freemarker.sourceforge.net/] "template engine".

Fanurio handles two types of text formats:

- **standard:** HTML, Microsoft Word 2007, OpenDocument Text and OpenDocument Spreadsheet.
- **generic:** plain text and XML.

All standard formats are explained in previous sections. This section is about the generic formats.

When scanning for templates, Fanurio will also include .txt and .xml files on the list. You can create a text-based template and save it as .txt or .xml and place it in the templates folder. Templates saved as .xml files are also checked to make sure they are valid XML files.

A generic text template can be used to implement any text-based format. For instance, you could use it to generate RTF (Rich Text Format) documents if you know the RTF syntax. The following section shows how to use a plain text file to create a Scribus template.

4.7.1. Example: Quicken

Quicken [http://quicken.intuit.com/] is a personal finance management tool developed by Intuit, Inc. Quicken can read and write data to QIF [http://en.wikipedia.org/wiki/Quicken_Interchange_Format] (Quicken Interchange Format) files. QIF files have the .qif extension and are plain text files.

This section shows what you need to do to export Fanurio invoices as QIF files. Before creating a QIF template, make sure you read the section that explains the template language.

```
!Type:Invoice
D${invoice.date?string("M/dd''yy")}
U${invoice.grandTotal.amount}
T${invoice.grandTotal.amount}
N${invoice.number}
P${client.name}
M${invoice.notes}
LBusiness Income/AA
A${client.attention}
A${client.address}
"A${client.city}, ${client.state} ${client.zip}"
A
A
[#foreach item in invoice.items]
SBusiness Income
"${item.description}"
${item.total}
[/#foreach]
XI1
XE${invoice.dueDate?string("M/dd''yy")}
XR0.0
XT0.00
[#foreach item in invoice.items]
XN${item.name}
X#${item.quantity}
```

```
X${item.price}
"XS${item.description}"
[/#foreach]
^
```

The best way to create a template in the QIF format is to create an invoice in Quicken and then export it to QIF. Once you do that, start replacing its fields with placeholders recognized by Fanurio.

The QIF template file is in the file quicken-invoice.qif [files/templates/qif/quicken-invoice.qif].

4.7.2. Example: Scribus (DTP)

Scribus [http://www.scribus.net/] is an open source, multi-platform desktop publishing application that competes with Adobe InDesign and Quark XPress. It uses an XML-based file format for its files. Scribus files have the .sla extension.

This section shows what you need to do to export Fanurio invoices as Scribus files. Before creating a Scribus template, make sure you read the section that explains the template language.

Step 1: Create a sample invoice in Scribus

Start by creating a sample invoice that includes your business details, the contact information of your client, the invoice details and a table for the items that you need to invoice. Instead of entering actual data, you can use placeholders to specify things like your address or the invoice number.

For a list of all placeholders that you can use in a template, please see the reference section. Instead of creating a Scribus file from scratch, you can use this one [files/templates/txt/scribus-invoice.sla].

Step 2: Transform the Scribus file into a template

There are a few things you need to do to transform a Scribus file into a template.

1. **File extension:** Although the Scribus file uses an XML-based format, it needs to be saved as a plain text file and not as an XML file. A Scribus file is not XML-valid. The first thing you need to do is change the file extension from .sla to .txt.

2. **Repeated rows:** The row that displays the details of an invoice item has to be repeated for each item. That's why all PAGEOBJECT elements must be wrapped by a list directive. Open the text template, find these elements and then surround them as shown below.

   ```
   [#list invoice.items as item]
       <PAGEBOBJECT ... />
       ...
       <PAGEBOBJECT ... />
   [/#list]
   ```

3. **Y coordinates:** Page objects have absolute coordinates. When repeating elements that are used to represent rows, their YPOS attribute must be different for each row. PAGEBOBJECT elements now look like this:

   ```
   [#list invoice.items as item]
       <PAGEBOBJECT ... YPOS="${270 + 30 * (item_index + 1)}" ... />
       ...
       <PAGEBOBJECT ... />
   [/#list]
   ```

   where 270 is the Y coordinate for the header row and 30 is the height of the item row.

   Rows below the items rows also have to have a calculated Y coordinate because their position may vary with the number of items of an invoice.

```
[#list invoice.items as item]
    <PAGEBOBJECT ... YPOS="${270 + 30 * (item_index + 1)}" ... />
    ...
    <PAGEBOBJECT ... />
[/#list]
```

`<PAGEBOBJECT ... YPOS="${270 + 30 * invoice.items?size + 1}" ... />`

Once you have the template ready, install it and then export an invoice. Make sure you change the extension manually to .sla because it is set by default to .txt.

The text template file based on the Scribus file is in the file scribus-invoice.txt [files/templates/txt/scribus-invoice.txt].

4.8. A comparison of all supported file formats

The following table shows all file formats that Fanurio can use as templates.

Table 4.39. Template file formats supported by Fanurio

Template Format	View	Print	Export	Email	Editor	Example
HTML (.html)	**+**	**+**	**.html, .pdf**	**.pdf**	**Text editor, HTML editor**	**html-invoice.html [files/templates/html/html-invoice.html]**
Microsoft Word 2007 (.docx)	-	-	.docx	.docx	Microsoft Word	word-invoice.docx [files/templates/docx/word-invoice.docx]
OpenDocument Text (.odt)	-	-	.odt	.odt	OpenOffice Writer, LibreOffice Writer and others [http://en.wikipedia.org/wiki/OpenDocument_software]	odt-invoice.odt [files/templates/odt/odt-invoice.odt]
OpenDocument Spreadsheet (.ods)	-	-	.ods	.ods	OpenOffice Calc, LibreOffice Calc and others [http://en.wikipedia.org/wiki/OpenDocument_software]	ods-invoice.ods [files/templates/ods/ods-invoice.ods]
XML (.xml)	+	+	.xml	.xml	Text editor, XML editor	-
Comma-Separated Values (.csv)	+	+	.csv	.csv	Text editors	Timelog.csv [files/templates/csv/Timelog.csv]
Plain Text (.txt)	+	+	.txt	.txt	Text editors	text-invoice.txt [files/templates/txt/text-invoice.txt]

Template Format	View	Print	Export	Email	Editor	Example
Quicken (.qif)	+	+	.qif	.qif	Text editors	quicken-invoice.qif [files/templates/qif/quicken-invoice.qif]
QuickBooks (.iif)	+	+	.iif	.iif	Text editors	-
TeX Document (.tex)	-	-	.tex	.tex	Text editors, TeX/LaTeX editor	-

If you don't know which format is right for you, here's a short summary of what each format can do:

- **HTML:** This is the default and recommended format for templates in Fanurio. You should use HTML templates if you want to view, print or export documents to PDF right from Fanurio. The other formats require an additional step to get the same results. You have to open them with their editor for viewing or printing.

  We recommend that you use the template editor to edit HTML templates but you can also edit them visually using HTML editors like Adobe Dreamweaver [http://www.adobe.com/products/dreamweaver/] (commercial) or KompoZer [http://www.kompozer.net/] (free).

- **Microsoft Word 2007:** Microsoft Word [http://office.microsoft.com] templates can be easily edited to create complex layouts. Use this format if you want to create great looking documents in a short time.

- **OpenDocument:** OpenDocument formats (.odt and .ods) help you get the same results as Microsoft Office but on multiple platforms. An OpenDocument Text template is a great choice if you are using Fanurio on Linux and you want to create an invoice template fast.

  OpenOffice [http://www.openoffice.org/] and LibreOffice [http://www.libreoffice.org/] are two popular free applications that can handle OpenDocument formats. A list of applications that support the OpenDocument format [http://en.wikipedia.org/wiki/OpenDocument_software] is available on Wikipedia.

- **XML:** Use this format if you want to export data to XML and then import it in other applications.

- **CSV:** Use this format if you want to export data as a table.

- **Plain Text:** The text format is not very practical unless you need to export your documents as plain text files. We are using plain text templates to explain the template language syntax.

  One could also use this format to export documents to any text-based format. For instance, you could use it to generate RTF (Rich Text Format) documents if you know the RTF syntax.

- **Quicken:** The QIF format is in fact a plain text format. Use this format if you need to keep your invoices in Quicken.

4.9. Placeholders reference

This is a reference section that shows all the fields that can be used in an invoice template. These fields can be used to include information about:

- your business
- your client's business
- the actual invoice content

Some fields contain an additional description to properly describe their meaning.

4.9.1. Business

The following table shows the fields that can be used to include information about your business in an invoice.

Table 4.40. Business Properties

Property	Version	Meaning
1. General		
business.name	1.0	self-explained
business.code	2.3	A short code that identifies your business.
business.businessNumber	2.3	The number assigned to your business when it was created.
business.otherNumber	2.3	A number that could be used as the trade register number.
business.taxNumber	1.4	The tax number assigned by the government to your business. In many countries, businesses must include their VAT number on their invoices. In such cases, the tax number is the VAT number.
2. Taxes		

business.taxLiable	1.5	Indicates whether the business pays taxes or not. A business pays taxes if they are enabled.
		
		If you want to do something when the business **pays** taxes:
		
		`[#if business.taxLiable]`
		`...`
		`[/#if]`
		
		If you want to do something when the business **doesn't pay** taxes:
		
		`[#if !business.taxLiable]`
		`...`
		`[/#if]`

3. Contact		

Property	Version	Meaning
business.attention	1.0	self-explained
business.address	1.0	self-explained
business.city	1.0	self-explained
business.state	1.0	self-explained
business.zip	1.0	self-explained
business.country	1.0	self-explained
business.phone	1.0	self-explained
business.fax	1.0	self-explained
business.mobile	2.1	self-explained
business.other	2.1	self-explained
business.email	1.0	self-explained
business.website	1.8	self-explained

4.9.2. Client

The following table shows the fields that can be used to include information about your client's business in an invoice.

Table 4.41. Client Properties

Property	Version	Meaning
1. General		
client.name	1.0	self-explained
client.code	1.5	The client code is especially useful if you want to number invoices automatically and to include the client code in the invoice number.
client.businessNumber	2.3	The number assigned to the business when it was created.
client.otherNumber	2.3	A number that could be used as the trade register number.
client.taxNumber	1.4	The tax number assigned by the government to your client's business. In many countries, invoices must include the client's VAT number. In such cases, the tax number is the VAT number.

client.invoices	2.0	A list with all the invoices of a client. This is useful if you want to create a statement. To access an invoice from the list use the following Freemarker code:
		
		`[#list client.invoices as invoice]`
		`...`
		`[/#list]`

client.balance	2.0	How much money the client has to pay for all open

Property	Version	Meaning
		(unpaid) invoices. Balance is the difference between the total of all invoices and the total of all payments.

client.deposits	3.1	A list with all the deposits of a client. To access a deposit from the list use the following Freemarker code:
		
		`[#list client.deposits as deposit]`
		`...`
		`[/#list]`

client.depositsBalance	3.1	How much money the client has in its deposits account. The balance increases with each new deposit and decreases with each withdrawal.
2. Contact		
client.attention	1.0	self-explained
client.address	1.0	self-explained
client.city	1.0	self-explained
client.state	1.0	self-explained
client.zip	1.0	self-explained
client.country	1.0	self-explained
client.phone	1.0	self-explained
client.fax	1.0	self-explained
client.mobile	2.1	self-explained
client.other	2.1	self-explained
client.email	1.0	self-explained
client.website	1.8	self-explained

4.9.3. Invoice

The following table shows the fields that can be used to include information about the actual invoice.

Table 4.42. Invoice Properties

Property	Version	Meaning
1. General		
invoice.number	1.0	self-explained
invoice.reference	1.9, 2.1, 3.1	A reference number like the purchase order number. **Note:** This field was known as **contract.reference** before version 3.1 and as **contract.clientNumber** before version 2.1
invoice.notes	2.0	self-explained

Property	Version	Meaning
invoice.date	1.0	Use ${invoice.date?date} to access the value of the invoice date.
invoice.period	2.3	Indicates the date or period when the invoiced services were provided.
invoice.dueDate	1.5	Use ${invoice.dueDate?date} to access the date when an invoice is due.
invoice.dueDays	1.5	When the invoice must be paid (in days since the invoice date).
invoice.overdue	2.1.2	Indicates whether an invoice is overdue or not.
2. Payments		
invoice.ageInDays	2.0	The number of days since an invoice was created until it was paid.
invoice.balance	2.0	How much money the client has to pay. Balance is the difference between the invoice total and payments total.
invoice.paymentsTotal	2.0	self-explained

invoice.payments	2.0	A list with all the payments of an invoice. To access a payment from the list use the following Freemarker code:
		
		`[#list invoice.payments as payment`
		`...`
		`[/#list]`

invoice.paid	2.0	Indicates whether an invoice is paid (completely) or not.
invoice.paymentDate	2.0	Use ${invoice.paymentDate?date} to access the value of the date when the invoice was paid completely.
3. Contents The contents of an invoice may be accessed either by project or directly by referencing its items.		

invoice.items	1.4, 2.0	A list with all the items of an invoice. To access an item from the list use the following Freemarker code:
		
		`[#list invoice.items as item]`
		`...`
		`[/#list]`
		
		Since version 2.0, this placeholder has a new meaning. The list contains all items, including products

Property	Version	Meaning
		and expenses. Previously, it contained only services. Instead of accessing the entire list of items, one can access the service items, the mileage items, the expense items and the product items separately using the **invoice.serviceItems**, **invoice.expenseItems**, **invoice.mileageItems** and **invoice.productItems** placeholders.

invoice.regularItems	3.0, 3.1	A list with all the regular (non-project) items of an invoice. If you don't invoice projects then ${invoice.regularItems} is the same as ${invoice.items}. To access an item from this list use the following Freemarker code:
		
		`[#list invoice.regularItems as ite`
		`...`
		`[/#list]`
		
		Note: **invoice.regularItems** replaces **invoice.nonProjectItems** which was used in version 3.0.

invoice.serviceItems	2.0, 2.7	A list with all the service items of an invoice. To access a service item from the list use the following Freemarker code:
		
		`[#list invoice.serviceItems as ite`
		`...`
		`[/#list]`
		
		Note: **invoice.serviceItems** replaces **invoice.services** which was used up until version 2.6.

invoice.expenseItems	1.4, 2.7	A list with all the expense items of an invoice. To access an expense item from the list use the following Freemarker code:
		
		`[#list invoice.expenseItems as ite`
		`...`
		`[/#list]`
		
		Note: **invoice.expenseItems** replaces **invoice.expenses** which was used up until version 2.6.

invoice.mileageItems	3.0	A list with all the mileage items of an invoice. To access a

Property	Version	Meaning

		mileage item from the list use the following Freemarker code:
		
		`[#list invoice.mileageItems as ite`
		`...`
		`[/#list]`

invoice.productItems	2.5, 2.7	A list with all the product items of an invoice. To access a product item from the list use the following Freemarker code:
		
		`[#list invoice.productItems as ite`
		`...`
		`[/#list]`
		
		Note: **invoice.productItems** replaces **invoice.products** which was used up until version 2.6.

invoice.projects	1.0	A list with all the projects whose items belong to this invoice. The list of projects is useful if you invoice multiple projects.
		
		To access a project from the list use the following Freemarker code:
		
		`[#list invoice.projects as project`
		`...`
		`[/#list]`

4. Totals		
invoice.serviceItemsSubtotal	2.0, 2.7	The total amount of money due for all the service items from the invoice. Note: **invoice.serviceItemsSubtotal** replaces **invoice.servicesSubtotal** which was used up until version 2.6.
invoice.expenseItemsSubtotal	1.0, 2.7	The total amount of money due for all the expense items from the invoice. Note: **invoice.expenseItemsSubtotal** replaces **invoice.expensesSubtotal** which was used up until version 2.6.
invoice.mileageItemsSubtotal	3.0	The total amount of money due for all the mileage items from the invoice.

Property	Version	Meaning
invoice.productItemsSubtotal	2.5, 2.7	The total amount of money due for all the product items from the invoice. Note: **invoice.productItemsSubtotal** replaces **invoice.productsSubtotal** which was used up until version 2.6.
invoice.total	1.0	The total amount of money (no taxes included).
invoice.grandTotal	1.4	The total amount of money (all taxes included).
invoice.profit	3.1	The invoice profit is calculated as the difference between the invoice total and the total amount of all invoice expenses.

invoice.exchangeRates	2.0	A list with all the exchange rates of an invoice if it uses multiple currencies. To access exchange rates from the list use the following Freemarker code:
		
		`[#list invoice.exchangeRates as ex`
		`...`
		`${exchangeRate.source}`
		`${exchangeRate.target}`
		`${exchangeRate.rate}`
		`...`
		`[/#list]`

5. Taxes		

invoice.taxable	1.8	Indicates whether the invoice has taxes or not. If you want to do something when the invoice **has** taxes:
		
		`[#if invoice.taxable]`
		`...`
		`[/#if]`
		
		If you want to do something when the invoice **doesn't have** taxes:
		
		`[#if !invoice.taxable]`
		`...`
		`[/#if]`

invoice.taxes	1.4	A list with all the taxes of an invoice. To access a tax from the list use the following Freemarker code: `[#list invoice.taxes as tax]`

Property	Version	Meaning
		`...` `[/#list]`
invoice.taxTotal(tax)	1.4	This is a function that gives you the total amount for a certain tax. The most common situation when this function is used is at the end of the invoice to display the total for each tax of the invoice. If the invoice has more than one tax.
invoice.taxesTotal	1.4	The taxes due for the invoice.
6. Discounts		

invoice.regularTotal	1.8	If the total of an invoice is discounted, this field indicates the **total before the discount**. You may want to use this field like this:
		
		`[#if item.totalDiscounted]`
		`  ${item.regularTotal}`
		`[/#if]`
		
		The regular total is accessed only when the total is discounted.

invoice.rawTotal	1.8	If an invoice has discounts for both its items and its total value, this field represents the value of the invoice without any discount. It's how much you would make if no discount is applied.
invoice.totalDiscount	1.8	If the total value of an invoice is discounted, this field indicates the discount. It can be used to display the actual discount.
invoice.totalDiscounted	1.8	Indicates if the total of an invoice is discounted. See the **item.regularPrice** field above.
7. Time		
invoice.billingTimeAsHour	1.5, 3.1, 4.0	The total invoiced time in hour format. For instance 1 hour and 30 minutes is represented as 1:30. Note: **invoice.billingTimeAsHour** replaces **invoice.billableTimeAsHour** which was used up until version 3.0 and **invoice.billedTimeAsHour** which was used up until version 4.0.

Property	Version	Meaning
invoice.billingTimeAsDecimal	1.5, 3.1, 4.0	The total invoiced time in decimal format. For instance 1 hour and 30 minutes is represented as 1.50. Note: **invoice.billingTimeAsDecimal** replaces **invoice.billableTimeAsDecimal** which was used up until version 3.0 and **invoice.billedTimeAsDecimal** which was used up until version 4.0.
8. Time Reporting The following placeholders are useful if you need to create invoice templates that group time entries by date. These placeholders are used by the invoice template editor (File » Template Editor) if time is displayed and time entris are grouped by date. It uses ${invoice.groupDatesByDate(item.timeEntries)} to determine the dates when time was recorded for a service item.		
invoice.timeEntries	3.1	A list with all the time entries that belong to the service items of an invoice. This list is useful if you want to determine the dates when the invoiced time was recorded (see invoice.groupDatesByDate). If you need to access the date of the first and last time entry, you can use the following expressions: • ${invoice.timeEntries?sort_by("date")?first.date?date} for the first date • ${invoice.timeEntries?sort_by("date")?last.date?date} for the last date

invoice.groupDatesByDate(timeEntries)	3.1	A list with the dates when the specified time entries were recorded.
		
		`[#list invoice.groupDatesByDate(in`
		`...`
		`[/#list]`
		
		Use **${date.toDate()?date}** to print each date and **${date.toInterval()}** to create an interval for a date.

Property	Version	Meaning

invoice.groupPeople(timeEntries)	4.0	A list with all the people.
		
		`[#list invoice.groupPeople(invoice`
		`...`
		`[/#list]`

invoice.filterByDate(timeEntries, interval)	3.1	Finds the subset of the specified time entries that were recorded in the specified interval. (their Date field belongs to the interval)
invoice.filterByPerson(timeEntries, person)	4.0	Finds the subset of the specified time entries that were created by the specified person.
invoice.calculateBillingTimeAsHour(timeEntries)	3.1, 4.0	The total billing time in hour format for the specified list of time entries. Note: **invoice.calculateBillingTimeAsHour** replaces **invoice.calculateRoundedElapsedTimeAsHour** which was used up until version 4.0.
invoice.calculateBillingTimeAsDecimal(timeEntries)	3.1, 4.0	The total billing time in decimal format for the specified list of time entries. Note: **invoice.calculateBillingTimeAsDecimal** replaces **invoice.calculateRoundedElapsedTimeAsDecim** which was used up until version 4.0.
invoice.calculateElapsedTimeAsHour(timeEntries)	3.1	The total elapsed time in hour format for the specified list of time entries.
invoice.calculateElapsedTimeAsDecimal(timeEntries)	3.1	The total elapsed time in decimal format for the specified list of time entries.

4.9.4. Item

The following table shows the fields of an item from an invoice. An item cannot be accessed directly. You can either access it from the list of items of an invoice (**invoice.items** or **invoice.serviceItems**, **invoice.productItems** and **invoice.expenseItems**) or from the list of items of an invoice project (**project.items** or **project.serviceItems**, **project.productItems** and **project.expenseItems**).

Table 4.43. Item Properties

Property	Version	Meaning
1. General The general fields are used to describe an item.		
item.name	1.0	self-explained

Property	Version	Meaning
item.description	1.0	self-explained

item.date	2.0, 3.0	This placeholder is deprecated starting with version 3.0. Items don't actually have a date field.
		• for service items
		  • with at least one time entry, the date represents the date of the oldest time entry
		  • with no time entries, the date represents the start date of the task
		• for expense items the date represents the date when the expense was made
		• for mileage items the date represents the invoice date
		• for product items the date represents the date when the product was made
		Use ${item.date?date} to access the value of the item date.

item.catalogItem	1.0, 2.0, 2.5	self-explained `${item.catalogItem!"-"}` Note: **item.catalogItem** replaces **item.itemCategory** (introduced in version 2.0) and **item.service** (used before version 2.0).

item.regularItem	3.1	Indicates whether this item is a regular item or not. All items that are added directly to an invoice (i.e. they don't invoice a project) are regular items. The following code prints a plus character if the item is a regular item.
		`[#if item.regularItem]`
		`  +`
		`[/#if]`
		To access the regular items of an invoice, see the **invoice.regularItems** placeholder.

item.serviceItem	2.0, 2.7	Indicates whether this item is a service item or not. The following code prints a plus

Property	Version	Meaning

		character if the item is a service item.
		
		`[#if item.serviceItem]`
		`+`
		`[/#if]`
		
		To access the list of service item of an invoice project see the **project.serviceItems** placeholder. For an invoice, see **invoice.serviceItems**.
		
		Note: **item.serviceItem** replaces **item.service** (used before version 2.7).

item.expenseItem	2.0, 2.7	Indicates whether this item is an expense item or not. The following code prints a plus character if the item is an expense item.
		
		`[#if item.expenseItem]`
		`+`
		`[/#if]`
		
		To access the list of expense items of a project see the **project.expenseItems** placeholder. For an invoice, see **invoice.expenseItems**.
		
		Note: **item.expenseItem** replaces **item.expense** (used before version 2.7).

item.mileageItem	3.0	Indicates whether this item is a mileage item or not. The following code prints a plus character if the item is a mileage item.
		
		`[#if item.mileageItem]`
		`+`
		`[/#if]`
		
		To access the list of mileage items of a project see the **project.mileageItems** placeholder. For an invoice, see **invoice.mileageItems**.

item.productItem	2.5, 2.7	Indicates whether this item is a product item or not. The following code prints a plus character if the item is a product item. `[#if item.productItem]`

Property	Version	Meaning
		+ [/#if] To access the list of product items of a project see the **project.productItems** placeholder. For an invoice, see **invoice.productItems**. Note: **item.productItem** replaces **item.product** (used before version 2.7).
2. Billing The billing fields are used to show billing information (price, quantity, total) about the item. If an item is discounted, you may also want to use the fields that show the regular (before any discount is applied) values. More information about discounts and when they are available is provided in the discount section below.		
item.taxExempt	2.0	Indicates whether this item is exempt from taxes or not. The following code prints a plus character if the item is exempt from taxes. [#if item.taxExempt] + [/#if] Items can be marked as exempt from taxes only if taxes are enabled for your business.
item.price	1.0, 2.0	The price used to bill the item. If the item is discounted, it indicates the **price after the discount**. Note: **item.price** replaces **item.rate** which was used up until version 1.11.
item.quantity	1.0, 2.0	The quantity used to bill the item. If the item is discounted, it indicates the **quantity after the discount**. If the item is billed in hours, this field indicates the number of hours in decimal format. Otherwise it indicates the number of units. Note: **item.quantity** replaces **item.units** which was used up until version 1.11.

Property	Version	Meaning

item.quantityType	2.0	Indicates whether a service item is billed in units or hours. Expense and product items are always billed in units.
		
		`[#if item.quantityType.id == 0]`
		`  units`
		`[/#if]`
		
		`[#if item.quantityType.id == 1]`
		`  hours`
		`[/#if]`

item.unitOfMeasure	2.3	self-explained
item.total	1.0, 2.0	The amount of money that is charged for the item. If the item is discounted, it indicates the **amount after all discounts**. Note: **item.total** replaces **item.amount** which was used up until version 1.11.

item.regularPrice	1.8, 2.0	If the price of an item is discounted, this field indicates the **price before the discount**. You may want to use this field like this:
		
		`[#if item.priceDiscounted]`
		`  ${item.regularPrice}`
		`[/#if]`
		
		The regular price is accessed only when the price is discounted.
		
		Note: **item.regularPrice** replaces **item.regularRate** which was used up until version 1.11.

item.regularQuantity	1.8, 2.0	If the quantity of an item is discounted, this field indicates the **quantity before the discount**. You may want to use this field like this:
		
		`[#if item.quantityDiscounted]`
		`  ${item.regularQuantity}`
		`[/#if]`
		
		The regular quantity is accessed only when it is discounted.
		
		Note: **item.regularQuantity** replaces **item.regularUnits** which was used up until version 1.11.

Property	Version	Meaning

item.regularTotal	1.8, 2.0	If an item is discounted, this field indicates the total **amount for the item before any discount is applied** to it. You may want to use this field like this:
		
		`[#if item.discounted]`
		`    ${item.regularTotal}`
		`[/#if]`
		
		The regular total is accessed only when it is discounted.
		
		Note: **item.regularTotal** replaces **item.regularAmount** which was used up until version 1.11.

3. Discount The discount fields can be used to determine the discounts for the price or quantity of an item. There are also a few fields that can be used to determine if a discount has been applied to an item or not.		
item.priceDiscount	1.8, 2.0	If an item is discounted by price, this field indicates the discount. It can be used to display the actual discount. Note: **item.priceDiscount** replaces **item.rateDiscount** which was used up until version 1.11.
item.quantityDiscount	1.8, 2.0	If an item is discounted by quantity, this field indicates the discount. It can be used to display the actual discount. Note: **item.quantityDiscount** replaces **item.unitsDiscount** which was used up until version 1.11.
item.priceDiscounted	1.8, 2.0	Indicates if the price of an item is discounted. See the **item.regularPrice** field above. Note: **item.priceDiscounted** replaces **item.rateDiscounted** which was used up until version 1.11.
item.quantityDiscounted	1.8, 2.0	Indicates if the quantity of an item is discounted. See the **item.regularQuantity** field above. Note: **item.quantityDiscounted** replaces **item.unitsDiscounted**

Property	Version	Meaning
		which was used up until version 1.11.
item.discounted	1.8	Indicates if an item is discounted (either by rate or number of units). See the **item.regularAmount** field above.
4. Time The time fields can be used to get more detailed information about the time recorded for a service item.		

item.timeEntries	1.5	A list with all the time entries recorded for a service item. This list is useful if you want to create a very detailed invoice that shows a breakdown for each invoiced item.
		
		To access a time entry from the list use the following Freemarker code:
		
		`[#list item.timeEntries as timeEnt`
		`...`
		`[/#list]`

item.elapsedTimeAsHour	1.5	The total time recorded for a service item in hour format. For instance 1 hour and 30 minutes is represented as 1:30.
item.elapsedTimeAsDecimal	1.5	The total time recorded for a service item in decimal format. For instance 1 hour and 30 minutes is represented as 1.50.
item.billingTimeAsHour	1.5, 3.1, 4.0	The total billing time of a service item in hour format. For instance 1 hour and 30 minutes is represented as 1:30. Note: **item.billingTimeAsHour** replaces **item.billableTimeAsHour** which was used up until version 3.0 and **item.billedTimeAsHour** which was used up until version 4.0.
item.billingTimeAsDecimal	1.5, 3.1, 4.0	The total billing time of a service item in decimal format. For instance 1 hour and 30 minutes is represented as 1.50. Note: **item.billingTimeAsDecimal** replaces **item.billableTimeAsDecimal**

Property	Version	Meaning
		which was used up until version 3.0 and **item.billedTimeAsDecimal** which was used up until version 4.0.
5. Trips The trip fields can be used to get more detailed information about the trips recorded for a mileage item.		

item.trips	3.0	A list with all the trips recorded for a mileage item. This list is useful if you want to create a very detailed invoice that shows a breakdown for each invoiced item.
		
		To access a trip from the list use the following Freemarker code:
		
		`[#list item.trips as trip]`
		`...`
		`[/#list]`

4.9.5. Payment

The following table shows the fields of a payment made to an invoice. A payment cannot be accessed directly. You can only access it from the list of payments of an invoice (**invoice.payments**).

Table 4.44. Payment Properties

Property	Version	Meaning
payment.date	2.0	self-explained
payment.notes	2.0	self-explained
payment.amount	2.0	self-explained
payment.reference	2.0	self-explained

4.9.6. Deposit

The following table shows the fields of a deposit transaction. A deposit cannot be accessed directly. You can only access it from the list of deposits of a client (**client.deposits**).

Table 4.45. Deposit Properties

Property	Version	Meaning
deposit.date	3.1	self-explained
deposit.notes	3.1	self-explained
deposit.amount	3.1	The amount deposited in or withdrawn from the deposits account. Positive amounts are deposited while negative amounts are withdrawn.
deposit.reference	3.1	self-explained
deposit.invoiceDeposit	3.1	Indicates whether this deposit was used to pay an invoice.

Property	Version	Meaning

		The following code prints the number of the invoice paid by this deposit.
		
		`[#if deposit.invoiceDeposit]`
		`  ${deposit.invoiceNumber}`
		`[/#if]`
		
		To access the list of deposits of a client see the **client.deposits** placeholder.

deposit.invoiceNumber	3.1	See **deposit.invoiceDeposit**.

4.9.7. Tax

The following table shows the fields of a tax from an invoice. A tax cannot be accessed directly. You can only access it from the list of taxes of an invoice (**invoice.taxes**).

Table 4.46. Tax Properties

Property	Version	Meaning
tax.name	1.4	self-explained

4.9.8. Invoice Project

The following table shows the fields of an invoice project. A project cannot be accessed directly. You have to access it from the list of projects of an invoice (**invoice.projects**).

Usually, you don't need to work with invoice projects. They are useful only if the invoice is for multiple projects.

Table 4.47. Project Properties

Property	Version	Meaning
1. General		
project.name	1.0	self-explained
project.number	2.6	self-explained
project.reference	2.6	self-explained
project.description	2.5	self-explained
project.notes	1.8	self-explained
2. Contents An invoice project contains items that can be service, expense, mileage or product items. The following fields let you access them.		

project.items	1.0, 2.0	A list with all the items of an invoice project. To access an item from the list use the following Freemarker code:
		
		`[#list project.items as item]`
		`...`
		`[/#list]`

Property	Version	Meaning
		Since version 2.0, this placeholder has a new meaning. The list contains all items, including expense items. Previously, it contained only services. Instead of accessing the entire list of items, one can access the services and the expenses separately using the **project.serviceItems**, **project.productItems** and the **project.expenseItems** placeholders.

project.serviceItems	2.0, 2.7	A list with all the service items of an invoice project. To access a service item from the list use the following Freemarker code:
		
		`[#list project.serviceItems as ite`
		`...`
		`[/#list]`
		
		Note: **project.serviceItems** replaces **project.services** which was used up until version 2.6.

project.expenseItems	1.0, 2.7	A list with all the expense items of an invoice project. To access an expense item from the list use the following Freemarker code:
		
		`[#list project.expenseItems as ite`
		`...`
		`[/#list]`
		
		Note: **project.expenseItems** replaces **project.expenses** which was used up until version 2.6.

project.mileageItems	3.0	A list with all the mileage items of an invoice project. To access a mileage item from the list use the following Freemarker code:
		
		`[#list project.mileageItems as ite`
		`...`
		`[/#list]`

project.productItems	2.5, 2.7	A list with all the product items of an invoice project. To access a product item from the list use the following Freemarker code:
		
		`[#list project.productItems as ite`
		`...`
		`[/#list]`

Property	Version	Meaning
		Note: **project.productItems** replaces **project.products** which was used up until version 2.6.
3. Totals		
project.serviceItemsSubtotal	2.0, 2.7	The total amount of money due for all the service items from this project. Note: **project.serviceItemsSubtotal** replaces **project.servicesSubtotal** which was used up until version 2.6.
project.expenseItemsSubtotal	1.0, 2.7	The total amount of money due for all the expense items from this project. Note: **project.expenseItemsSubtotal** replaces **project.expensesSubtotal** which was used up until version 2.6.
project.mileageItemsSubtotal	3.0	The total amount of money due for all the mileage items from this project.
project.productItemsSubtotal	2.5, 2.7	The total amount of money due for all the product items from this project. Note: **project.productItemsSubtotal** replaces **project.productsSubtotal** which was used up until version 2.6.
project.total	1.0	The total project amount of money (no taxes included).
4. Time		
project.billingTimeAsHour	1.5, 3.1, 4.0	The total time invoiced for a project in hour format. For instance 1 hour and 30 minutes is represented as 1:30. Note: **project.billingTimeAsHour** replaces **project.billableTimeAsHour** which was used up until version 3.0 and **project.billedTimeAsHour** which was used up until version 4.0.

Property	Version	Meaning
project.billingTimeAsDecimal	1.5, 3.1, 4.0	The total time invoiced for a project in decimal format. For instance 1 hour and 30 minutes is represented as 1.50. Note: **project.billingTimeAsDecimal** replaces **project.billableTimeAsDecimal** which was used up until version 3.0 and **project.billedTimeAsDecimal** which was used up until version 4.0.

4.9.9. Time Entry

The following table shows the fields of a time entry. A time entry cannot be accessed directly. You have to access it from the list of time entries of a project item (**item.timeEntries**).

Usually, you don't need to work with time entries. They are useful only if you need to create a very detailed invoice that breaks down the time spent on each item.

Table 4.48. Time Entry Properties

Property	Version	Meaning
timeEntry.date	1.5	Use ${timeEntry.date?time} to access the exact time when time has been recorded for the item.
timeEntry.endDate	1.5	Use ${timeEntry.endDate?time} to access the exact time recording has stopped.
timeEntry.description	1.5	self-explained
timeEntry.elapsedTimeAsHour	1.5	The total recorded time in hour format. For instance 1 hour and 30 minutes is represented as 1:30.
timeEntry.elapsedTimeAsDecimal	1.5	The total recorded time in decimal format. For instance 1 hour and 30 minutes is represented as 1.50.
timeEntry.pausedTimeAsHour	1.8	The pause associated with this time entry in hour format. For instance 1 hour and 30 minutes is represented as 1:30.
timeEntry.pausedTimeAsDecimal	1.8	The pause associated with this time entry in decimal format. For instance 1 hour and 30 minutes is represented as 1.50.
timeEntry.billingTimeAsHour	3.1, 4.0	The total billing time in hour format. For instance 1 hour and 30 minutes is represented as 1:30.

Property	Version	Meaning
		Note: **timeEntry.billingTimeAsHour** replaces **timeEntry.roundedElapsedTimeAsHour** which was used up until version 4.0.
timeEntry.billingTimeAsDecimal	3.1, 4.0	The total billing time recorded in decimal format. For instance 1 hour and 30 minutes is represented as 1.50. Note: **timeEntry.billingTimeAsDecimal** replaces **timeEntry.roundedElapsedTimeAsDecimal** which was used up until version 4.0.
timeEntry.units	4.0	self-explained Here's how to use it in order to display the number of units as an integer: `${timeEntry.units?string.number}`
timeEntry.unitsPerHour	4.0	self-explained The calculated value can be null if the time is zero (00:00:00). The following example shows the best way to use this placeholder: `${timeEntry.unitsPerHour!"N/A"}`
timeEntry.timePerUnitAsHour	4.0	self-explained The calculated value can be null if the number of units is zero. The following example shows the best way to use this placeholder: `${timeEntry.timePerUnitAsHour!"N/A`

4.9.10. Task

The following table shows the fields of a task. A task cannot be accessed directly. You have to access it from a project service item (**item.task**).

Usually, you don't need to work with tasks. They are useful only if you need to create a very detailed invoice that includes details not available in a service item (eg the task reference number).

Table 4.49. Task Properties

Property	Version	Meaning
1. General		

Property	Version	Meaning
task.name	3.0	self-explained
task.reference	3.0	self-explained
task.category	3.0	self-explained
task.description	3.0	self-explained
task.notes	3.0	self-explained
task.tags	3.0	self-explained
2. Planning		
task.startDate	3.0	self-explained
task.dueDate	3.0	self-explained
task.completedDate	3.0	self-explained

task.completed	3.0	Indicates whether the task is completed or not. If you want to do something when the task **is** completed:
		`[#if task.completed]`
		`...`
		`[/#if]`
		If you want to do something when the task **isn't** completed:
		`[#if !task.completed]`
		`...`
		`[/#if]`

task.overdue	3.0	Indicates whether the task is overdue or not. If you want to do something when the task **is** overdue:
		`[#if task.overdue]`
		`...`
		`[/#if]`
		If you want to do something when the task **isn't** overdue:
		`[#if !task.overdue]`
		`...`
		`[/#if]`

3. Time tracking		

task.timeEntries	3.0	A list with all the time entries of a task. To access a time entry from the list use the following Freemarker code:
		`[#list task.timeEntries as timeEnt`
		`...`
		`[/#list]`

task.elapsedTimeAsHour	3.0	The total time recorded in hour format. For instance 1 hour and

Property	Version	Meaning
		30 minutes is represented as 1:30.
task.estimatedTimeAsHour	3.0	The total estimated time in hour format. For instance 1 hour and 30 minutes is represented as 1:30.
task.remainingTimeAsHour	3.0	The total remaining time in hour format. For instance 1 hour and 30 minutes is represented as 1:30.

4.9.11. Expense

The following table shows the fields of an expense. An expense cannot be accessed directly. You have to access it from a project expense item (**item.expense**).

Usually, you don't need to work with expenses. They are useful only if you need to create a very detailed invoice that includes details not available in an expense item (eg the expense reference number).

Table 4.50. Expense Properties

Property	Version	Meaning
expense.reference	3.0	self-explained
expense.category	3.0	self-explained
expense.description	3.0	self-explained
expense.notes	3.0	self-explained
expense.date	3.0	self-explained
expense.amount	3.0	self-explained

4.9.12. Trip

The following table shows the fields of a trip. A trip cannot be accessed directly. You have to access it from the list of trips of a project item (**item.trips**).

Usually, you don't need to work with trips. They are useful only if you need to create a very detailed invoice that shows all the trips of each mileage item.

Table 4.51. Trip Properties

Property	Version	Meaning
trip.startTime	3.0	Use ${trip.startTime?time} to access the time when the trip started. If you are only interested in the date, use ${trip.startTime?date}.
trip.endTime	3.0	Use ${trip.endTime?time} to access the time when the trip ended.
trip.durationAsHour	3.0	The duration of the trip in hour format. For instance 1 hour and

Property	Version	Meaning
		30 minutes is represented as 1:30.
trip.durationAsDecimal	3.0	The duration of the trip in decimal format. For instance 1 hour and 30 minutes is represented as 1.50.
trip.distance	3.0	The distance recorded by the trip. ${trip.distance} represents distance as value followed by unit (eg 30 km). If you need to access each field separately then you should use ${trip.distance.value} and ${trip.distance.unit}.
trip.startLocation	3.0	self-explained
trip.endLocation	3.0	self-explained
trip.description	3.0	self-explained
trip.tags	3.0	self-explained

4.9.13. Person

The following table shows the fields of a person. A person cannot be accessed directly. You can only access it from a report (**timereport.groupPeople(timeEntries)**).

Table 4.52. Person Properties

Property	Version	Meaning
person.name	4.0	self-explained

4.9.14. Projects Report

The following table shows the placeholders that can be used to create projects reports. Besides these placeholders you can also use business placeholders to print information about your business in a projects report.

Table 4.53. Projects Report Properties

Property	Version	Meaning

1. Parameters
 	
The date interval used to build the report can be accessed from the template.
 	
Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *projectsreport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if projectsreport.dateIntervalStart??]`
`[#if projectsreport.dateIntervalEnd??]`
`${projectsreport.dateIntervalStart?date} - ${projectsreport.dateIntervalEnd?da`
`[#else]`
`After ${projectsreport.dateIntervalStart?date}`
`[/#if]`
`[#else]`		

Property	Version	Meaning

`[#if projectsreport.dateIntervalEnd??]`
`Before ${projectsreport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

projectsreport.dateIntervalStart	3.1	self-explained
projectsreport.dateIntervalEnd	3.1	self-explained
2. Projects The report allows you to obtain a list with all its projects and project elements (tasks, expenses, trips and products).		
projectsreport.projects	3.1	A list with all the projects included in the report. This is probably most important placeholder because almost all reports need to use it.
projectsreport.getTimeEntries(task)	3.1	A list with all the time entries of a task that belong to the reporting period.
projectsreport.getTasks(project)	3.1	A list with all the tasks of a project that belong to the reporting period.
projectsreport.getTrips(project)	3.1	A list with all the trips of a project that belong to the reporting period.
projectsreport.getExpenses(project)	3.1	A list with all the expenses of a project that belong to the reporting period.
projectsreport.getProducts(project)	3.1	A list with all the products of a project that belong to the reporting period.
projectsreport.filterWithActivity(projects)	3.1	Finds the subset of the specified projects that have elements (tasks, time, expenses, trips or products) in the reporting period. This filter is useful if you don't want to deal with projects that were active during the reporting period but nothing was recorded on them.
projectsreport.filterByClient(projects, client)	3.1	Finds the subset of the specified projects that belong to the specified client.
3. Grouping Fields like clients can be extracted as separate groups from a list of projects.		

projectsreport.groupClients(projects)	3.1	A list with all the clients.
		
		`[#list projectssreport.groupClient`
		`...`

Property	Version	Meaning
		`[/#list]`
4. Totals The report can calculate totals for a list of projects. The list of projects can either be *projectsreport.projects* (all the projects included in the report) or a partial list determined by one of the filter functions.		
projectsreport.calculateTaskTotal(task)	3.1	The total for the specified task.
projectsreport.calculateElapsedTimeAsHour(task)	3.1	The total elapsed time in hour format for the specified task.
projectsreport.calculateElapsedTimeAsHour(project)	3.1	The total elapsed time in hour format for the specified project.
projectsreport.calculateElapsedTimeAsHour(projects)	3.1	The total elapsed time in hour format for the specified list of projects.
projectsreport.calculateExpensesAmount(project)	3.1	The total expenses amount for the specified project.
projectsreport.calculateExpensesAmount(projects)	3.1	The total expenses amount for the specified list of projects.
projectsreport.calculateTripsDistance(project)	3.1	The total distance for the specified project.
projectsreport.calculateTripsDistance(projects)	3.1	The total distance for the specified list of projects.
projectsreport.calculateTotal(project)	3.1	The total for the specified project.
projectsreport.calculateTotal(projects)	3.1	The total for the specified list of projects.
projectsreport.calculateTasksTotal(project)	3.1	The tasks subtotal for the specified project.
projectsreport.calculateExpensesTotal(project)	3.1	The expenses subtotal for the specified project.
projectsreport.calculateTripsTotal(project)	3.1	The trips subtotal for the specified project.
projectsreport.calculateProductsTotal(project)	3.1	The products subtotal for the specified project.

4.9.15. Time Report

The following table shows the placeholders that can be used to create time reports. Besides these placeholders you can also use business placeholders to print information about your business in a time report.

Here's an example that shows how to calculate totals by date. It groups time entries by date and then it calculates the total elapsed time for each date.

```
[!-- Determine the dates when time was recorded. --]
[#assign dates=timereport.groupDatesByDate(timereport.timeEntries)]

[#list dates?sort as date]

  [!-- Use timereport.filterByDate to find time entries recorded on a specific
  [#assign dateTimeEntries=timereport.filterByDate(timereport.timeEntries, date
```

```
  [!-- Print the date and the total elapsed time for this date. --]
  ${date.toDate()?date}: ${timereport.calculateElapsedTimeAsHour(dateTimeEntrie

[/#list]

[!-- Calculate the total elapsed time for all the time entries included in the
Total: ${timereport.calculateElapsedTimeAsHour(timereport.timeEntries)}
```

Table 4.54. Time Report Properties

Property	Version	Meaning

1. Parameters
 	
The date interval used to build the report can be accessed from the template.
 	
Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *timereport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if timereport.dateIntervalStart??]`
`[#if timereport.dateIntervalEnd??]`
`${timereport.dateIntervalStart?date} - ${timereport.dateIntervalEnd?date}`
`[#else]`
`After ${timereport.dateIntervalStart?date}`
`[/#if]`
`[#else]`
`[#if timereport.dateIntervalEnd??]`
`Before ${timereport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

timereport.dateIntervalStart	3.0	self-explained
timereport.dateIntervalEnd	3.0	self-explained
2. Time entries The report allows you to obtain a list with all its time entries or partial lists (for instance time entries that were recorded in a certain month).		
timereport.timeEntries	3.0	A list with all the time entries included in the report. This is probably most important placeholder because almost all reports need to use it.
timereport.filterByDate(timeEntries, interval)	3.0	Finds the subset of the specified time entries that were recorded in the specified interval. (their Date field belongs to the interval)
timereport.filterByTask(timeEntries, task)	3.0	Finds the subset of the specified time entries that belong to the specified task.
timereport.filterByTaskCategory(timeEntries, taskCategory)	3.2	Finds the subset of the specified time entries recorded on tasks that have the specified task category.

Property	Version	Meaning
timereport.filterByTaskCategoryNone(timeEntries)	3.2	Finds the subset of the specified time entries recorded on tasks that don't have a task category.
timereport.filterByProject(timeEntries, project)	3.0	Finds the subset of the specified time entries that belong to the specified project.
timereport.filterByClient(timeEntries, client)	3.0	Finds the subset of the specified time entries that belong to the specified client.
timereport.filterByPerson(timeEntries, person)	4.0	Finds the subset of the specified time entries that were created by the specified person.
3. Grouping Fields like clients or dates can be extracted as separate groups from a list of time entries.		

timereport.groupPeople(timeEntries)	4.0	A list with all the people.
		
		`[#list timereport.groupPeople(time`
		`...`
		`[/#list]`

timereport.groupClients(timeEntries)	3.0	A list with all the clients.
		
		`[#list timereport.groupClients(tim`
		`...`
		`[/#list]`

timereport.groupProjects(timeEntries)	3.0	A list with all the projects.
		
		`[#list timereport.groupProjects(ti`
		`...`
		`[/#list]`

timereport.groupTasks(timeEntries)	3.0	A list with all the tasks.
		
		`[#list timereport.groupTasks(timer`
		`...`
		`[/#list]`

timereport.groupTaskCategories(timeEntries)	3.2	A list with all the task categories.
		
		`[#list timereport.groupTaskCategor`
		`...`
		`[/#list]`

timereport.groupDatesByDate(timeEntries)	3.0	A list with the dates when the time entries included in the report were recorded.
		
		`[#list timereport.groupDatesByDate`
		`...`
		`[/#list]`
		
		Use **${date.toDate()?date}** to print each date and **${date.toInterval()}** to create an interval for a date.

Property	Version	Meaning

timereport.groupDatesByWeek(timeEntries)	3.0	A list with the weeks when the time entries included in the report were recorded.
		
		`[#list timereport.groupDatesByWeek`
		`...`
		`[/#list]`
		
		Use **${week.toString("yyyy-ww")}** to print each week and **${week.toInterval()}** to create an interval for a week.

timereport.groupDatesByMonth(timeEntries)	3.0	A list with the months when the time entries included in the report were recorded.
		
		`[#list timereport.groupDatesByMont`
		`...`
		`[/#list]`
		
		Use **${month.toString("MMM yyyy")}** to print each month and **${month.toInterval()}** to create an interval for a month.

timereport.groupDatesByYear(timeEntries)	3.0	A list with the years when the time entries included in the report were recorded.
		
		`[#list timereport.groupDatesByYear`
		`...`
		`[/#list]`
		
		Use **${year}** to print each year and **${year.toInterval()}** to create an interval for an year.

4. Totals The report can calculate totals for a list of time entries. The list of time entries can either be *timereport.timeEntries* (all the time entries included in the report) or a partial list determined by one of the filter functions.		
timereport.calculateElapsedTimeAsHour(timeEntries)	3.0	The total elapsed time in hour format for the specified list of time entries.
timereport.calculatePausedTimeAsHour(timeEntries)	3.0	The total pause time in hour format for the specified list of time entries.
timereport.calculateBillingTimeAsHour(timeEntries)	3.1.4.0	The total billing time in hour format for the specified list of time entries. Note: **timereport.calculateBillingTimeAsHour** replaces **timereport.calculateRoundedElapsedTimeAsHo**

Property	Version	Meaning
		which was used up until version 4.0.
timereport.calculateUnits(timeEntries)	4.0	The total units for the specified list of time entries.
timereport.calculateUnitsPerHour(timeEntries)	4.0	The total units per hour for the specified list of time entries. The calculated value can be null if the total time is zero (00:00:00). The following example shows the best way to use this placeholder: `${timereport.calculateUnitsPerHour`
timereport.calculateTimePerUnitAsHour(timeEntries)	4.0	The time per unit for the specified list of time entries. The calculated value can be null if the number of units is zero. The following example shows the best way to use this placeholder: `${timereport.calculateTimePerUnitA`
timereport.calculateElapsedTimeAsDecimal(timeEntries)	3.0	The total elapsed time in decimal format for the specified list of time entries.
timereport.calculatePausedTimeAsDecimal(timeEntries)	3.0	The total pause time in decimal format for the specified list of time entries.
timereport.calculateBillingTimeAsDecimal(timeEntries)	3.0, 4.0	The total billing time in decimal format for the specified list of time entries. Note: **timereport.calculateBillingTimeAsDecimal** replaces **timereport.calculateRoundedElapsedTimeAsDe** which was used up until version 4.0.

4.9.16. Tasks Report

The following table shows the placeholders that can be used to create tasks reports. Besides these placeholders you can also use business placeholders to print information about your business in a tasks report.

Table 4.55. Tasks Report Properties

Property	Version	Meaning
1. Parameters The date interval used to build the report can be accessed from the template.		

Property	Version	Meaning

Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *tasksreport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if tasksreport.dateIntervalStart??]`
`[#if tasksreport.dateIntervalEnd??]`
`${tasksreport.dateIntervalStart?date} - ${tasksreport.dateIntervalEnd?date}`
`[#else]`
`After ${tasksreport.dateIntervalStart?date}`
`[/#if]`
`[#else]`
`[#if tasksreport.dateIntervalEnd??]`
`Before ${tasksreport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

tasksreport.dateIntervalStart	3.0	self-explained
tasksreport.dateIntervalEnd	3.0	self-explained
2. Tasks The report allows you to obtain a list with all its tasks or partial lists (for instance tasks that were recorded for a certain project).		
tasksreport.tasks	3.0	A list with all the tasks included in the report. This is probably most important placeholder because almost all reports need to use it.
tasksreport.getTimeEntries(task)	3.0	A list with all the time entries of task that belong to the reporting period.
tasksreport.filterWithTimeOrMoney(tasks)	3.0	Finds the subset of the specified tasks that have time or money in the reporting period. This filter is useful if you don't want to deal with tasks that were active during the reporting period but nothing was recorded on them.
tasksreport.filterByProject(tasks, project)	3.0	Finds the subset of the specified tasks that belong to the specified project.
tasksreport.filterByClient(tasks, client)	3.0	Finds the subset of the specified tasks that belong to the specified client.
3. Grouping Fields like clients or projects can be extracted as separate groups from a list of tasks.		

tasksreport.groupClients(tasks)	3.0	A list with all the clients.
		
		`[#list tasksreport.groupClients(ta`
		`...`

Property	Version	Meaning
		`[/#list]`

tasksreport.groupProjects(tasks)	3.0	A list with all the projects.
		
		`[#list tasksreport.groupProjects(t`
		`...`
		`[/#list]`

4. Totals The report can calculate totals for a list of tasks. The list of tasks can either be *tasksreport.tasks* (all the tasks included in the report) or a partial list determined by one of the filter functions.		
tasksreport.calculateEstimatedTimeAsHour(tasks)	3.1	The total estimated time in hour format for the specified list of tasks.
tasksreport.calculateEstimatedTimeAsDecimal(tasks)	3.2	The total estimated time in decimal format for the specified list of tasks.
tasksreport.calculateRemainingTimeAsHour(tasks)	3.1	The total remaining time in hour format for the specified list of tasks.
tasksreport.calculateRemainingTimeAsDecimal(tasks)	3.2	The total remaining time in decimal format for the specified list of tasks.
tasksreport.calculateElapsedTimeAsHour(task)	3.0	The total elapsed time in hour format for the specified task.
tasksreport.calculateElapsedTimeAsHour(tasks)	3.0	The total elapsed time in hour format for the specified list of tasks.
tasksreport.calculateElapsedTimeAsDecimal(tasks)	3.2	The total elapsed time in decimal format for the specified list of tasks.
tasksreport.calculateBillableTotal(task)	3.0	The billable total for the specified task. How much money can be invoiced for the task in the reporting period.
tasksreport.calculateBillableTotal(tasks)	3.0	The billable total for the specified list of tasks. How much money can be invoiced for the list of tasks in the reporting period.
tasksreport.calculateBilledTotal(task)	3.0	The billed total for the specified task. How much money was invoiced for the task in the reporting period. For instance, if a task started on January 1st and it was invoiced on February 10th for the whole month of January for $100 then

Property	Version	Meaning
		the billed total for January will be $100.
tasksreport.calculateBilledTotal(tasks)	3.0	The billed total for the specified list of tasks. How much money was invoiced for the list of tasks in the reporting period.
tasksreport.calculateTotal(task)	3.0	The total (billable + billed) for the specified task. How much money is the task worth in the reporting period.
tasksreport.calculateTotal(tasks)	3.0	The total (billable + billed) for the specified list of tasks. How much money is the list of tasks worth in the reporting period.

4.9.17. Expenses Report

The following table shows the placeholders that can be used to create expenses reports. Besides these placeholders you can also use business placeholders to print information about your business in an expenses report.

Table 4.56. Expenses Report Properties

Property	Version	Meaning

1. Parameters
 	
The date interval used to build the report can be accessed from the template.
 	
Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *expensesreport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if expensesreport.dateIntervalStart??]`
`[#if expensesreport.dateIntervalEnd??]`
`${expensesreport.dateIntervalStart?date} - ${expensesreport.dateIntervalEnd?da`
`[#else]`
`After ${expensesreport.dateIntervalStart?date}`
`[/#if]`
`[#else]`
`[#if expensesreport.dateIntervalEnd??]`
`Before ${expensesreport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

expensesreport.dateIntervalStart	3.1	self-explained
expensesreport.dateIntervalEnd	3.1	self-explained
2. Expenses		

Property	Version	Meaning
The report allows you to obtain a list with all its expenses or partial lists (for instance expenses that were recorded for a certain project).		
expensesreport.expenses	3.1	A list with all the expenses included in the report. This is probably most important placeholder because almost all reports need to use it.
expensesreport.filterByDate(expenses, interval)	3.2	Finds the subset of the specified expenses that were recorded in the specified interval. (their Date field belongs to the interval)
expensesreport.filterByProject(expenses, project)	3.1	Finds the subset of the specified expenses that belong to the specified project.
expensesreport.filterByClient(expenses, client)	3.1	Finds the subset of the specified expenses that belong to the specified client.
expensesreport.filterByPerson(expenses, person)	4.0	Finds the subset of the specified expenses that were created by the specified person.
3. Grouping Fields like clients or projects can be extracted as separate groups from a list of expenses.		

expensesreport.groupPeople(expenses)	4.0	A list with all the people.
		
		`[#list expensesreport.groupPeople(`
		`...`
		`[/#list]`

expensesreport.groupClients(expenses)	3.1	A list with all the clients.
		
		`[#list expensesreport.groupClients`
		`...`
		`[/#list]`

expensesreport.groupProjects(expenses)	3.1	A list with all the projects.
		
		`[#list expensesreport.groupProject`
		`...`
		`[/#list]`

expensesreport.groupDatesByDate(expenses)	3.2	A list with the dates when the expenses included in the report were recorded.
		
		`[#list expensesreport.groupDatesBy`
		`...`
		`[/#list]`
		
		Use **${date.toDate()?date}** to print each date and **${date.toInterval()}** to create an interval for a date.

expensesreport.groupDatesByWeek(expenses)	3.2	A list with the weeks when the expenses included in the report were recorded.

Property	Version	Meaning

		`[#list expensesreport.groupDatesBy`
		`...`
		`[/#list]`
		
		Use **${week.toString("yyyy-ww")}** to print each week and **${week.toInterval()}** to create an interval for a week.

expensesreport.groupDatesByMonth(expenses)	3.2	A list with the months when the expenses included in the report were recorded.
		
		`[#list expensesreport.groupDatesBy`
		`...`
		`[/#list]`
		
		Use **${month.toString("MMM yyyy")}** to print each month and **${month.toInterval()}** to create an interval for a month.

expensesreport.groupDatesByYear(expenses)	3.2	A list with the years when the expenses included in the report were recorded.
		
		`[#list expensesreport.groupDatesBy`
		`...`
		`[/#list]`
		
		Use **${year}** to print each year and **${year.toInterval()}** to create an interval for an year.

4. Totals The report can calculate totals for a list of expenses. The list of expenses can either be *expensesreport.expenses* (all the expenses included in the report) or a partial list determined by one of the filter functions.		
expensesreport.calculateAmount(expenses)	3.1	The total amount for the specified list of expenses. Amount specifies the total amount of money spent.
expensesreport.calculateTotal(expenses)	3.1	The total for the specified list of expenses. Total specifies the total amount of money billed if the expense is billable.

4.9.18. Trips Report

The following table shows the placeholders that can be used to create trips reports. Besides these placeholders you can also use business placeholders to print information about your business in a trips report.

Table 4.57. Trips Report Properties

Property	Version	Meaning

1. Parameters
 	
The date interval used to build the report can be accessed from the template.
 	
Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *tripssreport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if tripsreport.dateIntervalStart??]`
`[#if tripsreport.dateIntervalEnd??]`
`${tripsreport.dateIntervalStart?date} - ${tripsreport.dateIntervalEnd?date}`
`[#else]`
`After ${tripsreport.dateIntervalStart?date}`
`[/#if]`
`[#else]`
`[#if tripsreport.dateIntervalEnd??]`
`Before ${tripsreport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

tripsreport.dateIntervalStart	3.1	self-explained
tripsreport.dateIntervalEnd	3.1	self-explained
2. Trips The report allows you to obtain a list with all its trips or partial lists (for instance trips that were recorded for a certain project).		
tripsreport.trips	3.1	A list with all the trips included in the report. This is probably most important placeholder because almost all reports need to use it.
tripsreport.filterByStartTime(trips, interval)	3.2	Finds the subset of the specified trips that were recorded in the specified interval. (their Start Time field belongs to the interval)
tripsreport.filterByProject(trips, project)	3.1	Finds the subset of the specified trips that belong to the specified project.
tripsreport.filterByClient(trips, client)	3.1	Finds the subset of the specified trips that belong to the specified client.
tripsreport.filterByPerson(trips, person)	4.0	Finds the subset of the specified trips that were created by the specified person.
3. Grouping Fields like clients or projects can be extracted as separate groups from a list of trips.		
tripsreport.groupPeople(trips)	4.0	A list with all the people. `[#list tripsreport.groupPeople(tri`

Property	Version	Meaning
		`...` `[/#list]`

tripsreport.groupClients(trips)	3.1	A list with all the clients.
		`[#list tripsreport.groupClients(tr`
		`...`
		`[/#list]`

tripsreport.groupProjects(trips)	3.1	A list with all the projects.
		`[#list tripsreport.groupProjects(t`
		`...`
		`[/#list]`

tripsreport.groupStartTimesByDate(trips)	3.2	A list with the dates when the trips included in the report were recorded.
		`[#list tripsreport.groupStartTimes`
		`...`
		`[/#list]`
		Use **${date.toDate()?date}** to print each date and **${date.toInterval()}** to create an interval for a date.

tripsreport.groupStartTimesByWeek(trips)	3.2	A list with the weeks when the trips included in the report were recorded.
		`[#list tripsreport.groupStartTimes`
		`...`
		`[/#list]`
		Use **${week.toString("yyyy-ww")}** to print each week and **${week.toInterval()}** to create an interval for a week.

tripsreport.groupStartTimesByMonth(trips)	3.2	A list with the months when the trips included in the report were recorded.
		`[#list tripsreport.groupStartTimes`
		`...`
		`[/#list]`
		Use **${month.toString("MMM yyyy")}** to print each month and **${month.toInterval()}** to create an interval for a month.

tripsreport.groupStartTimesByYear(trips)	3.2	A list with the years when the trips included in the report were recorded.
		`[#list tripsreport.groupStartTimes`
		`...`
		`[/#list]`

Property	Version	Meaning
		Use **${year}** to print each year and **${year.toInterval()}** to create an interval for an year.
4. Totals The report can calculate totals for a list of trips. The list of trips can either be *tripsreport.trips* (all the trips included in the report) or a partial list determined by one of the filter functions.		
tripsreport.calculateDistance(trips)	3.1	The total distance for the specified list of trips.
tripsreport.calculateTotal(trips)	3.1	The total for the specified list of trips.

4.9.19. Sales Report

The following table shows the placeholders that can be used to create sales reports. Besides these placeholders you can also use business placeholders to print information about your business in a sales report.

Here's an example that shows how to calculate totals by month. It groups invoices by month and then it calculates the total for each month.

```
[!-- Determine the months when invoices where issued. --]
[#assign months=salesreport.groupDatesByMonth(salesreport.invoices)]

[#list months?sort as month]

  [!-- Use salesreport.filterByDate to find invoices issued on a specific month
  [#assign monthInvoices=salesreport.filterByDate(salesreport.invoices, month.t

  [!-- Print the month and calculate the total for the invoices issued on that
  ${month.toString("MMM yyyy")}: ${salesreport.calculateTotal(monthInvoices)}

[/#list]

[!-- Calculate the total for all the invoices included in the report --]
Total: ${salesreport.calculateTotal(salesreport.invoices)}
```

Table 4.58. Sales Report Properties

Property	Version	Meaning

1. Parameters
 	
The date and paid date intervals used to build the report can be accessed from the template.
 	
Each interval may or may not have a start date and an end date. If a date interval is defined only by its start date then the end date will not be defined and *salesreport.dateIntervalEnd??* will return false. The folowing code can be used to cover all situations, whether these fields are defined or not.
 	
`[#if salesreport.dateIntervalStart??]`
`[#if salesreport.dateIntervalEnd??]`
`${salesreport.dateIntervalStart?date} - ${salesreport.dateIntervalEnd?date}`
`[#else]`
`After ${salesreport.dateIntervalStart?date}`
`[/#if]`
`[#else]`
`[#if salesreport.dateIntervalEnd??]`		

Property	Version	Meaning

`Before ${salesreport.dateIntervalEnd?date}`
`[#else]`
`All dates`
`[/#if]`
`[/#if]`		

salesreport.dateIntervalStart	3.0	self-explained
salesreport.dateIntervalEnd	3.0	self-explained
salesreport.paidDateIntervalStart	3.0	self-explained
salesreport.paidDateIntervalEnd	3.0	self-explained
2. Invoices The report allows you to obtain a list with all its invoices or partial lists (for instance invoices that were issued in a certain month).		
salesreport.invoices	3.0	A list with all the invoices included in the report. This is probably most important placeholder because almost all reports need to use it.
salesreport.paidInvoices	3.0	A list with all the paid invoices included in the report.
salesreport.filterByDate(invoices, interval)	3.0	Finds the subset of the specified invoices that were issued in the specified interval. (their Date field belongs to the interval)
salesreport.filterByPaidDate(invoices, interval)	3.0	Finds the subset of the specified invoices that were paid in the specified interval. (their Paid Date field belongs to the interval)
salesreport.filterByClient(invoices, client)	3.0	Finds the subset of the specified invoices that belong to the specified client.
3. Grouping Fields like clients or dates can be extracted as separate groups from a list of invoices. Dates can be grouped by month or year.		

salesreport.groupClients(invoices)	3.0	A list with all the invoiced clients.
		
		`[#list salesreport.groupClients(sa`
		`...`
		`[/#list]`

salesreport.groupDatesByMonth(invoices)	3.0	A list with the months when the invoices included in the report were issued.
		
		`[#list salesreport.groupDatesByMon`
		`...`
		`[/#list]`
		
		Use **${month.toString("MMM yyyy")}** to print each month and

Property	Version	Meaning
		${month.toInterval()} to create an interval for a month.

salesreport.groupDatesByYear(invoices)	3.0	A list with the years when the invoices included in the report were issued.
		
		`[#list salesreport.groupDatesByYea`
		`...`
		`[/#list]`
		
		Use **${year}** to print each year and **${year.toInterval()}** to create an interval for an year.

salesreport.groupPaidDatesByMonth(invoices)	3.0	A list with the months when the invoices included in the report were paid.
salesreport.groupPaidDatesByYear(invoices)	3.0	A list with the years when the invoices included in the report were paid.
4. Totals The report can calculate totals for a list of invoices. The list of invoices can either be *salesreport.invoices* (all the invoices included in the report) or a partial list determined by one of the filter functions.		
salesreport.calculateBalance(invoices)	3.0	The balance for the specified list of invoices.
salesreport.calculatePaymentsTotal(invoices)	3.0	The total amount paid for the specified list of invoices.
salesreport.calculateTotal(invoices)	3.0	The total for the specified list of invoices.
salesreport.calculateGrandTotal(invoices)	3.0	The grand total for the specified list of invoices.
salesreport.calculateTaxesTotal(invoices)	3.0	The taxes total for the specified list of invoices.

Chapter 5. Troubleshooting and maintenance

5.1. Installing Fanurio

5.1.1. Requirements

Fanurio runs on Windows (including Windows 8.1), Mac OS X (10.5 or later, including Mavericks), Linux (including Debian-based distributions like Ubuntu) and other flavors of Unix.

Ubuntu (.deb installer)

Ubuntu comes installed with OpenJDK but we recommend using Oracle Java instead. Follow these steps to install it after you open a Terminal window:

1. Download [http://www.oracle.com/technetwork/java/javase/downloads/index.html] the tar.gz file for Linux from Oracle.

   This guide assumes the file is called *jdk-7-linux-i586.tar.gz* but depending on which version you decide to download, it may have a different name.

2. Untar and copy it to */usr/lib/jvm/* where Java is installed. The new Java launcher will be */usr/lib/jvm/jdk1.7.0/jre/bin/java*.

   sudo tar -zxvf ~/Downloads/jdk-7-linux-i586.tar.gz -C /usr/lib/jvm/

3. Add the new Java installation to the Java alternative lists.

   sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk1.7.0/jre/bin/java 1

4. Make the new Java installation the default one. Run the following command and choose the option that points to the new Java installation.

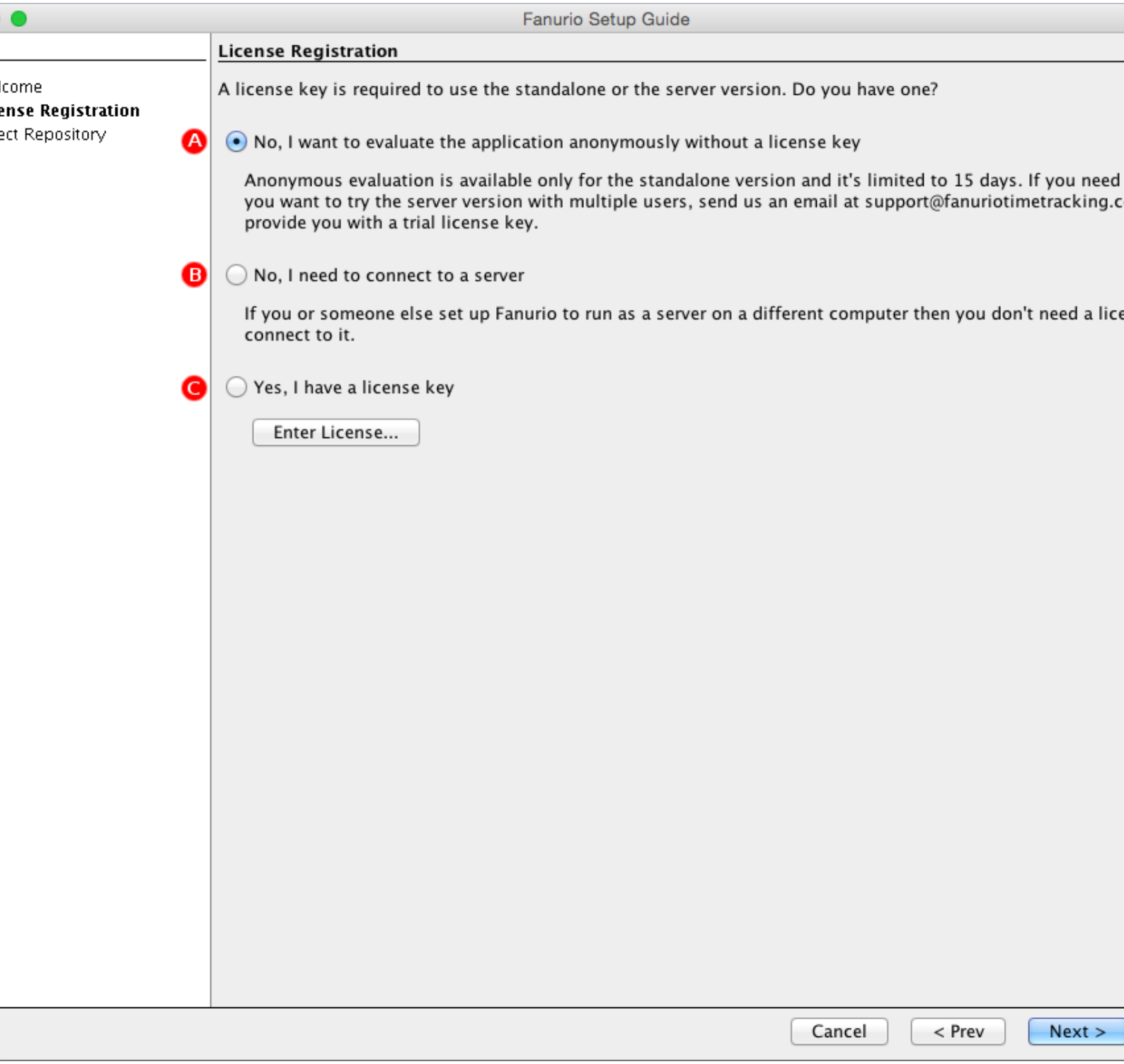
   sudo update-alternatives --config java

5. Test it.

   java -version

5.1.2. Setting up Fanurio for the first time

When you start Fanurio for the first time, it needs to determine if you will be using the standalone version or the server version. It does this by asking whether you have a license key or not.



Fanurio Setup Guide - License Registration

5.1.2.1. The standalone version

If you want to use the standalone version, you can either try it anonymously for 15 days without a license key (option A) or register a standalone license key (option C). You can get a license key either by purchasing a standalone license or by requesting a trial extension.

The setup guide will then help you create a new repository for your data or use an existing one.

- If you don't have a repository, leave the "Yes, I'm a new user and I don't have one" option selected. Fanurio will help you create a new repository and configure your business. Once you create the repository, you can change your business settings from Business » My Business Details.
- If you already have a repository, select "No, I already have one". This option makes sense if you share the repository through Dropbox so you can use it on multiple computers.

Once the setup is completed, you may want to review the default settings (timer reminders, language, etc).

- Go to Tools » Options to access the settings on Windows
- Go to Fanurio » Preferences to access the settings on Mac
- Go to Edit » Preferences to access the settings on Linux

5.1.2.2. The server version

If you want to use the server version, you first need to decide which computer will run the server and which computers will connect to the server.

1. On the computer that runs the server, you need to register a server license key (option C). You can get a license key either by purchasing a server license or by requesting a trial key.

   The setup guide will then help you create a new repository for your data. It will also ask you to create a user so you can log in after the setup is completed.

   After logging in for the first time, go to Business » Manage People to add or invite more people.

   Then go to Server » Server Settings to see the address of your server and copy it. Users will need this address to connect to the server from other computers.

   Finally, go to Server » Start Server to start the server.

2. On the other computers, you need to tell Fanurio that you want to connect to a server (option B).

   The setup guide will then finish and you will be able to log in to a server. To log in, you will need to ask the server administrator for the server address and your access details (username and password).

   The server must be online when you log in for the first time because Fanurio needs to make a local copy of your data. After that, the server doesn't have to be online for you to log in again but it must be online if you need to synchronize local changes.

Once the setup is completed, you may want to review the default settings (timer reminders, language, etc).

- Go to Tools » Options to access the settings on Windows
- Go to Fanurio » Preferences to access the settings on Mac
- Go to Edit » Preferences to access the settings on Linux

5.1.3. Upgrading from the standalone version to the server version

When upgrading from the standalone version to the server version, you first need to decide which computer will run the server and which computers will connect to the server.

1. On the computer that runs the server, you need to register your new license key. If your repository is open, Fanurio will close it and switch to server mode.

Since the server version can handle multiple users, you need to log in to access the repository. By default, a standalone repository has one user called Admin. To log in for the first time, use the **admin** username and leave the password field empty. Once logged in, you can change the name and password from Account » Account Settings.

Next, go to Business » Manage People to add or invite more people.

Then go to Server » Server Settings to see the address of your server and copy it. Users will need this address to connect to the server from other computers.

Finally, go to Server » Start Server to start the server.

2. On the other computers, you need to tell Fanurio that you want to connect to a server. To do this, go to Help » License Registration and select "No, I need to connect to a server". If your repository is open, Fanurio will close it and switch to client mode.

   To log in, you will need to ask the server administrator for the server address and your access details (username and password).

   The server must be online when you log in for the first time because Fanurio needs to make a local copy of your data. After that, the server doesn't have to be online for you to log in again but it must be online if you need to synchronize local changes.

5.1.4. Running Fanurio over SSL (HTTPS)

5.1.4.1. Enabling SSL on the server

A Fanurio server can encrypt communications to and from clients with SSL [https://en.wikipedia.org/wiki/HTTPS] to protect your data from being intercepted by others. To do this, it needs to be configured with a keystore that contains your SSL certificates. Here's how to do it:

1. Go to Server » Server Settings
2. Check Enable SSL

   When you enable SSL, the server can only be accessed over HTTPS. Fanurio does not support both HTTP and HTTPS at the same time. If you have clients that connected to the server over HTTP, once you enable SSL they will have to connect over HTTPS (use the HTTPS address).

3. Locate the keystore file

   If you don't have a keystore file, see the next section to learn how to create one. Fanurio supports only keystores in the JKS format.

4. Enter the keystore password

   Just to be sure, Fanurio encrypts the keystore password when saves it in the settings file.

5.1.4.2. Creating an SSL keystore using the Java keytool

Before you learn how to create a keystore file to hold your SSL certificates so you can enable SSL for a Fanurio server, you need to understand the difference between CA-signed certificates and self-signed certificates:

- **CA-signed certificates**

  These are certificates that have been digitally signed by a Certificate Authority (CA).

  A Certificate Authority is a third-party entity that issues digital certificates to organizations or individuals after verifying their identity. Many CAs simply verify the domain name and issue the

certificate, whereas others verify the existence of your business, the ownership of your domain name, and your authority to apply for the certificate, providing a higher standard of authentication. See this page for a list of CAs [https://www.sslshopper.com/certificate-authority-reviews.html].

CA-signed certificates are recommended for production servers.

- **Self-signed certificates**

  These are certificates that have not been digitally signed by a Certificate Authority (CA).

  A self-signed certificate [https://en.wikipedia.org/wiki/Self-signed_certificate] is an identity certificate that is signed by the same entity whose identity it certifies. This term has nothing to do with the identity of the person or organization that actually performed the signing procedure. In technical terms a self-signed certificate is one signed with its own private key. If a server uses a self-signed certificate, a warning is displayed when you're connecting to it informing you that its identity was not verified by a trusted authority.

  Self-signed certificates are recommended for test and internal servers.

This section shows how to create both types of certificates.

About Java keytool

To generate a keystore, you need a JDK installed and the keytool [https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html] utility on your PATH. You can verify this setup by typing keytool at the shell prompt. You should see a list with all the keytool commands, similar to this:

```
$keytool
Key and Certificate Management Tool

Commands:

 -certreq            Generates a certificate request
 -changealias        Changes an entry's alias
 -delete             Deletes an entry
 -exportcert         Exports certificate
 -genkeypair         Generates a key pair
 -genseckey          Generates a secret key
 -gencert            Generates certificate from a certificate request
 -importcert         Imports a certificate or a certificate chain
 -importpass         Imports a password
 -importkeystore     Imports one or all entries from another keystore
 -keypasswd          Changes the key password of an entry
 -list               Lists entries in a keystore
 -printcert          Prints the content of a certificate
 -printcertreq       Prints the content of a certificate request
 -printcrl           Prints the content of a CRL file
 -storepasswd        Changes the store password of a keystore

Use "keytool -command_name -help" for usage of command_name
```

Creating a keystore with a self-signed certificate

To generate a private key and add it to a keystore, you need to use the **genkeypair** keytool command. The following command creates a keystore file called fanurio.jks in the current directory.

- The "-ext SAN=..." option allows you to specify all the DNS names and/or IP addresses that will be allowed during hostname verification. In addition to localhost and 127.0.0.1, this example includes a LAN IP address 192.168.2.105 for the machine the server will be running on.

- The -dname option contains the server identity, called the Common Name (CN), as well as other relevant information about your Organizational Unit (OU), Organization (O), Locality (L), State (S) and Country (C). If you are using a specific hostname for your server (e.g. fanurio.mydomain.com), use it in the CN field (e.g. CN=fanurio.mydomain.com ...).
- The -keypass and -storepass options must specify the same password.

```
keytool -genkeypair -alias fanurio -keyalg RSA -keysize 2048 -keypass changeit
```

You can verify its contents using the **list** keytool command:

```
keytool -list -v -keystore fanurio.jks
```

The resulting keystore file contains a **self-signed certificate** that can be used to configure your server.

Creating a keystore with a CA-signed certificate

To create a CA-signed certificate for your server, you need a self-signed certificate (read above). Then you need to follow these steps:

1. **Generate CSR**

   First, you need to generate a Certificate Signing Request (CSR) for the CA to sign and confirm the identity of the certificate. To do so, issue the following command:

   ```
   keytool -certreq -v -alias fanurio -file fanurio.csr -keystore fanurio.jks
   ```

   It will create a file called fanurio.csr in your current directory.

2. **Submit CSR to CA**

   Then you need to submit the contents of fanurio.csr to a CA for signing. They will provide a signed certificate (CA reply) and a set of root/intermediate CA certificates that you need to import in the keystore.

   If you want to create CA-signed certificate for tests, you can get a free trial certificate [https://ssl.comodo.com/free-ssl-certificate.php] from Comodo.

3. **Import root certificate from CA**

   The following command imports the root certificate provided by the CA to your keystore. It expects the root certificate to be saved in a file called **root-cert.pem**.

   ```
   keytool -importcert -v -noprompt -trustcacerts -alias cacert -file root-cert.
   ```

4. **Import signed certificate (CA reply)**

   The following command imports the signed certificate provided by the CA to your keystore. It expects the signed certificate to be saved in a file called **signed-cert.pem**.

   ```
   keytool -importcert -v -alias fanurio -file signed-cert.pem -keystore fanurio
   ```

The keystore file now contains a **CA-signed certificate** that can be used to configure your server.

5.1.5. Whitelisting trusted IPs

Setting up a list of trusted IP addresses (whitelist) allows you to control the IPs from which the server can be accessed. This is an optional security measure that should be used with care because you risk blocking access for your users especially if they need to connect to the server from dynamic IP addresses.

To whitelist IP addresses:

- Go to Server » Server Settings and then
- Click on the Trusted IPs tab.

  Once enabled, you need to enter one or more IP addresses (e.g. 192.168.2.105) or ranges of IP addresses (e.g. 192.168.2.105/24) separated by commas. Wildcards are not supported for IP addresses while ranges of IP addresses must be provided in Classless Inter-Domain Routing (CIDR) [https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing] notation.

  You can use a CIDR calculator [http://grox.net/utils/whatmask/] to see the IP address range.

5.1.6. How synchronization works in client mode

When using the server version, Fanurio allows users to work on their projects even if the server is not online. This is possible because Fanurio keeps a local copy of their data. This feature is especially useful when working outside the office where the server may not be available.

When a user makes some changes, they are first recorded locally and then they are sent to the server. Fanurio also makes sure the local copy is up to date in case new changes are recorded on the server. The local repository and the server repository are kept identical through synchronization, either manual or automatic.

Manual synchronization can be performed from File » Synchronize, by pressing the F9 shortcut or by clicking the synchronization button from the toolbar.

Manual synchronization is rarely needed because Fanurio can also perform automatic synchronization as follows:

- **On login**

  Fanurio synchronizes the client with the server when a user logs in so that the local repository has the latest changes available on the server.

- **After each local change**

  Fanurio sends local changes to the server as they are made. Instead of synchronizing each individual change immediately, Fanurio waits for 5 seconds to see if multiple changes are made and only then it sends them to the server.

- **Periodically**

  Fanurio synchronizes the client with the server periodically at 2 AM, 8 AM, 2 PM, and 8 PM so that the local repository has the latest changes available on the server.

  For experts: Just in case you want to configure Fanurio to synchronize more often, you need to change the following setting manually. This example shows how to synchronize every hour. When specifying new synchronization patterns, make sure you test them [http://www.cronmaker.com/] first.

  ```
  synchronization.scheduled.pattern=0 0 0/1 * * ?
  ```

If the server is not online or if the synchronization fails for some other reason, the synchronization icon from the toolbar shows a red badge with an exclamation point.

Conflict resolution

Fanurio resolves all synchronization conflicts automatically by using a "server wins" strategy. For instance, if the description of a project changes both locally and on the server, the server description will be replace the local description when the next synchronization is performed.

The best strategy to avoid conflicts is to sync often with the server or make sure Fanurio can sync automatically with the server. That's why Fanurio will show a warning message if you don't sync for more than a day.

5.1.7. Changing the server address in client mode

Servers rarely change their address but if that happens and you're running Fanurio in client mode, go to File » Change Server Address... to update it.

The server address may change for one of the following reasons:

- The IP address of the server has changed.
- The port number has changed.
- The protocol has changed (e.g. http:// to https://).

The server administrator should inform you about such changes and about the new address. If the server has a different address (IP/URL) then you will no longer be able to synchronize your local changes with the server until you update it.

IMPORTANT: If the new address points to a different server (unlikely), the local repository will not be able to synchronize with the server. The new address must point to the same server.

5.1.8. Changing the password

On the server:

- Go to Account » Account Settings to change your password.
- Go to Business » Manage People to change the password of other users.

On the client:

- Click the **Can't access your account?** link on the login screen.
- Enter the email address associated with your account and click Send.
- Check your email for a link to reset your password.

  Note: Passwords can be reset only if the server administrator has configured a notification email account under Server » Server Settings+Email Notification. Password reset emails are sent from this email account.

  Note: Password reset links are valid by default for 15 minutes. As a server administrator, you can change this setting manually. The following example changes it to 30 minutes.

  ```
  server.password.reset.token.lifetime.minutes=30
  ```

5.1.9. Permissions

Fanurio determines what users can and cannot access based on the role assigned to them and the projects they are assigned to.

5.1.9.1. Roles

There are four possible roles a person can have in Fanurio, each with their own level of permissions.

Table 5.1. Roles

Role	Description
Admin	Admins have access to all the data with no restrictions. Some features are not available to Admins when accessing Fanurio in client mode. Admins can perform the following actions only on the server to protect sensitive data like passwords: • managing users and inviting people to create an account (Business » Manage People), • configuring server settings like the SMTP server for email notifications (Server » Server Settings). The user created when setting up Fanurio for the first time has the Admin role by default. Also, the default user created when upgrading from the standalone version to the server version has the Admin role.
Manager	Managers are similar to co-workers with a few exceptions: • they can assign people to their projects (Business » Manage Project Memberships), • they can update the settings of their projects. This role is ideal for people who need to manage a group of other people on their projects.
Co-worker	Co-workers can access only projects assigned to them. Co-workers can see all project records, including those of other users. They can update and delete only their own records if they were not invoiced. Co-workers are not allowed to see or change any billing details. This includes invoices, payments, deposits, taxes, catalog items, products, trip rates and billing settings available on clients, projects, tasks, task categories, time entries, expenses, expense categories, and trips. This role is ideal for members of a team who need to work together on the same project.
Individual	Individuals are similar to co-workers with a few exceptions: • they can't see who else is working on their projects, • they can't create tasks, • they can't see the records (time, expenses or trips) of other users.

Role	Description
	This role is ideal for people who don't need to collaborate with others on their projects.

The following table outlines these roles and what they're allowed to do:

Table 5.2. Roles and permissions

Permission	Admin	Manager	Co-worker	Individual
Billing				
Manage invoices, payments, deposits, taxes, catalog items, products, and trip rates	+			
View billing settings for businss, clients, projects, tasks, task categories, time entries, expenses, expense categories, and trips	+			
People				
Manage people and invitations	+			
View other people (only names, no other details)	+	+	+	+
Update their account (name and password)	+	+	+	+
Business				
Update settings	+			
View currencies (for expenses) and the default distance unit (for trips)	+	+	+	+
Task categories and expense categories				
Manage categories	+			
View categories	+	+	+	+
Clients				
Manage clients	+			
View assigned clients (only names, no other details)	+	+	+	+
Projects				
Manage projects	+			

Permission	Admin	Manager	Co-worker	Individual
Move projects	+			
Assign projects to others	+	+		
Grant and revoke project access	+	+		
View members of assigned projects	+	+	+	
View assigned projects	+	+	+	+
Tasks				
Manage tasks of other users	+			
Manage their own tasks Tasks created by users without billing permissions are saved as billable on the server if they are added to a billable project where tasks are billable by default (Edit Project » Billing). The billing settings of a new task are either the billing settings of its category if the task uses a billable task category or the billing settings of its project.	+	+	+	
View tasks from assigned projects	+	+	+	+
Time, Expenses, Trips				
Manage records of other users	+			
View records of other users	+	+	+	
Manage their own records **Note:** Records created by users without billing permissions are	+	+	+	+

Permission	Admin	Manager	Co-worker	Individual
saved as billable on the server if: • Time entries are added to a billable task. • Expenses are added to a billable project where expenses are billable by default (Edit Project » Billing). • Trips are added to a billable project where trips are billable by default (Edit Project » Billing). The default trip rate is the one used for the most recent billable trip or the one added most recently to the business (Business » My Business Details +Projects).				

Roles with limited access work with a partial copy of the data available on the server. For instance, tasks are downloaded without their billing settings by users who don't have billing permissions. This ensures that sensitive information like billing details or personal information remains protected.

5.1.9.2. Assigning roles

Admins can assign or change the role of other users from Business » Manage People (available only in server mode).

When the role of a user is changed, Fanurio notifies the user about it the first time he or she connects to the server to synchronize their data. When that happens, Fanurio also resets the local repository to download the data the user is allowed to access.

If there are any local changes that haven't been synchronized, Fanurio exports them to disk so they can be reviewed manually. Since reviewing local changes manually can be a laborious process, admins should ask their users to synchronize local changes before changing their role. If a user hasn't synchronized their changes, admins can revert their role temporarily so they can synchronize them.

5.1.9.3. Granting and revoking project access

Admins can grant and revoke project access to other users in two ways:

1. Edit the project (right-click the project and select Edit Project from the contextual menu) then add the person as a member of that project or

2. Go to Business » Manage Project Memberships and double click a person to change the projects assigned to them.

When access to a project is granted or revoked to a user, Fanurio notifies the user about it the first time he or she connects to the server to synchronize their data.

5.1.10. Registering a license key

Fanurio needs a license key to know whether you have licensed the standalone version or the server version. The license key also tells it which updates you are allowed to use. That's why you need to register a new license key each time you renew your annual maintenance plan.

If you have licensed the server version, the license key tells it how many people are allowed to use the application. You need to register the license key only on the server, not on each client.

Here's how to register a license key:

1. Save the license file you received by email (license.dat) on your Desktop.

   IMPORTANT: Don't try to open the license file as there is nothing meaningful in there. Just save it on your computer. Fanurio knows how to read its contents.

2. Go to Help » License Registration to open the License Registration dialog.

3. Click the Enter License... button to select the license key.

   a. Click the Browse button to locate the license file on your Desktop.

   b. Enter the email address where you received the license file in the Email field.

   c. Click OK to confirm the selection.

4. Click OK to register the license key.

5.1.11. Importing data from other applications

Fanurio can import the following data to help you get started faster:

- Clients from a CSV file and from Apple Contacts.

- Time from a CSV file, from iCalendar files and from QuickBooks.

- An iBiz database.

If you need to import something that Fanurio can't handle, please contact us and we'll try to help you get your data into Fanurio.

5.1.12. Installing Fanurio on a USB stick

You can install Fanurio on a USB stick and run it from there on any computer, whether it uses Windows, Linux or Mac OS X. You may want to do this if you work on your client’s computer and you don’t want to install Fanurio there.

The cross-platform version requires Java 6 or higher. If Java is not installed on your computer, you can download the latest version from java.com [http://java.com/en/download/index.jsp].

Here's what you need to do:

1. Download the cross-platform version (.tar.gz)
2. Install it on the USB stick by unzipping it
3. Launch it using:
   - fanuriolocal.applescript on Mac OS X
   - fanuriolocal.exe on Windows
   - fanuriolocal.sh on Linux and other flavors of Unix

Note: It's important to start Fanurio using the local scripts. The other scripts (without local in their name) will start Fanurio but will not keep the data on the stick.

5.1.13. Uninstalling Fanurio

Please follow these steps to uninstall Fanurio from your computer. If you've had problems using it, we would appreciate it if you could tell us more about it.

- **Mac OS X**

  In the Finder sidebar, click Applications. Drag the app from the Applications folder to the Trash (located at the end of the Dock), then choose Finder > Empty Trash.

  If you change your mind, before emptying the Trash, select the app in the Trash, then choose File > Put Back.

  Learn more [https://support.apple.com/kb/PH18752]

- **Linux**

  If you're using Ubuntu, open Ubuntu Software Center and search for fanurio. Then click the Remove button.

  On other Debian-based distributions, open a Terminal window and run the following command:

  ```
  sudo apt-get remove fanurio
  ```

  Learn more [http://askubuntu.com/a/1151/29789]

- **Windows 10**

  1. Open the Start menu.
  2. Click Settings.
  3. Click System on the Settings menu.
  4. Select Apps & features from the left pane.
  5. Select an app you wish to uninstall.
  6. Click the Uninstall button that appears. If it is grayed out, this is a system app you cannot remove.
  7. Click the Uninstall pop-up button to confirm.

  Learn more [http://www.laptopmag.com/articles/uninstall-programs-windows-10]

- **Windows 8.1**

1. Point to the upper-right corner of the screen, move the mouse pointer down, and then click Search.
2. Enter control panel in the search box, and then click Control Panel.
3. Under View by:, select Large Icons, and then click Programs and features.
4. Click the program, and then click Uninstall.
5. Follow the instructions on the screen.

Learn more [http://windows.microsoft.com/en-us/windows-8/uninstall-change-program]

- **Windows 7 and Vista**

  1. Open Programs and Features by clicking the Start button, clicking Control Panel, clicking Programs, and then clicking Programs and Features.
  2. Select a program, and then click Uninstall.

  Learn more [http://windows.microsoft.com/en-us/windows/uninstall-change-program#uninstall-change-program=windows-7]

- **Windows XP**

  1. Click Start, click Control Panel, and then double-click Add or Remove Programs.
  2. In the Currently installed programs box, click the program that you want to remove, and then click Remove.
  3. If you are prompted to confirm the removal of the program, click Yes.

  Learn more [https://support.microsoft.com/en-us/kb/307895]

- **Cross-platform**

  Delete the fanurio folder. Please make sure you backup all data from the local subfolder first.

5.2. User data

5.2.1. Location

The location of data and settings depends on the operating system. Fanurio does this to integrate smoothly with each platform it runs on. As a rule, it stores them somewhere in the user's home folder. Knowing the location can be helpful if you want to backup the settings and data by hand.

By data and settings we mean:

- **repository folder:** This is the folder where Fanurio keeps the database and templates. To change the location of the repository folder, use File » Change Repository Folder....
  - **data subfolder:** This folder contains the user entered data like clients, projects and invoices.
    - **audit file:** This file is used by Fanurio to log database actions so that data can be recovered if something goes wrong with the database or the backup copies. Fanurio keeps at most three audit files. The size of an audit file is no larger than 2 MB.
  - **templates subfolder:** This folder contains subfolders with user-defined templates. You can open these folders by using commands from File » Open Templates Folder.

    Fanurio also has some default templates [files/templates.zip].
- **backup folder:** This folder contains backups made by Fanurio. You can open it from File » Open Backup Folder. To change the location of the backup folder, use File » Change Backup Folder....
- **settings file:** This file keeps application specific settings like the position and size of the Fanurio window.
- **log file:** This file is used by Fanurio to log messages and errors related to its execution. Fanurio keeps at most two log files, depending on the volume of errors and warnings. The size of a log file is no larger than 2 MB.
- **certificates file:** This file stores the SSL certificates trusted by the user. Fanurio prompts users to save invalid certificates (self-signed, untrusted, expired, etc) when connecting to a secure server.
- **user translations folder:** The folder where users can add their (unofficial) translations of the application.

Table 5.3. Location of user data on Windows (XP, 2000)

Item	Location
Log	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\fanurio.log
Settings	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\fanurio.properties
Certificates	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\certs
Default repository folder	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\
Default backup folder	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\backup
Default templates folder	C:\Program Files\Fanurio\share\templates
User translations folder	C:\Documents and Settings\USER_NAME \Application Data\Fanurio\i18n

Table 5.4. Location of user data on Windows (Vista, 7)

Item	Location
Log	C:\Users\USER_NAME\AppData\Roaming\Fanurio\fanurio.log
Settings	C:\Users\USER_NAME\AppData\Roaming\Fanurio\fanurio.properties
Certificates	C:\Users\USER_NAME\AppData\Roaming\Fanurio\certs
Default repository folder	C:\Users\USER_NAME\AppData\Roaming\Fanurio\
Default backup folder	C:\Users\USER_NAME\AppData\Roaming\Fanurio\backup
Default templates folder	C:\Program Files\Fanurio\share\templates
User translations folder	C:\Users\USER_NAME\AppData\Roaming\Fanurio\i18n

Table 5.5. Location of user data on Mac OS X

Item	Location
Log	~/Library/Logs/Fanurio/fanurio.log
Settings	~/Library/Preferences/Fanurio/fanurio.properties
Certificates	~/Library/Preferences/Fanurio/certs
Default repository folder	~/Library/Application Support/Fanurio/
Default backup folder	~/Library/Application Support/Fanurio/backup
Default templates folder	/Applications/Fanurio.app/Contents/Resources/Java/share/templates
User translations folder	~/Library/Application Support/Fanurio/i18n

Table 5.6. Location of user data on Linux

Item	Location
Log	~/.fanurio/fanurio.log
Settings	~/.fanurio/fanurio.properties
Certificates	~/.fanurio/certs
Default repository folder	~/.fanurio/
Default backup folder	~/.fanurio/backup
Default templates folder	./share/templates /usr/share/fanurio/share/templates
User translations folder	~/.fanurio/i18n

5.2.2. About repositories

A repository is a folder on your hard-disk where Fanurio stores data and files. A repository folder contains:

- a data subfolder for its database where the actual data (clients, projects, etc) is saved and
- a templates subfolder where user-defined templates may be installed.

A repository is created when Fanurio is set up for the first time but other repositories may be created from File » New Repository... especially if you need to manage multiple repositories.

Besides creating a new repository, you can also:

- **Open a different repository:** Go to File » Change Repository Folder... to select the new repository.
- **Move the current repository:** Go to File » Change Repository Folder... to select the new location and check the "Copy files to the new folder" box. Fanurio will copy the current repository to the new location and then open it.

Fanurio can manage only one repository at a time. The current repository is opened automatically when you start the application.

5.2.3. About backups

We know data is a very important asset. That's why Fanurio creates backup copies every six hours (1:00, 7:00, 13:00, 19:00) when it is running and another copy when it is started. A backup copy is also created when a new version is installed. Fanurio keeps at most 25 backup copies.

An additional security measure is the audit file that contains the latest database operations. The audit file can be at most 6MB in size and that should be enough to record the data entered during the last week. The audit file can be used to recover data just in case there is no backup copy and the database is corrupted. We don't imagine how both these things could happen and that's why this is an additional measure.

For experts: Just in case you want to change the number of backup copies or the time when Fanurio saves the backup copies, you need to change the following settings manually. This example shows how to create a new backup copy every 5 minutes and keep 20 backup copies. When specifying new autosave patterns, make sure you test them [http://www.cronmaker.com/] first.

```
backup.autosave.running.pattern=0 0/5 * * * ?
backup.copies.number=20
```

5.2.4. Creating and restoring backups

You can manage backups using the following commands:

- Go to File » New Backup... to create a backup.
- Go to File » Restore Backup... to restore an existing backup.
- Go to File » Open Backup Folder to open the backup folder. You can then delete or copy older backups.
- Go to File » Change Backup Folder... to change the location of the backup folder. When you change the location of the backup folder, old backup copies are copied to the new location.

Besides being useful in case something wrong happens with your data, backup copies can also be used to transfer data between two computers. Whether you want to replace your desktop PC with a laptop or you decide to switch from Windows to Mac OS X or Linux, you can transfer your data from the old computer to the new computer using backup copies. Here's what you need to do:

Old Computer

1. Start Fanurio on your old computer
2. Go to File » New Backup... and name the backup migration
3. Go to File » Open Backup Folder to open the backups folder

4. Copy the file called migration.fro to your new computer

New Computer

1. Start Fanurio on your new computer
2. Go to File » Open Backup Folder to open the backups folder on the new computer
3. Copy the migration.fro file to the backups folder on the new computer
4. Go to File » Restore Backup... and choose migration from the list of backups

Once you complete these steps, the data from the old computer will be available on the new computer.

5.2.5. How single users can sync data between multiple computers

If you need to use Fanurio on multiple computers just for yourself, it's very likely you will want to keep your data synchronized. For instance, you may use Fanurio on a desktop PC in your office and on a laptop when you're travelling.

As a single user, you have two options for synchronizing your data, you can either use the server version and let Fanurio synchronize it for you can use the standalone version and let a file-synchronization service like Dropbox do it.

We recommend using the server version because:

- you don't risk losing data as it may happen if you decide to use a file-synchronization service,
- you don't have to worry that your local data is not up-to-date before starting Fanurio,
- you don't have to install Dropbox if you don't need it and
- you can run Fanurio on multiple computers at the same time.

5.2.5.1. How file synchronization works

This section explains how a file-synchronization service like Dropbox synchronizes data across multiple computers and why we're not recommending it for Fanurio.

If Fanurio stores its files in Dropbox then a copy of your database will be available on all computers where Dropbox is installed and running. To keep the data synchronized across all computers, Dropbox detects when a file changes and updates it on all the other computers. It does this by replacing the existing file with the new file.

For instance, if you add time in Fanurio on one computer, Dropbox detects that some database files have changed and uploads them in the cloud. Then the other computers running Dropbox get the new files and save them locally, replacing existing files.

That's why BEFORE running Fanurio on a new computer, you need to make sure that:

- Fanurio is closed on the current computer (File / Exit, Fanurio / Quit, File / Quit) and
- Dropbox is running on the current computer so that all the changes are uploaded in the cloud and that
- Dropbox is running on the new computer so that its Dropbox folder is up-to-date.

If you don't do this, you will be working on an old copy of your data on the new computer and your changes from the current computer will be lost. This also means you can't run Fanurio at the same time on multiple computers because each instance will overwrite the database files of the other instance.

Although we don't recommend this solution it doesn't mean you can't use it. You need to be careful to follow the instructions listed below if you don't want to lose some of your data.

5.2.5.2. Using the server version

If you haven't used Fanurio before, you need to set up the server version. If you're already using the standalone version, you need to upgrade to the server version.

Before using the server version, you may want to read about how Fanurio synchronizes data across multiple computers.

5.2.5.3. Using a file-synchronization service (Dropbox)

IMPORTANT: Before reading this section make sure you read about how a file-synchronization service works and why we're not recommending it.

Here's how you can configure Fanurio to run on two computers. We're assuming you will be using Dropbox but the steps are the same for other services (Google Drive, Microsoft OneDrive).

1. Preparing Dropbox
    a. Make sure Dropbox [http://www.dropbox.com] is installed on all computers where you want to use Fanurio.
    b. In your Dropbox folder, create a folder called Fanurio with two subfolders: repository and backup.
2. Configuring Fanurio on the first computer
    a. Start Fanurio
    b. Change the repository folder from File » Change Repository Folder... to point to the Dropbox repository folder created at step 1.b. Make sure you check the "Copy files to the new folder" box to copy existing files to the new location.
    c. Change the backup folder from File » Change Backup Folder... to point to the Dropbox backup folder created at step 1.b. Make sure you check the "Copy files to the new folder" box to copy existing files to the new location.
    d. Close Fanurio on your first computer as indicated below, don't close it by clicking the window close button (X).
        - Use File » Exit to close the application on Windows
        - Use Fanurio » Quit to close the application on Mac
        - Use File » Quit to close the application on Linux
3. Configuring Fanurio on the second computer
    a. Make sure Dropbox has finished syncing its data so that the repository from the first computer is copied on the second computer.
    b. Start Fanurio
    c. Repeat steps 2.b and 2.c without checking the "Copy files to the new folder" box. There are already some files in there so you don't need to copy them again from the second computer. Now both computers will keep their data in the Dropbox folder.
4. Running Fanurio on either computer (important!)
    a. Make sure Fanurio is not running on the other computer. If it is, close it (see step 2.d.).

b. Make sure Dropbox has finished syncing its data on both computers so that the local repository is up to date.

   If you don't do this, you will be working on an older version that will conflict with the copy saved in Dropbox. Dropbox can't handle conflicts and you will lose some of your data.

c. Start Fanurio

5.2.6. Exporting the log file

If you notice something unusual about Fanurio, you can always send us the log file. You can either locate it on disk or use the Export Log button from the About dialog.

When you export the log from the About dialog, Fanurio creates a file called **fanurio.log.zip** at a location you specify. It's recommended that you export it to Desktop so that you can locate it easier.

5.2.7. Changing settings manually

Warning: Changing settings manually should be done with care and only if it's absolutely necessary. In some situations (very very rare, maybe never), it's the only way Fanurio can be configured.

The settings are different from the actual data (like clients or projects) that users enter in the application. Settings refer to anything Fanurio must remember between successive runs like the position of the application window or the output folder for exported invoices, just to name a few.

Some settings can be changed from the application but others can only be changed by editing the settings file. By doing this, we keep the user interface simple to use and learn with only a few options.

Follow these steps to edit the settings file manually:

1. Make sure Fanurio is not running
   - Use File » Exit to close the application on Windows
   - Use Fanurio » Quit to close the application on Mac
   - Use File » Quit to close the application on Linux
2. Find the file fanurio.properties in:
   - C:\Documents and Settings\USER_NAME\Application Data\Fanurio\ on Windows
   - C:\Users\USER_NAME\AppData\Roaming\Fanurio\ on Vista
   - ~/Library/Preferences/Fanurio/ on Mac OS X
   - *~/.fanurio* on Linux
3. Open it with a text editor
4. Add, edit or remove the lines that contain the settings that you want to edit
5. Save the file and close the editor
6. Restart Fanurio

Example 5.1. Mini-timer size and coordinates

Fanurio uses the **ui.window.mini.bounds** setting to remember the size and location of the iTunes-like mini timer. If you want to reset this setting, you have to follow the above steps and delete the line that starts with **ui.window.mini.bounds**.

5.2.8. Translating to other languages

Fanurio keeps all translations in an archive called **fanurio-i18n.zip**.

You need this even if you create a brand new translation. The file fanurio_i18n.properties is the main file that contains the English translation of the application. The others contain translations of other languages and have a suffix of the form _ll_CC where ll is the language code and CC is the country code.

For instance, fanurio_i18n_fr_CA.properties represents the French translation for Canada while fanurio_i18n_en_CA.properties represents the English translation for Canada.

1. Download fanurio-i18n.zip [files/fanurio-i18n.zip]
2. Extract the contents of the .zip archive somewhere on your computer (e.g. Desktop)
3. Download [http://www.fanuriotimetracking.com/files/prbeditor-0.9.7_2.zip] PRBEditor and start it from your computer

   PRBEditor [http://java.net/projects/prbeditor/] is an application that can help you translate Fanurio easier. Just in case you are wondering what PRB stands for, it means Property Resource Bundle and is a technical term that Java uses when dealing with *.properties files like the ones found in fanurio-i18n.zip.

   A *.properties file can also be edited using a plain text editor but it's a lot more work.
4. Once PRBEditor starts, select **Open resource file** from the **Welcome Form**
5. Locate fanurio_i18n.properties in the folder created at step 2
6. Check the languages that you want to see in the **Select Locales to Open** dialog.

   If you want to translate to a language that's not displayed there, don't select anything.
7. Select Locale > New Locale from the menu to add a new language.

   You don't have to do this if the language you want to translate to already exists. When adding a new locale, make sure you select one that includes both the language and the country, Greek (Greece) for instance and not just Greek.
8. Click on a cell from your language column and type the translation to translate a key.
9. When you're done, go to File > Save. It will save the language file in the folder created at step 2.

While using PRBEditor, you can press F4 to display the Statistics Form. It shows how many keys are translated.

If you choose to share your translation with other users, please send it to us so that we can make it public.

5.2.9. Installing a custom language file

Follow these steps to install a language file or to test a translation file that you created:

1. Locate the user translations folder:
   - C:\Users\USER_NAME\AppData\Roaming\Fanurio\i18n on Windows Vista or Windows 7
   - C:\Documents and Settings\USER_NAME\Application Data\Fanurio\i18n on Windows
   - ~/.fanurio/i18n on Linux

- ~/Library/Application Support/Fanurio/i18n on Mac OS X

2. Copy the language file to the translations folder. It can be the language file created above or any other language file from fanurio-i18n.zip [files/fanurio-i18n.zip].

3. Start Fanurio

4. Go to the Options/Preferences dialog

5. Go to the Locale section

6. Change the language and press Done

7. Restart Fanurio

5.2.10. Password encryption

Fanurio encrypts passwords using your computer's id so that they cannot be decrypted anywhere else. In other words, if someone (our support team, for instance) has access to the configuration file where one or more of your passwords are saved, they will not be able to decrypt them on other computers unless they know your computer's id.

Fanurio decrypts the password and uses it only when it needs to send an email, for instance when emailing invoices. If Fanurio fails to decrypt the password, it will ask you to reenter it. This is not supposed to happen but if it does, you should contact us.

Fanurio uses the MAC address [http://en.wikipedia.org/wiki/MAC_address] to determine your computer's id.

5.3. Known issues

This section contains a list of known issues that may prevent Fanurio from running as expected. Usually, they are quirks of the underlying operating system. Each issue documents the problem and shows a possible solution if there is one.

5.3.1. Printing

If you want to print an invoice from Fanurio, you have two choices:

1. **Directly:** View the invoice then use the Print button.

2. **Indirectly:** Export the invoice to PDF then print it.

Indirect printing (PDF) gives slightly better results but if you are not interested in having perfect alignment for all pixels this shouldn't matter.

5.3.2. Tray icon

Linux Notes

The tray icon is not enabled by default on Linux since the tray implementation is not that good as the Windows one. We believe it's good enough to be part of Fanurio and we provide it because the advantages outweigh the disadvantages.

- If you do not see the tray icon, the notification area is probably disabled. You can enable it by right-clicking the GNOME panel and selecting Add to Panel > Utility > Notification Area.

- Make sure the panel size is >= 26 pixels for the tray icon to align nicely. To change the panel size, right-click on the GNOME panel and select Properties.

- The tray icon background is gray due to a Java bug. To fix this problem, you can change the panel color to match the tray icon background color. Right-click on the GNOME panel, select Properties and change the color to "Solid Color" with approximately 90% opacity.

Windows Notes

The tray icon disappears if Windows Explorer is restarted. This problem is fixed in Java 7 and it only occurs when using Java 6.

You can use Ctrl-Shift-F to bring the main window to front if the tray icon disappears and you can't access it anymore. This shortcut only works if you have Global hotkeys enabled in Tools > Options.

You can also just relaunch Fanurio and the main window of the application will become visible.

5.3.3. Ubuntu time zone

Fanurio (Java) may not read time correctly on certain Ubuntu installations. You can notice this problem if you are trying to add time to a project item using Business » New Time. The "Start" time is different than what your computer says.

If you notice this problem, please use this solution [https://bugs.launchpad.net/ubuntu/+source/sun-java6/+bug/49068/comments/13] to fix it.

5.3.4. Ubuntu 13.10 mini timer

On Ubuntu 13.10 running Unity, the mini timer is larger than it should be. This problem doesn't occur on GNOME 3. Here's what you can do to fix it:

1. Switch from the main window to the mini timer.
2. Click somewhere on the Desktop outside the mini timer.
3. Click the mini timer.

   The mini timer should now be resized to its default size.

5.3.5. White or black window on Windows 7

On Windows 7, the main window is all black or white and the user interface is not visible anymore. This is a known Java bug [http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6429812].

To fix this problem, just copy this file [files/fanurio.l4j.ini] (fanurio.l4j.ini) to the folder where Fanurio is installed (most likely C:\Program Files\Fanurio). If you already have this file, open it with a text editor and add the following line:

```
-Dswing.defaultlaf=com.jgoodies.looks.plastic.PlasticXPLookAndFeel
```

5.3.6. Email servers with untrusted security certificates

Some email servers use untrusted certificates that haven't been verified by a recognized authority. When configuring Fanurio to use such a server, it will not send emails because it only works with trusted servers.

If you know your email server doesn't use trusted identification for a good reason, you can configure your computer to trust it. To do that, please follow these steps:

1. Download the application [http://www.fanuriotimetracking.com/files/installcert.zip] that adds a server's certificate to the list of trusted certificates (this is actually the list of certificates that Java uses)
2. Unzip the archive and double-click runInstallCert to start the application
3. Enter the address of the server (e.g. smtp.gmail.com)
4. Then type the port number (e.g. 465)
5. If the server uses an untrusted certificate, the application will inform you about it and it will display one or more certificates that it uses
6. If you trust any of the listed certificates, type its number and then press enter to add it to the list of trusted certificates
7. Restart Fanurio and try to send an email

Acknowledgments: The application that adds a server's certificate to the list of trusted certificates is a slightly modified version of the application written by Andreas Sterbenz [http://blogs.oracle.com/gc/entry/unable_to_find_valid_certification]. We've changed it to make it easier to specify the server address and port number.

5.4. More help

Whether you want to ask for help, request a feature or simply tell us what you think about Fanurio, you can contact our support team at <support@fanuriotimetracking.com> or using the online contact form [http://www.fanuriotimetracking.com/feedback.php]. You can also subscribe to our blog [http://www.fanuriotimetracking.com/blog], follow us on Twitter, [http://www.twitter.com/fanurio] follow us on LinkedIn [http://www.linkedin.com/company/fanurio-time-tracking/fanurio-886021/product], like us on Facebook [http://www.facebook.com/fanurio] or circle us on Google+ [https://plus.google.com/100175624114354153179/?prsrc=3] to learn what's going on with Fanurio.

We are friendly people who want to help you be more productive.

Chapter 6. Changelog

This chapter contains the changes made in each version of Fanurio. If you want to be notified about new releases, you can always subscribe to the news feed [http://feeds.feedburner.com/fanuriotimetracking].

6.1. Version 4.0-beta3 (December 12, 2017)

Starting with this release, there will be two versions of Fanurio, a standalone version and a server version. The standalone version is designed for freelancers who work on a single computer while the server version is designed for teams and freelancers who work on multiple computers.

See this section to learn how to upgrade from the standalone version to the sever version.

New Features

- **Billable time entries:** You can mark each time entry as billable or non-billable and you can specify the amount of billable time for each time entry.

  We've also added a new user guide section that explains how time billing works.

- **Quantifiable time entries:** Added a new optional field to time entries to track quantities. For instance, translators can track how much time it took them to translate a text but now they can also track how many words they translated during that time.

- **Invoice tax total rounding:** Added a new setting that specifies how invoice tax totals are rounded (half even, half up or half down).

- **Product date:** Added a new field called Date to products. You can use it to track the date when the product was added to the project.

6.2. Version 3.2.1 (December 12, 2017)

This version contains bug fixes and small improvements for version 3.2.

Bug Fixes

- The Pause/Resume Timer action was not initialized correctly. The Resume Timer reminder notification displayed the name Pause Timer instead of Resume Timer.
- Windows: Only English locales where available for selection under Tools » Options » Locale.
- macOS: Some child windows opened behind their parent window. For instance, when editing a time entry in the Edit Task window, the Edit Time window was displayed behind the Edit Task window.
- The report preview area didn't show the horizontal scrollbar if the content didn't fit.
- Improved Dutch translation.

6.3. Version 3.2 (November 2, 2017)

This version contains many new features, improvements and bug fixes. Here's a list with all the changes:

New Features

- **Reports**
  - Redesigned the report windows to show their settings on the left so that the preview area is taller.
  - Can preview reports and invoices created from plain text templates (.txt, .csv, .xml, .iif, and .qif).
  - Redesigned all report templates to use a clean and simple design.
  - New report templates:
    - Time Statistics - new time report that displays various time statistics and **charts**.
    - Sales Statistics - new sales report that displays various sales statistics and **charts**.
    - Tasks Progress by Client and Project - new tasks report that displays the estimated time, the actual recorded time and the progress for the tasks included in the report.
    - Expenses - new expenses report that displays all expenses in a table just like they are displayed in the Expenses view.
    - Expenses by Category - new expenses report that shows expenses grouped by category.
    - Trips - new trips report that displays all trips in a table just like they are displayed in the Trips view.
  - Renamed the following report templates:
    - Project Expenses to Expenses by Client and Project
    - Project Tasks to Tasks by Client and Project
    - Project Trips to Trips by Client and Project
    - Client Summary by Month to Invoices Summary by Client and Month
    - Client Summary by Year to Invoices Summary by Client and Year
    - Client Summary to Invoices Summary by Client
    - Task Summary by Date to Time Summary by Task and Date
    - Task Summary by Week to Time Summary by Task and Week
    - Task Summary by Month to Time Summary by Task and Month
    - Project Timelog by Date to Timelog by Date, Client, Project and Task
    - Project Timelog to Timelog by Client, Project and Task
    - Projects to Projects by Client
  - Added new time, tasks, expenses, trips and sales report placeholders.
  - The ${item.date} invoice placeholder returns a different date for service items.

For service items with at least one time entry, this placeholder returns the date of the oldest time entry while for service items with no time entries it returns the start date of the task it bills. Starting with version 3.0 up until now, the ${item.date} placeholder returned the start date of the task.

If your service items are sorted by date in the invoice template that you are using to export your invoices (most likely) then older invoices may display services in a different order. This will happen especially for tasks that were billed multiple times.

- Added support for CSV templates.

  Most tables can export their data as CSV files but if that doesn't work for you because you want the CSV file in a specific format, you can use a CSV template instead. CSV templates are plain text files that have the .csv extension. The advantage of using CSV templates over plain text templates is that the exported files will have the .csv extension.

  Timelog.csv [files/templates/csv/Timelog.csv] is a template that creates a CSV time report with the Client, Project, Task, Description, and Time columns. Go to File / Open Templates Folder / Time Reports to open the templates folder and copy it there if you want to use it.

- Fanurio can handle inline HTML images represented with the data URI scheme [https://en.wikipedia.org/wiki/Data_URI_scheme]. This new feature is relevant for HTML templates (reports or invoices) that need to embed images.

- Added a new Freemarker directive that can be used to include charts in Freemarker templates (for instance report templates). It can handle area charts, bar charts, line charts, pie charts and ring charts. More chart types will be added in the future.

- Updated Freemarker from version 2.3.20 to version 2.3.23. This update is relevant only to expert users who need to create templates that use features available in versions newer than 2.3.20.

- **Time tracking**

  - Added a new project section that makes it possible to view and manage the time recorded on a project. This section is similar to the Timesheet view except that it shows only the time entries of the project selected in the Projects view.

  - By default, Fanurio tracks time in minutes but it can now be configured to track time in seconds.

  - If you are not recording time in seconds, Fanurio rounds the time recorded by a timer to the nearest minute when it's stopped. Previous versions always rounded time down. Here's how it woks now:

    - If the timer shows **01:29:15** (1 hour, 29 minutes and 15 seconds) then this time will be rounded down to **01:29:00** (1 hour, 29 minutes).

    - If the timer shows **01:29:35** (1 hour, 29 minutes and 35 seconds) then this time will be rounded up to **01:30:00** (1 hour, 30 minutes).

  - The Edit Time and Start Timer dialogs display the name and the price of billable tasks.

  - Time fields are set to 00:00 if their content is deleted.

  - The duration of the Time and Pause fields from the New Time dialog can be adjusted using the up and down arrow keys.

  - Deleting time entries from Edit Task | Time now asks for confirmation.

  - The Timesheet contextual menu that's displayed when a time entry is right-clicked or Ctrl-clicked on macOS contains two new actions: Start New Timer and Start New Timer... . They start the timer for the same task and with the same description.

- The default date for new time entries depends on the selected view and its date filter. When the Timesheet view is selected and its date filter is set to a specific date or period like yesterday, the default date for new time entries will be yesterday instead of today. This feature makes it easy to enter records in the past and it was first implemented in version 2.2.

  The New Time action has been improved to use the same default date no matter where it is called from (menu, toolbar, table button, table double-click, or contextual menu). This behavior was also improved for expenses and trips.

- The default date for the first time entry of a task is the start date of the task.
- The delimiter field from Import | Import Timesheet (CSV) is now editable.
- Added a new setting (ui.window.mini.alwaysontop) that can change whether the mini-timer window is always on top or not.

  This setting is enabled by default but if you want to disable it, you must edit the settings file manually and add the following line at step #4:

  ```
  ui.window.mini.alwaysontop=false
  ```

- **Projects**
  - Projects can now be moved from one client to another. To do this, right-click a project in the Projects view, select Edit Project from the contextual menu and then set the new client in the Edit Project window.

    Invoiced projects can't be moved since they can't be billed to a client and then be assigned to another client but uninvoiced tasks, expenses, trips and products can be moved to other projects.

  - When trying to edit an invoiced project element (expense, trip, product or time entry), Fanurio will inform you that it can't be edited but it will now offer you the option to view it.
  - Right-click a trip in the trips table and select **New Reverse Trip** from the contextual menu to create a copy of the selected trip that reverses the start and end locations.
  - The due date for new tasks defaults to the due date of their project if the optional Due Date field is enabled for tasks.
- **Contacts**
  - Go to File » Export » Export Clients to export all the clients you recorded in Fanurio to CSV or Excel.
  - Added a search field to File » Import » Import Contacts from Apple Contacts ... to locate contacts faster when the list of contacts is very large.

Improvements

- **Database loading time:** The database is loaded faster now. For large databases (data recorded over several years with many projects and invoices), the loading time is reduced to more than half of what it used to be. This means that the application will start a few seconds faster.
- **Table improvements**
  - Added a new table shortcut to edit the selected record (Command-I on OS X and Alt-Enter on Windows and Linux). Tables already had a shortcut for deleting the selected record (Delete).

    To avoid confusion between the new Command-I shortcut and the existing "Switch to Invoices View" shortcut (Command-Shift-I), the latter now uses Command-Shift-V (OS X) and Control-Shift-V (Windows, Linux).

- Double-clicking the empty area of a table will create a new record for that table. For instance, double-clicking the empty area of the Timesheet table will create a new time entry.
- The totals displayed at the bottom of a table are calculated for all records when one row is selected and for all the selected records when two or more records are selected.
- Added more space between the totals displayed at the bottom of a table to make them easier to read.
- Added a new Select Columns action to the table header popup menu that makes it easier to select the visible columns.
- Removed all table buttons except for the New button to declutter the area below each table. The same actions are available from the contextual menu of each table. Ctrl-click on macOS and right-click on Windows and Linux to open the contextual menu.

- **Date filters**
  - Date filters have two new buttons that allow you to navigate to the next or previous period. The buttons are enabled only if a day, week, month or year is selected, they are not enabled if an arbitrary period is selected.
  - Date filters across the application use the same grouping of predefined periods, the current period (Today, This Week, This Month, This Year) and the previous period (Yesterday, Last Week, Last Month, Last Year).
  - Report date filters have two new periods when billing is enabled, This Financial Year and Last Financial Year.
  - Redesigned the looks of the date picker calendar and added two new buttons to scroll its year.
  - The date filter popup has a new action that can select a month called Select Month. Besides the standard periods (Today, This Week, This Month, etc) you can now select specific days, months or custom periods.
- **Duplicates**
  - Renamed all "Copy X" actions to "Duplicate X" to avoid confusion on whether the selected object is copied to clipboard (it isn't). For instance, Copy Project is now called Duplicate Project while Copy Expense is called Duplicate Expense.
  - The folowing fields were not copied when creating a duplicate object: Task.notes, Project.description and Project.notes.
- **Projects table**
  - The Projects table has a new Invoices column that displays the invoices for each project. The column is hidden by default, right-click any column name to make it visible.
  - The Number and Reference columns from the Projects table are left-aligned since they can contain any values, not just numbers.
  - Added a new setting (ui.projects.largerows) that can change whether the projects table has large rows or not.

    This setting is enabled by default but if you want to disable it, you must edit the settings file manually and add the following line at step #4:

    ```
    ui.projects.largerows=false
    ```
- **Invoices**

- The Invoices table has a new Projects column that displays the invoiced projects. The column is hidden by default, right-click any column name to make it visible.
- The tasks tables have a new Invoices column that displays the invoices for invoiced tasks. The column is hidden by default, right-click any column name to make it visible.
- Added the action New Product (Ctrl-U, Command-U) to the Business menu.
- Added the action View Invoice to the payments contextual menu.
- The default currency used for foreign clients is the client currency.

  For instance, if your business is based in Canada and you're billing in both CAD and USD, invoices created for US clients (must be marked as foreign clients) will use USD as their default currency.
- Improved Email Invoice to ask for the email password if it can't be decrypted instead of displaying the "javax.crypto.BadPaddingException: Given final block not properly padded" error message.
- New tasks billed in units, products and regular invoice items are checked whether they have a zero quantity.

- **Default invoice template**
  - The template editor can configure a template to show two more totals: the client balance and the balance of all other invoices.
  - The default invoice template and invoice templates created by the Template Editor show totals only on the last page if the invoice has multiple pages.
  - The default invoice template doesn't display anything in the Time column for invoice items with no time.
  - The default invoice template doesn't display a Time subtotal for invoices with no time.
- **User interface**
  - Improved the layout of several dialogs: Select Project, Select Projects, Configure Optional Fields, and Global Hotkeys.
  - Search fields can now handle multiple tags. The tags must be separated by commas and matching objects must have all the specified tags.
  - Improved the Edit Tags dialog to handle a long list of tags.
  - Most dropdown lists display more elements. For instance, the category dropdown from the New Task dialog shows 25 elements. This makes it easier to select a category if you have lots of them.
  - Repository-related actions can be accessed from the File menu when the repository is closed.
  - Improved French, German and Italian translations.

    Special thanks to Jan-Christoph Ihrens from Comprehensive Computer Services [http://www.cc-services.de] for the updated German translation.
- **Java-free Windows version:** The Windows version no longer requires Java in order to run. You can uninstall Java if you installed it to run Fanurio.

Bug Fixes

- iBiz import - Failed to import job events without names. If a job event doesn't have a name, Fanurio will use the name Noname when it imports it.
- Windows 10: Fanurio wasn't using the Windows task bar features (thumbnails, pinning, badge icons) and the right user interface look and feel.
- Fanurio prevented Windows from shutting down if it was still running.
- The date filters "This Financial Year" and "Last Financial Year" were not initialized properly.
- Windows are moved to the primary monitor if they are displayed in a secondary monitor that's no longer enabled.
- When clicking the Start New Timer button, the timer was not started for the selected task but for the last task that was right-clicked.
- Trips were not sorted by date in the *Trips by Client and Project* template.

6.4. Version 3.1.2 (July 12, 2016)

This version contains bug fixes and small improvements for version 3.1.1.

Bug Fixes

- Failed to exit on systems without a tray bar (Fedora Linux with openjdk 1.8.0).
- Time was not filtered correctly by project reports. The Invoiced and Billable filters didn't apply to time entries, they only applied to tasks, expenses, trips and products.
- The Task Summary by Week time report failed to calculate the week number for the last week of the year.
- The week filters from the New Invoice | Add Project Items dialog always used Monday as the first day of week.
- Failed to close on certain Linux systems (e.g. Ubuntu) if Fanurio was running when the computer was shut down.
- On Linux systems that don't have the DBus GNONME Session Manager, if something was recorded in Fanurio and then the OS was shut down immediately (40 seconds or less) without closing Fanurio first (File | Quit), those changes were lost (not saved to the database).
- The .deb installer failed to run on Linux systems that don't have openjdk-6-jre. It now works on all systems with openjdk-6-jre, openjdk-7-jre, openjdk-8-jre or openjdk-9-jre.
- The projects tree tried to select new or updated clients and projects that were not visible.
- Fixed the time formats used to import time from CSV. The Start field can match time in the 12-hour format (hh:mm a, hh:mm:ss a) and the 24-hour format (HH:mm, HH:mm:ss).

6.5. Version 3.1.1 (September 7, 2015)

This version contains bug fixes and small improvements for version 3.1.

Improvements

- Enabled retina font support on OS X.
- Improved the user interface to look better on HiDPI monitors (Windows and OS X). Most fonts, components, and layouts scale correctly now.

  This version doesn't have HiDPI icons and it will not work in HiDPI mode on Linux. These improvements are scheduled for another release.

Bug Fixes

- iBiz import - Failed to convert some invoices that contained job event groups.
- The time recorded by the timer was not always rounded down to the nearest minute. If the time entry dialog was configured to enter time relatively to start and end (both), the time recorded by the timer included seconds as well. For instance a timer session of 00:06:36 hours was not saved as 6 minutes (0.10 hours) but as 6 minutes and 36 seconds (0.11 hours).
- Time-related totals were not updated in the Tasks section of a project if a time entry was edited in the Edit Task dialog.
- Text fields from the New Invoice window collapsed when the window was not wide enough.
- Catalog items were not removed if deleted from Business » My Business Details.
- The first day of week displayed by the date range picker didn't depend upon the selected region.
- The paid flag was not updated when invoices were edited.

6.6. Version 3.1 (March 11, 2015)

This version contains new many features, improvements and bug fixes. Here's a list with all the changes:

New Features

- **New setup guide:** A setup guide is displayed automatically when Fanurio is started for the first time. It helps new users create a new repository and configure their business.

  See this section for more details.

- **Refreshed user interface**

  - **Views bar:** The new views bar is located below the toolbar and it allows you to switch between views. It replaces the toolbar dropdown box and the sidebar to provide only one method of changing the selected view. The place occupied by the toolbar dropdown box is now taken by a new button that closes the application.

    In previous versions, the sidebar allowed you to configure which views were visible but now you can disable the features that you don't use (e.g. billing, expenses, trips or products) and their associated views will be hidden.

  - **Lighter user interface:** The status bar is now white instead of dark gray while the divider lines and the filters areas use a lighter gray than before. Also, all tables have slightly taller rows so that text is more readable.

- **Database-related enhancements**

  - **Database loading time:** The database is loaded faster now. This change should be visible especially to users with large databases (data recorded over several years with many projects and invoices). In some cases, the database loading time will be reduced to a quarter of what it used to take.

  - **Newer versions of the database:** The application can now detect whether the database has a newer version than what it can handle. If the database version is newer then it will not be opened.

    This situation can happen if you try to restore a backup created with a newer version of the application or if you share the database between multiple computers using Dropbox but not all the computers run the same version of the application.

  - **Manage repositories:** In previous versions it was possible to configure the location of the database and the user-defined templates folder independently of each other but to make things easier, we decided to group them under a single folder called repository folder.

    The location of the repository folder can now be changed from File » Change Repository Folder... while the location of the backup folder can be changed from File » Change Backup Folder....

    See this section for more details about repositories.

- **Disable optional modules (billing, expenses, trips, products)**

  Starting with version 3.0, Fanurio is built around two main modules: **billing** and **projects**. These modules can be used separately from each other or they can be used together.

  - **just billing, no projects:** create regular (non-project) invoices

  - **just projects, no billing:** work on non-billable projects

  - **both billing and projects:** work on billable projects and create invoices for them

Version 3.0 doesn't allow you to specify whether a module is active or not, both modules are active all the time. This can be a problem if don't need to use both modules. For instance, if you don't need to bill your clients, you should be able to just turn billing off and never see anything related to billing like prices or invoices in the user interface. This version allows you to disable billing and certain project features such as expenses and trips that are not used frequently. Here's what has changed:

- **Business settings:** Reorganized the tabs from Business » My Business Details so that the Projects tab contains all the project-related settings while the Billing tab contains all the billing-related settings.

  The Tasks, Expenses and Trips tabs are now sections in the Projects tab while Catalog and Taxes are sections in the Billing tab.

- **Non-billable clients:** In previous versions, all clients were billable. Now you can mark them as billable or non-billable. All the projects of a non-billable client are non-billable.

  You can now mark clients, projects, tasks, expenses and trips as non-billable.

- **Disable billing:** If you don't need to bill your clients then you can disable billing at business level. Go to Business » My Business Details+Billing to do this. Please note that you can't disable billing if you have at least one invoice or if you have projects with at least one product.

  When billing is disabled at business level, the following things happen:

  - The Invoices and Payments views are removed.
  - The menu and toolbar actions related to invoices and payments are removed.
  - All billing columns are removed from the projects, tasks, expenses and trips tables.
  - All billing filters are removed from all views.
  - All clients, projects, tasks, expenses and trips are changed to non-billable.
  - Projects can no longer manage products.

- **Disable expenses, trips and products:** Just as you can disable billing, you can also disable expenses, trips and products. Go to Business » My Business Details+Projects to do this. Products can be disabled only if billing is enabled because products can only be used in billable projects.

  When expenses, trips or products are disabled at business level, the user interface is updated to remove all references to them.

- **Search, sort and filter projects by various fields**

In previous versions, it was difficult to manage the list of projects if you had many of them because Fanurio could record only a few details about a project (name, description, number, reference and status).

When you manage many projects, it's important to be able to search, sort and filter them by various fields. That's why the following new features are meant to make this task easier.

- **New projects table view:** The tree view displays projects grouped by client and shows only the name and the status of a project (finished or not finished). It can't be searched or sorted and it can't display detailed information about a project. To fix these problems, we added a new projects view.

  The new table view shows projects in a table that can display as many project fields as you need (right-click the table header to choose the visible columns). It also allows you to sort and search projects by any field. For instance, you can choose to see the total time recorded on a project or its total value.

The table view is not visible by default but it can be enabled from the View menu using View » View Projects as Table.

- **New project fields:** The list of projects can be managed easier now because we added several new fields.
  - **Start date:** You can now track when a project was started. This allows you to go back and see past projects, for instance projects you did last year.

    For old projects, the start date is calculated automatically. Fanurio uses the first date when something was recorded on a project to determine its start date. It compares the date of the first task, the date of the first expense, the date of the first trip and the date when the project was created to determine the start date.
  - **Finished date:** Besides being able to mark a project as finished, you can now set the date when it was finished. Finished projects are displayed in gray in the projects table.

    For old finished projects, the finished date is calculated automatically. Fanurio uses the last date when something was recorded on a finished project to determine its finished date. It compares the completed date of the last task, the date of the last expense and the date of the last trip to determine the finished date.
  - **Due date:** You can also specify a due date for your projects. This helps you prioritize projects by due date but also see overdue projects. Overdue projects are displayed in red in the projects table.
  - **Tags:** Use tags to record the project type, status or anything else that can help you manage projects easier. See this section for more details.
- **Optional project fields:** The number, reference, due date and tags fields are optional and can be hidden when creating or editing projects. Only the Number field is visible by default.

  See this section for more details.
- **Automatic project numbering:** Project numbers can be now generated automatically. This feature allows you to generate a unique number for your projects so you can find them easier. To enable this feature, go to Business » My Business Details+Projects.

  By default, Fanurio generates numbers using four digits but you can define your own format.
- **New clients filter (show only clients with unfinished projects):** The clients tree (projects tree view) and the clients list (projects table view) have a new filter that only shows clients with unfinished projects. This feature can help you hide active clients that don't have any ongoing projects. Up until now, the only solution to hide such clients was to mark them as inactive.

  To use this filter, click the small gears button and select "Active with Unfinished projects".
- **Complete tasks when a project is finished:** If a project is marked as finished and it has tasks that haven't been completed, Fanurio will prompt you to decide whether these tasks should be marked as completed or not.

- **Billing enhancements**
  - **Exact precision for billable time:** Up until this version, Fanurio used **two-decimal precision** to calculate billable time but now it can also use **exact precision** (unlimited number of decimals).

    For instance, if a billable task was billed at a rate of $60/hour and it had 20 minutes of work then the task was evaluated at $19.80 (0.33 hours) instead of $20.00 (0.333...3 hours).

The precision method is configurable and is set to exact for new repositories by default. Old repositories will be migrated to use two-decimal precision unless Fanurio was configured to use the undocummented "decimal.time.precision" system property in which case old repositories will use exact precision.

See this section for more details about the two methods of precision.

- **Deposits:** Billable clients have a deposits account that can track money paid in advance. You can use money from this account to pay client invoices. See this section for more details.

  Also added new template placeholders (client.deposits and client.depositsBalance) to allow you to access deposits-related information from invoice templates. See the templates placeholders section for more details.

- **Default client billing settings:** Clients have new billing settings for their tasks. These settings are used when new projects are created to set their default settings for billing tasks.

  For instance you can specify a default hourly rate and a default rule for rounding time for each client. Each time a new project is created, it will have the default rate and rounding specified at client level.

  See this section for more details.

- **Financial year:** Added a new setting for the start date of the financial year under Business » My Business Details+Billing. This makes it possible to select the current or the last financial year when creating sales reports (Reports » Sales Report) or when filtering invoices and payments.

  The default start date for the financial year is January 1st. If your financial year starts on a different date, you should change it from Business » My Business Details+Billing.

- **Task category billing:** In the previous version, when a billable task used a billable category, the pricing of the task couldn't be changed. It was fixed to whatever the category had, if the task category was billed in hours so was the task.

  The billing settings of a task are no longer limited to the billing settings of its category. You can now define a category that's billed in units and use it with tasks that are billed in hours.

- **Optional invoice fields:** The attention, reference (used to record the purchase order reference number) and period fields are optional and can be hidden when creating or editing invoices. All these fields are now hidden by default. See this section for more details.

- **Reports**

  - **New reports:** Added new reports for projects, expenses and trips.

  - **New Time Report filters:** The Time Report has two new filters that allow you to filter time entries by their billable and invoiced status. These filters are visible only if billing is enabled.

  - **New Tasks Report filters:** The Tasks Report has two new filters that allow you to filter tasks by their billable and invoiced status. These filters are visible only if billing is enabled.

  - **New Tasks Report placeholders:** Added two new placeholders that calculate the estimated and the remaining time for a list of tasks: tasksreport.calculateEstimatedTimeAsHour, tasksreport.calculateRemainingTimeAsHour.

  - **Non-HTML report templates:** You can now use non-HTML templates (eg Microsoft Word, OpenDocument Text, etc) to export reports but not to view them. Only HTML templates can be used to view reports in Fanurio.

- **Rounded elapsed time for time entries:** If a time entry belongs to a billable task or to a service item that rounds time, the rounded time may be different from the actual recorded time. This property is useful in invoices and in time reports where you need to display the rounded (billable) time and not the actual time.

  For instance, if a time entry has 12 minutes of recorded time and it belongs to a billable task that rounds time up to 15 minutes then the rounded elapsed time for this time entry is 15 minutes.

  Fanurio has two additional placeholders that allow you to access this property from invoice templates and time report templates:

  - timeEntry.roundedElapsedTimeAsHour
  - timeEntry.roundedElapsedTimeAsDecimal

  and another two placeholders that can be used in time report templates to calculate the total rounded elapsed time for a list of time entries.

  - timereport.calculateRoundedElapsedTimeAsHour
  - timereport.calculateRoundedElapsedTimeAsDecimal

  Templates created by the invoice template editor (File » Template Editor) now display the rounded time instead of the actual time.

- **Renamed placeholders that indicate invoiced time**
  - invoice.billedTimeAsDecimal replaces invoice.billableTimeAsDecimal
  - invoice.billedTimeAsHour replaces invoice.billableTimeAsHour
  - project.billedTimeAsDecimal replaces project.billableTimeAsDecimal
  - project.billedTimeAsHour replaces project.billableTimeAsHour
  - item.billedTimeAsDecimal replaces item.billableTimeAsDecimal
  - item.billedTimeAsHour replaces item.billableTimeAsHour
- **New time reporting placeholders for invoices:** Added new invoice placeholders for invoice templates that need to group time entries by date.

- **Integration with other applications**
  - **Apple Contacts**
    - Replaced the name Address Book with Apple Contacts in all actions. Address Book was the name of the application before OS X 10.8 Mountain Lion.

      If you're using an older version of OS X, the following actions refer to Address Book: Import from Apple Contacts, Update from Apple Contacts, Show Contact in Apple Contacts and Edit Contact in Apple Contacts.

    - You can now update the contact details of a client imported from Apple Contacts by ctrl-clicking it and selecting Update from Apple Contacts from the contextual menu.

      Before version 3.1 Fanurio didn't associate a client with a contact. That's why the first time you'll be updating a client imported with an older version, Fanurio will ask you to associate it with a contact from Apple Contacts. You only need to do this once for each client you imported from Apple Contacts in the past.

    - You can now access and edit the contact associated with a client by ctrl-clicking it and selecting Show Contact in Apple Contacts and Edit Contact in Apple Contacts from the contextual menu.
    - Improved Apple Contacts field mapping to also collect data from home fields if work fields are empty.
- **iCalendar-compatible applications**
    - **Export projects in the iCalendar format:** Fanurio can export projects in the iCalendar format so you can use them in popular calendar applications like Google Calendar, Apple Calendar and Reminders (formerly iCal), Microsoft Outlook, Yahoo! Calendar or the Lightning extension for Mozilla Thunderbird.

      See this section for more details.
    - **Import time from iCalendar:** Fanurio can import events from popular calendar applications as time entries. Fanurio is compatible with any calendar application that can export its events as an iCalendar file (eg Google Calendar, Apple Calendar (formerly iCal), Yahoo! Calendar or the Lightning extension for Mozilla Thunderbird).

      See this section for more details.
- **Migrate from iBiz:** IGG Software have announced that they are officially ceasing the development of iBiz, their time tracking and billing application. To help iBiz users move to a new solution and keep their old iBiz data, we've created an import module that converts an iBiz (4.1.4 and older) database to a Fanurio repository.

  See this guide to learn how to migrate from iBiz to Fanurio and how Fanurio is different from iBiz.

- **Runs on Java 6 and later:** Fanurio now requires Java 6 or later to run. As a result, Mac OS X 10.4 is no longer supported but all other platforms (Mac OS X 10.5+, Windows or Linux) are supported.

  See this section for more details on how to install the latest version of Java on your computer.

Improvements

- **Mileage items:** In the previous version, a mileage item could group trips with the same rate from multiple projects. Now, it can only bill trips with the same rate from the same project.

  Mileage items are project items so it makes sense to bill only trips from the same project.

- **Use Start TLS for SMTP servers:** If you use Fanurio to send invoices by email, you can now configure the SMTP server to secure the connection with Start TLS.

  The "Edit Outgoing Mail Server (SMTP)" window has a list of settings for popular email services like Comcast, Gmail, iCloud, Outlook.com and Yahoo! Mail.

- **Contextual menus:** Added contextual menus to all major tables and lists. Up until now, only clients and projects from the Projects view had contextual menus but now all tables from all views have one.

- **Time input:** Time can be entered easier in the New Time dialog (Time, Pause) and in the New Task dialog (Estimated Time). The two spinners have been replaced with a text field that accepts time in both hour and decimal format. Here are a few input examples:
    - 2:30 - enter the number of hours and minutes.
    - :30 - enter the number of minutes.
    - 2.5 or 2,5 - enter the number of hours.
    - .5 or ,5 - enter the number of hours.

- 2 - enter the number of hours.

A time input field also has two links (hh and mm) that allow you to enter time using the mouse.

- **More columns**
  - The project tasks table has two new columns: Description and Notes. These columns are not visible by default.
  - The project tasks table and the tasks view table have two new columns: Billable Time and Billed Time. These columns are not visible by default. When visible, their totals are displayed at the bottom of the table.
  - The tasks table from the New Invoice > Add Project Items window has configurable columns now. It also has a new column for Reference.
  - The Completed Tasks table from the New Invoice > Add Project Items window has configurable columns now. It also has a new column for Reference.
  - The timesheet table has a new column for task reference. This column is not visible by default.
  - The invoices table has a new column for invoice profit. The invoice profit is calculated as the difference between the invoice total and the total amount of all invoice expenses. This column is not visible by default.
- **User interface**
  - The task window has a new field that displays the total time recorded on the task. Next to that field is the 'New Time' link that allows users to add time to the task without having to switch to the Time tab.
  - The Period drop-down from the New Invoice > Add Project Items window has two new periods: Last Two Weeks and Last Half-Month [http://en.wikipedia.org/wiki/Half-month].
  - Improved editing for numeric fields including money fields.
  - Most date pickers are now editable.
  - Date columns are aligned to the left instead of the right.
  - The tags field is now optional and is hidden by default when time entries are created or edited. See this section for more details.
  - The clients tree remembers the expanded client nodes when the application is restarted and when the filters (Any, Active, Not Active Clients and Any, Finished, Not Finished Projects) change.
  - Actions from File » Export now export all records, for instance File » Export » Export Expenses... exports all expenses recorded in the application. To export only the records visible in the table from the Expenses view, use the new Export Table button below that table.
  - Improved the New Invoice window to display totals at the bottom of the items table. The Notes field was moved to a separate tab.
  - Using a standard file chooser to open or save a backup copy. You can now save and open backups from anywhere, not just from the backup folder.
  - Using a standard file chooser to export invoices and payments.
  - The **Start New Timer...** dialog allows you to specify the start time relatively in minutes (e.g. 13 minutes ago).

- The timer reminders are now focused on the action that they should trigger. For instance, the Start New Timer reminder allows you to specify when the timer should be started (now or in the past).
- The template editor can now specify the character encoding so that templates don't have to be changed manually. This setting is useful to people who use custom TrueType (*.ttf) or OpenType (*.otf) fonts that don't render correctly when the document is exported to PDF.
- Clients have a new tab that groups company registration fields: business number, tax number, other number and code.
- When exporting invoices, you can tell Fanurio to show you the file on disk (in Windows Explorer on Windows or in Finder on OS X).
- The user interface displays consistently when a high contrast theme is used on Windows.
- Improved Spanish translation.

Bug Fixes

- The width was not always saved for hidden table columns that were made visible and then resized.
- On OS X, the application can be installed in any folder, not just in Applications. If installed in subfolders that had a space in their name, the default templates and the user guide were not found.
- Invoiced filters also showed non-billable elements. For instance, if both billable and non-billable expenses were recorded, when the Not Invoiced filter was selected in the Expenses view, non-billable expenses where still visible.
- If a client had two or more projects with the same name and one of them was clicked, the first project was selected instead of the one that was clicked.
- Failed to delete clients with invoices.

6.7. Version 3.0.1 (March 31, 2014)

This version contains bug fixes and small improvements for version 3.0.

Improvements

- **"Not Fully Invoiced" shows new billable tasks:** Changed the "Not Fully Invoiced" tasks filter to also include new billable tasks with no billable quantity. If this filter is set and you create a new billable task, the task will be visible in the tasks table now.

  The idea behind this change is that even though a new billable task has no billable quantity, it will probably have otherwise it doesn't make sense to make it billable in the first place.

- **New Task for the selected project:** The New Task action (from the menu or from the toolbar) creates a task for the selected project if the Projects view is active.

- **Firewall warning on OS X:** When launching Fanurio on OS X, it shows a warning window with the following question:

Do you want the application "Fanurio.app" to accept incoming network connections?

This warning is triggered by the code that checks whether multiple instances of Fanurio are launched. This code is no longer used now on OS X when launching Fanurio.app but it is used if you launch Fanurio using one of the .applescript scripts from the cross-platform distribution (.tar.gz).

We are not making this check on OS X any longer because no matter how many times you are launching Fanurio.app, OS X will not launch a second instance unless you are a power user who opens a Terminal window and uses "open -n". More details about this special case here [http://reviews.cnet.com/8301-13727_7-57564478-263/how-to-open-multiple-instances-of-an-application-in-os-x/].

Bug Fixes

- Failed to import contacts from Contacts on OS X 10.9 (Mavericks).
- Missing invoice.mileageItems and invoice.mileageItemsSubtotal placeholders.
- Failed to set a filter configuration for tasks.
- Failed to upgrade to 3.0 when an item and its catalog item were of different types (e.g. product item that used an expense catalog item).
- Fixed a memory leak that occured when a project was opened and then closed.

- Windows 8.1: Fanurio wasn't using the Windows task bar features (thumbnails, pinning, badge icons) and the right user interface look and feel.
- New Task fields are reset when selecting a project for the first time.
- Updates the unit of measure for tasks and projects when Pricing changes.
- Saves time entries that are edited when an invoice is created.
- Time is not rounded correctly for service items that have time entries with no time.
- When exporting time entries, expenses, trips and payments to CSV or Microsoft Excel, the Invoice column contained an internal identifier instead of the actual invoice number.
- Failed to update the time total from the Tasks view when time was added, removed or updated.
- The cursor jumped to the next field when entering time using the "enter time relative to finish" option.
- Distance was not sorted correctly.

6.8. Version 3.0 (December 3, 2013)

This version changes the way projects are organized so that they can be used for more than just billing. Projects manage four lists of different elements (tasks, expenses, trips, and products) instead of just one (items). Each element can record specific information, for instance tasks can have due dates while trips can record distances. Items were not capable of this because they were designed to handle only billing.

As a result of this change, new features like date range invoices or non-billable projects are now possible. Here's a list with all the changes:

New Features

- **Introducing tasks:** Tasks replace service items at project level. In fact, tasks extend service items to provide more functionality and to make billing easier. Here's what's different:

  - Unlike service items, tasks can be billed more than once. You don't have to create a service item for each billing period, you only have to create one task and then bill it multiple times. This is one of the main reasons we decided to replace service items with tasks. If you don't need to bill a task multiple times then you don't need to worry, it works just like before.

    Since tasks can be billed multiple times, they have a billing history that shows when they were invoiced and how much time or quantity was invoiced. The tasks tables have multiple columns for quantity (billed, billable) and total (billed, billable). Not all of them are visible by default.

    Although service items were replaced by tasks at project level, service items are still used at invoice level. Fanurio creates a service item for a task whenever it is billed. In other words, tasks are used to manage work at project level while service items are used to bill work recorded on tasks. Service items are created automatically by Fanurio when a task is billed in the New Invoice > Add Project Items window.

    By default, service items use the billing settings of their tasks but a service item can have different billing settings. For instance, you can bill some time entries at a certain rate while others at a different rate (eg overtime rate). You can edit a service item in the New Invoice window.

  - Task planning should now be easier because we added a few more fields.

    - The Date field was renamed to Started Date.

    - The new Estimated Time field can help you track the remaining time for a task. Fanurio calculates the remaining time by subtracting the time recorded on a task from the estimated time.

    - Tasks can be marked as completed. For completed tasks, you can also specify the date when they were completed. You can also configure Fanurio to mark tasks as completed when they are invoiced.

    - The Due Date field can help you schedule the completion date for a task. You can also use task filters to see overdue tasks.

  - The new Tags field can help you organize tasks better.

  - Task categories can be managed from Business » My Business Details+Tasks.

  - Projects have a new view only for tasks. Tasks are no longer managed together with the other project items.

  - The application has a new Tasks view. You can use it to manage all the tasks you record for your projects. The Tasks view works just like the other views (Timesheet, Invoices, Payments), you

can filter tasks by different properties (for instance you can see overdue tasks) and you can export them to CSV or Microsoft Excel (File » Export » Export Tasks).

You can find more details about tasks in their own section. For more details on how old project service items are migrated to tasks, see the Migration section below.

- **Introducing trips:** The new trips module allows you to record the distance and time you travel with a vehicle whether it's for billing or for tax purposes.

  Up until now the only way to bill mileage was to use expense items but the new trips module allows you to keep accurate records of your journeys whether you need to bill them or not. Distances can be tracked in miles, kilometers or both. You can also define multiple rates for billable trips.

  We've also added new placeholders to let you access trip information from invoice templates. You can access the list of mileage items that bill trips using **invoice.mileageItems** or **project.mileageItems** while the trips of an item can be accessed using **item.trips**. The invoice template editor has been updated to customize the way trips are displayed. See the templates placeholders section to learn about the placeholders introduced in this release.

  You can find more details about trips in their own section.

- **Redesigned expenses:** We've redesigned expenses to allow you to record more details about the money that you spend. Here's what's different:

  - The new Tags and Reference fields can help you organize expenses better.

  - Expenses now record the total amount and not the price of individual items.

    This means that you can no longer say things like "this expense is for two tablets, each costing $500" instead you will record it as "this expense is for two tablets that cost $1000". Notice that the amount is for the whole expense. This also means that whenever you will bill an expense, the billable quantity will always be 1. If you need to bill individual items (quantity is different from 1) then you need to use products.

  - Expense categories can be managed from Business » My Business Details+Expenses.

  - Expenses can now be billed with a markup.

  - Projects have a new view only for expenses. Expenses are no longer managed together with the other project items.

  - The application has a new Expenses view. You can use it to manage all the expenses you record for your projects. The Expenses view works just like the other views (Timesheet, Invoices, Payments), you can filter expenses by different properties (for instance you can see only non-billable expenses) and you can export them to CSV or Microsoft Excel (File » Export » Export Expenses).

  You can find more details about expenses in their own section. For more details on how old expenses are migrated to new expenses, see the Migration section below.

- **Track costs using expenses only:** All costs can only be tracked using expenses. In previous versions, costs could also be recorded on service and product items using the Cost field. This change allows you to keep a better track of your costs because they can all be found in a single place. Here's what's different:

  - Product and service items no longer have a Cost field because all costs are now recorded using expenses. To preserve costs recorded on old items, we create non-billable expenses for them.

  - The following placeholders were removed: item.cost, item.profit, project.profit, invoice.profit.

- Since products no longer record costs, it doesn't make sense to have non-billable product items anymore. All product items are now billable and they are used exclusively for billing.
- Billable projects have a new view only for products.

For more details on how costs from product and service items are migrated as expenses, see the Migration section below.

- **Non-billable projects:** In previous versions, all projects were billable. Now you can mark them as billable or non-billable.

  When working with non-billable projects, all tasks, expenses and trips are non-billable and you don't have to set any billing attributes to them. Also, if all your projects are non-billable you can hide billing views from the sidebar and billing columns from the tables (right-click their header to set visible columns) to make the interface simpler.

- **Non-project invoices:** In previous versions, Fanurio was able to create only project invoices, invoices for clients with billable projects. This works most of the time but restricting invoices to projects prevents users who don't use projects from billing their clients. That's why you can now create invoices without having to deal with projects.

  When creating an invoice using the New Invoice window, you can now use:

  - the New button to add new product items (not related to a project) and
  - the Add Project Items to add project items.

  We've also added the **invoice.nonProjectItems** placeholder to let you access the non-project items of an invoice in an invoice template. Older invoice templates will have to be updated to use this placeholder if you need to create invoices with non-project items.

- **Redesigned invoice items:** Items are now used strictly for billing, they can no longer be marked as billable or non-billable. There are four types of items:

  - service items bill tasks,
  - expenses items bill expenses,
  - mileage items bill trips and
  - product items bill anything at project level or directly at invoice level.

  Items no longer have date and notes fields. The item.date placeholder is now deprecated and it has the following meaning:

  - for service items, it returns the start date of the task
  - for expense items, it returns the date when the expense was made
  - for mileage items, it returns the date of the invoice
  - for product items, it returns the date of the invoice

  The item.notes placeholder was removed. Older invoice templates that use this placeholder will have to be updated.

  The business catalog accessible from Business » My Business Details+Catalog only manages catalog items for products. Service items and expense items no longer have a catalog item. You can now use task categories to organize tasks and expense categories to organize expenses.

- **Date range invoices:** The Add Project Items window has a new field that allows you to specify the billing period. The Period field is very helpful if you need to bill a specific period like last month because it automatically selects the matching tasks, expenses and trips.

  Tasks that contain time entries outside the billing period will be billed partially, something that wasn't possible in previous versions. You can now create a single task (Consulting) and bill it multiple times instead of creating multiple service items, one for each billing period (Consulting January, Consulting February, etc.).

  When you specify a period in the Add Project Items window, Fanurio fills in the Period field from the New Invoice window automatically.

- **Time, tasks and sales reports:** Project reports were replaced by time, tasks and sales reports. We've also added a new set of placeholders that you can use to create your own templates.

  You can now create sales reports for your invoices whether you need to send a client statement, analyze your billing history by month or create a report at the end of the year.

Improvements

- **Import time from CSV**
  - CSV formats needed to import time in Fanurio can now be exported to and imported from disk. It's now easier to share import formats with other users.
  - Added two more separators: colon (:) and vertical bar or pipe (|).
- **User interface**
  - The status bar and the filters from the main views use a smaller font on OS X.
  - The timer display also shows the client name and the project name. Now, when the timer is running, the timer display shows the client, the project and the task name. If they are too long, they are truncated.
  - Added a Close button to the Edit Timer window.
  - Reordered filters used for the projects tree.
- **New timesheet format:** The timesheet format version is now "3.0". Fanurio can still import files created with the old formats (version "1.0" and "2.0"). The new format uses tasks instead of service items.
- **Freemarker 2.3.20:** Updated Freemarker from version 2.3.8 to version 2.3.20. This update is relevant only to expert users who need to create templates that use features available in versions newer than 2.3.8.
- **Taxes with three decimals:** Taxes can now use three decimals instead of two (eg 7.275%).

Bug Fixes

- The tax total wasn't calculated for new invoices.
- Finished projects were available for billing although only unfinished projects should be billed.
- When invoicing a specific project, that project is selected by default in the Add Project Items window.
- A client tree node was expanded when its popup menu was displayed.

- The elapsed time was not calculated correctly in the New Time window when the "both" option was used.
- If Fanurio was configured to confirm the exit and it was closed using the window close button (red x button) then the confirmation message was displayed twice if the exit was canceled.
- The mini-timer isn't resized correctly on Ubuntu 13.10 running Unity. See this note [issues-ubuntu-undecorated-window] for more details.

Migration

- **Services:** Here's how services are migrated to the current version.
  - Project service items are converted to project tasks.
  - Invoiced tasks have one service item that bills them.
  - Invoiced tasks are marked as completed and their completed date is set to the invoice date.
  - Invoiced tasks billed in units have the billable quantity set to 0 (zero), there's nothing else to bill for these tasks.
  - Service catalog items are converted to task categories. Task categories can be managed from Business » My Business Details+Tasks.
- **Expenses:** Here's how expenses are migrated to the current version.
  - New expenses don't have a Name field so their Description contains the Name and the Description of old expenses.
  - New expenses don't have a Quantity field so the Amount of a new expense is the Quantity x the Cost of the old expense.
  - All invoiced expense items retain their properties (quantity and price).
  - Expense catalog items are converted to expense categories. Expense categories can be managed from Business » My Business Details+Expenses.
- **Products and services with costs:** Here's how products and services with costs are migrated to the current version.
  - Non-billable product items are converted to non-billable expenses because all products are now billable. Expenses resulted from non-billable products have the tags "#migration,#non-billable-product".
  - For billable product items with costs (their Cost field is not zero) we create non-billable expenses because products no longer have a Cost field. Expenses resulted from billable products with costs have the tags "#migration,#product-with-cost".
  - For service items with costs (their Cost field is not zero) we create non-billable expenses because services no longer have a Cost field. Expenses resulted from services with costs have the tags "#migration,#service-with-cost".

    Hourly-rated service items with costs are harder to migrate and the amount of their expense is zero. The Notes field of these expenses contains the value of the Cost field.

  If you want to review expenses resulted from products and services, go to the Expenses view and search them by the Tags field.
- **Projects:** Projects no longer have billing settings for the cost of expenses and their unit of measure. The only expense-related setting is the one that indicates whether expenses are billable or not by default.

Also, all projects are now billable by default. If some of your projects are non-billable, you need to edit them. To edit a project, go to the Projects view, right click it in the Projects tree and select Edit Project.

- **Items:** Items no longer have the date and notes fields.

6.9. Version 2.7 (July 9, 2013)

This version contains bug fixes, small improvements and new features for version 2.7.

New Features

- **Sidebar:** The sidebar is displayed on the left of the main window and allows you to change the current view. You can change the visibility of the sidebar from the menu (View » Hide / Show Sidebar), using the Ctrl-Shift-S (Cmd-Shift-S) shortcut or from the toolbar by clicking the button next to the views drop-down box.

  If you don't need all the views, you can simply customize the sidebar to display only the views that you use. From the main menu, select View » Customize Sidebar or click the button located at the bottom of the sidebar. This can be useful if you don't use Fanurio for billing and you want to hide the Invoices and Payments views.

  If you used an older version, the view buttons from the toolbar were replaced by a drop-down box. This solution allows us to introduce more views in the future without crowding the toolbar.

- **Rounding time for each time entry:** In previous versions, time rounding applied to the sum of all the time entries of a service item. Now, you can choose whether time is rounded for the sum of all the time entries or for each individual time entry.

  Let's suppose you have a service item with two time entries (t1 = 16 minutes and t2 = 32 minutes) that rounds time up to 15 minutes. Here's how time is rounded in both cases:

  - sum of all the time entries: **round(**0:16 + 0:32**)** = **round(**0:48**)** = 1:00. (previous versions)
  - each time entry: **round(**0:16**)** + **round(**0:32**)** = 0:30 + 0:45 = 1:15. (option added in this version)

- **Terms per client:** You can now specify payment terms at client level, not just at business level. This means that whenever you create an invoice for a client, it will use the terms of that client by default.

  To edit the terms of a client, go to the Projects view, right-click the client in the projects tree and select Edit Client. Then go to the Billing tab where you can access the terms in the Terms field.

- **New Timesheet columns**
  - **Billable** indicates whether a time entry is billable or not
  - **Invoiced** indicates whether a time entry was invoiced or not
  - **Invoice** show the invoice number for an invoiced time entry

- **Optional columns:** Some of the main tables don't show all the columns by default to keep the interface simple. For instance, the Timesheet table doesn't show the following columns by default: Invoiced, Invoice, Finish, Pause and Hours.

  In order to configure the visible columns of a table, right-click its header and choose which columns should be visible and which columns should be hidden.

Improvements

- **Adding items to an invoice:** The New Invoice and Edit Invoice windows have an Add button that opens the Add Project Items window. This window lets you add billable items from the projects to the invoice.

  Adding project items to an invoice is now easier because items can be selected by project or by type (service, expense or product).

- **New Time service items are sorted alphabetically:** Service items from the New Time dialog are sorted alphabetically. Previously, they were sorted by the date of their latest time entry.

- **Tax exempt text in invoice template:** The template editor can now configure the text that is displayed when an item is exempt from taxes. By default, it shows 0.

  This setting can be changed from File » Template Editor, box Invoice > Items > Tax Exempt. This setting works only if box Invoice > Columns > Tax is checked.

- **Table totals for visible columns:** All views display totals at the bottom of their tables. For instance, the Timesheet view displays totals for the Time and Hours columns if these columns are visible. If a column is hidden, its total is no longer displayed at the bottom of the table.

  In previous versions, totals where displayed all the time regardless of the visibility state of their columns.

- **Renamed item-related placeholders:** Renamed some item-related placeholders in order to make it clear that they are referring to items.

  See the templates placeholders section to learn which placeholders were renamed in this release.

- **ISO 8601 date and time format:** The date and time format used in the timesheet XML file is now ISO 8601 [http://en.wikipedia.org/wiki/ISO_8601]. The timesheet format version is now "2.0". Fanurio can still import files created with the old format (version "1.0").

  This change is important if you rely on automated tools to import time from Fanurio.

- **New database migration mechanism:** The application uses a new database migration mechanism that upgrades databases from older versions of Fanurio to the latest version automatically.

Bug Fixes

- Tax total wasn't calculated correctly when using multiple currencies.

- Failed to change the Date field in New Time when time was entered relatively to finish after midnight and the start time was in the previous day.

- Table columns sometimes disappeared and only one of them was visible.

Migration

- **Invoice templates:** Invoice templates from the templates folder (File » Show Templates » Invoices) are migrated automatically to use the renamed placeholders.

  Before migrating an invoice template, Fanurio makes a copy for it with the same name and the timestamp when it was migrated.

- **Database:** The database is migrated automatically using the new migration mechanism.

  Two backup copies are saved in the backup folder: one copy has the .fro extension and it can be used to restore the database from File > Restore Backup while the other copy has the .zip extension and it will never be deleted by Fanurio when there are too many backup copies.

6.10. Version 2.6.1 (March 29, 2013)

Bug Fixes

- Failed to start with Java 5.
- Failed to start if global hotkeys where enabled in the previous version.
- Time entry descriptions were not available when displaying them grouped by date in the invoice template.
- Linux: Using xdg-open instead of gnome-open to open files, folders and urls.
- Windows 8: Run on startup didn't work.
- Windows 8: Fanurio wasn't using the Windows task bar features (thumbnails, pinning, badge icons).
- Windows 8: Fixed the user interface look and feel.
- The template editor failed to save templates that contained Freemarker code.
- The dialog used to create and edit new Timesheet Import configurations is now scrollable so it can fit on any screen.
- Mac OS X: The folder chooser didn't work with Java 7.
- Mac OS X: The menu bar and the status bar were not painted correctly on Java 7.
- The age of an invoice is calculated as the difference between the dates and not the difference between the time stamps. If an invoice was created yesterday at 11:00 PM and now it's 8:00 AM then in the old version the invoice was 0 days old while in the new version it's 1 day old.

6.11. Version 2.6 (November 6, 2012)

This version contains new many features, improvements and bug fixes. Here's a list with all the changes:

New Features

- **Multiple timers:** Fanurio can now manage multiple timers, not just one.

  This feature is very useful if you start working on something (service A) and then you get a call from a client that forces you to interrupt what you were doing and work on something else (service B). In this case, you start a timer for service A and then when the client calls, you pause the timer (interrupt service A) and you start a new one for service B. When service B is over, you stop its timer and save the time. Then you can resume the timer for service A.

  When you start a timer, you can use Start New Timer (F5) to start it immediately or you can use Start New Timer... (Shift-F5) to start it a few minutes in the past just in case you forgot to do it. You can pause and resume a timer as many times as you need to. Previous versions allowed only one timer. This new version makes it possible to pause the active timer and start a new one. In fact, if you have an active timer and start a new one, the active timer is paused and the new timer becomes active. You can have only one active timer but an unlimited number of paused timers.

  - **Starting a new timer:** The Start New Timer action starts a new timer. If a timer is running, it is paused and the new timer replaces it and becomes active. Each new timer is added to the list of timers.

    If at least one timer was started, the Timers icon has a badge that shows the total number of timers.

  - **Start New Timer:** The Start New Timer... (Shift-F5) action allows you to start the timer even if a service item is not specified. This change makes the action more flexible.

  - **Resume timer on crash:** Just in case the application crashes, it saves the state of the timer every minute so that on the next restart it will ask you to resume it.

  - **Edit Timer:** The Timer Info action was renamed to Edit Timer.

- **New Project field for item dialogs:** Item dialogs have a new field that shows the project to which they belong to.

  New items (services, products or expenses) can now be created from the menu, just go to Business » New Item and select New Service Item, New Product Item, New Expense Item or click one of the catalog items to create an item based on a catalog item. This feature helps you create a new item without opening a project.

  You can still create project items by opening a project and clicking the New button at the bottom of the project table.

  Note: Items can no longer be moved to a different project by dragging from the project table to a project from the projects tree. The same can be done now by editing the item and changing its project.

- **Disable reminders:** The timer reminders can now be disabled temporary.

  If you enabled the reminder for starting the timer and at some point you no longer want to be reminded to do it in the near future, you can simply disable the reminder temporarily by unchecking the box from the reminder window. This will disable the reminder until the timer is started or until the application is restarted.

- **New project fields:** Projects have two new fields called **Number** and **Reference** that are meant to complement the existing **Name** field. You can use Number to identify the project internally and Reference to keep the client's project number.

- **Import time from any CSV file:** Fanurio can now import time from any CSV file. The new wizard lets you specify how the columns of a CSV file can be mapped to its own fields. The expressions that map the CSV columns can be quite complex so that time is imported accurately in Fanurio.

  See this section for more details.

- **Third-party timesheet import:** Fanurio can now import time from HoursTracker [http://hourstrackerapp.com/] (an iPhone app). See the list of all third-party applications from which it can import time.

- **New Timesheet columns**

  - **Pause** shows pause time

  - **Hours** shows time in decimal format

- **Hide and resize table columns:** Columns from the project, timesheet, invoices and payments tables can be hidden and resized by right clicking them. Fanurio can remember the position and width of a column and whether it is visible or not.

- **Search time entries by invoice number:** The search field from the Timesheet view has a new filter that can find invoiced time entries by their invoice number.

  See this section for more details on all the available search filters.

- **Delete, tag and move multiple time entries:** Multiple time entries can now be deleted, tagged or moved in the Timesheet table because it allows multiple time entries to be selected.

- **Edit tax groups:** The name and status of a tax group can now be edited (Business > My Business Details > Taxes). This can be helpful if your taxes (eg VAT) changed and you need to hide the old tax group so it doesn't appear in the New Invoice window.

- **Edit invoices:** Invoices can now be edited. In previous versions, if something was wrong with the invoice or some of its items, it had to be deleted and then recreated. To edit an invoice, go to the Invoices view, select the invoice you want to edit and the click the Edit button below the invoices table.

  Important: Editing an invoice is not recommended once it was sent to a client because that would alter an official document.

  The New Invoice window was also redesigned to make it easier to edit the items of an invoice. The table from the New/Edit Invoice window shows more information about the items that belong to the invoice.

- **Minimum time when rounding time for billing:** Time rounding rules can now have a minimum amount. This helps create rules like "round time to the nearest 15 minutes interval but bill at least 30 minutes".

Improvements

- **User interface**

  - The service and product dialogs now show the discount links all the time.

  - The time rounding link from the service dialogs is now next to the hourly radio button instead of the Quantity field.

  - The New Time dialog has a small button next to the Date field that can hide the time input options. Use it to unclutter the dialog if you prefer the same method of adding time.
  - In the New Time dialog, the service link was replaced by two fields: project and service. The new layout makes it easier to select a project and to create new service items.
  - The layout of Start New Timer... and Edit Timer dialogs has been changed to resemble that of New Time.
  - Changed the icons used to show the state of the timer on Windows 7 in the task bar.
  - The main window is restored correctly if it was maximized before the application is closed.
  - New Time, Start New Timer and Start New Timer... use the service item selected in the Projects view by default only when this view is visible.
  - The Add Payment action was renamed to New Payment. The New Payment action can also be found in the Business menu and in the Payments view, not just in the Invoices view.
- **Simpler reminders:** Changed the timer reminders so they suggest only one action. They are simpler now because they suggest only one action.

Bug Fixes

- The F6 shortcut that pauses the timer didn't work.
- Global hotkeys didn't work on 64-bit platforms.
- Non-ASCII characters weren't saved correctly by the template editor.
- Failed to start when upgrading from version 2.0 or older.
- The Background check box from File > Template Editor > Page was always disabled.
- Failed to create a new project when the selected client was inactive.
- PDF documents with multiple pages displayed the background layer only on the last page.
- Changed the New Time Entry shortcut from Ctrl-M (Cmd-M) to Ctrl-T (Cmd-T). This shortcut conflicted with the minimize action on Mac OS X.
- The Notes field from the New Invoice dialog didn't wrap its text.
- The main window was undocked from the maximized state when a new time entry was created.
- The elapsed time was calculated incorrectly when the timer was running for more than 24 hours.

6.12. Version 2.5 (February 28, 2012)

This version introduces product items for billing goods and adds a new template editor. Here's a list with all the changes:

New Features

- **Introducing products:** Although you can use service items to bill both work and materials, it doesn't feel right to create a service item to bill goods. For instance, if you have a small IT shop and you need to charge a client for 2 computers, creating a service item is a little strange because selling 2 computers is not actually a service.

  You can now use services to bill work performed for the client and products to bill materials, equipment or anything else that doesn't involve actual work. Use products when you want to sell or resell goods.

- **Business catalog:** A business catalog is a list of all products and services offered by a business. The catalog can be found under Business » My Business Details » Catalog and can contain one or more catalog items. Catalog items make it easier to create new project items since they contain almost all billing information, except for the quantity. See the business section for more details.

  This is not a new feature in Fanurio but a new name for an old feature. The old catalog could be found under Business » My Business Details » Services and Rates and its elements were called **item categories**. As a result of this change, some templates may not work because the **item.itemCategory** placeholder was renamed to **item.catalogItem**.

  Catalog items have three new fields: code, description and cost. These fields make it easier to create project items.

- **Markup calculator:** The price of a service item or product item can be calculated easier using the markup calculator. Just click the Markup link next to the Cost field to enter the markup and the price will be calculated automatically. This feature only works if you enter a non-zero amount in the Cost field first.

- **Cancelled Invoices:** If you have unpaid invoices for which you don't expect the client to pay, you can now mark them as cancelled. See the invoice section for more details.

- **New Timesheet filters:** Added new filters to the Timesheet view that make it possible to display time entries by client, project and service item status. For instance, it's now possible to display time entries from active clients with unfinished projects and uninvoiced service items.

  The search field can also search by service items. If you have a client called **Aristotle** with a project called **Rhetoric** and a service item called **Proofreading the manuscript**, just type **Aristotle;Rhetoric;Proofreading the manuscript** and it will display this exact service item. If you type **Aristotle** it will display all time entries for this client whereas if you type **;Rhetoric;** it will display all time entries for the project. The semicolon is very important as it helps Fanurio distinguish between fields.

- **Save Timesheet filters:** The new filters allow you to see time entries for *"active clients with unfinished projects and uninvoiced service items"* but you could also set the filters to display time for *"last month"* or *"this month"*. Switching between any of these filter configurations means that you have to adjust each filter individually.

  To make things easier, you can now save filters and give the configuration a name. Just click the small cog icon from the filters area (on the right) to save them or use one of previously saved configurations.

- **Persistent date filters:** Date filters from the Timesheet, Invoices and Payments views are now saved when the application is closed.

You can now set the Date in the Timesheet view to Today or This Week and you won't have to set it again when the application is restarted.

- **Previously removed features**
  - Clients can be marked as "tax exempt".
  - Invoices can now be exported and emailed from the View Invoice window.
- **New icons:** Changed most of the icons to make the actions they represent more recognizable. New icons for the button that switches between the main window and the mini-timer window.
- **Third-party timesheet import:** Fanurio can now import time from Toggl [http://www.toggl.com] and Freckle [http://letsfreckle.com/]. See the list of all third-party applications from which it can import time.
- **New project field:** Projects have a new field called **Description** that is meant to complement the existing **Notes** field.
- **Wizard for three taxes:** The tax wizard can now create three taxes. Italian users can now use it to define the three taxes (Rivalsa previdenziale 4%, IVA 21%, and Ritenuta d'acconto -20%).
- **New templates editor:** The templates editor has been redesigned so that it can also edit templates. See this section for more details on how to use it.
- **New languages:** Fanurio is now available in Czech, Italian and French.

Improvements

- **Time entry**
  - Changed the order of the fields in the time entry dialog.
  - The start and finish times of a time entry can be specified partially by entering only the hour. If the minute is not specified, it will assume it's 0 (zero). If time uses the AM/PM format and the field is not entered, it will assume AM. In other words, entering 11 will result in 11:00 AM.
  - Selecting 'earlier' from Start Timer ... (Shift-F5) will display a time field to enter the actual time when the timer should be started instead of specifying the start time relatively in minutes.
  - Renamed the Add Time action to New Time and changed its shortcut from Ctrl-T to Ctrl-M. The action is also available from the toolbar.
- **User interface**
  - The timer tooltip from the main window and from the mini timer shows the client, project and service item name.
  - Projects and clients from the Projects Report widow can now be filtered by their status. This makes it easier to select which projects should be included in the report.
  - The names of attachments can be changed when emailing invoices.
  - When exporting invoices, payments or the timesheet and the file format is changed, the name of the file remains unchanged and the extension changes.

Bug Fixes

- The Start Timer tray menu is enabled only if the timer is stopped. Previously, it was also enabled if the timer was paused.

- MAC address was not detected correctly on Linux.
- Fixed a memory leak introduced by Apple in Java 1.6.0_29 running on Mac OS X 10.7 and 10.6.
- Fixed several memory leaks that slowed down the application if used for a long time. Memory leaks occured when a project was opened, the timer was stopped or an invoice was exported. Also improved it to use memory more efficiently.
- Idle time was notified without any indication when the inactivity duration started which was confusing if multiple notifications were made. Also, the idle duration changed after it was notified.

6.13. Version 2.4.3 (November 10, 2011)

This version contains bug fixes and small improvements for version 2.4.

Bug Fixes

- Tax groups are sorted by name in the New Invoice dialog. The default tax group is now the first one lexicographically.
- Fanurio didn't delete older backup copies as documented in the manual.
- Time rounding didn't work for new category-based service items.
- Can't open folders on Mac OS X.
- Fixed the user interface for Mac OS X Snow Leopard and Lion.
- Some dialogs occupy the whole screen when running Fanurio using Open JDK on Linux.
- The application icon is no longer blurry in Ubuntu's Unity launcher.
- The .deb package now depends on Open JDK or Sun Java. Ubuntu 11.10 cannot install the .deb package if it depends only on Sun Java.

6.14. Version 2.4.2 (June 10, 2011)

This version contains bug fixes and small improvements for version 2.4.

Improvements

- Fanurio didn't close gracefully on Windows and Linux (Gnome) sometimes resulting in corrupted databases or messages that the database is in use by other instances of Fanurio.

Bug Fixes

- Drop down lists were not visible on Mac OS X when using the mini timer.
- When creating new service items from the"Select Service" dialog, the new item doesn't use the project billing settings.
- DBus library wasn't loaded on 64-bit Linux systems. This prevented Fanurio to close gracefully on these systems.
- Couldn't open files with spaces in their paths on systems running Java 5.
- Can't create a service item for a new client from the "Select Service" dialog.
- New email server settings were used only upon restart.
- Failed to start on Ubuntu 11.04.
- Windows are minimized using Cmd-M and closed using Cmd-W on Mac OS X.
- The Shift-F5 shortcut also works for the mini timer.
- The folder chooser was slow when displayed for the first time on Windows.
- Didn't shutdown gracefully when running on Windows using 64-bit Java.
- The warning that the database is already open is displayed if the database is accessed from two different IP addresses.
- Sometimes crashed at shutdown on Windows creating files called **hs_err_pidxxxx.log** in the installation folder. No data was lost but the message was annoying.

6.15. Version 2.4 (January 31, 2011)

This version has an improved user interface that is optimized for Mac OS X and Windows 7. It can also integrate with QuickBooks Pro. Here's a list with all the changes:

New Features

- **User interface**
  - All
    - New toolbar icons.
    - The timer display has been moved from the status bar to the toolbar.
    - The toolbar has an info button that can be used when the timer is running to attach a description.
    - Each view now has the buttons on the left and the totals on the right.
    - The project and client filters can now be accessed using the button placed below the tree.
  - Mac OS X 10.4+
    - New Mac OS X-like user interface.
    - New Preferences pane.
    - The timer can be accessed from the dock icon menu besides the menu bar icon.
  - Windows 7
    - New Windows 7-like user interface.
    - Fanurio can be pinned to the taskbar.
    - Fanurio displays an overlay icon on the taskbar button when the timer is running to show its status.
    - The taskbar button's thumbnail has a toolbar that can control the timer.
- **New user manual:** The manual has been rewritten to make it easier for both new and existing users to find what they need.
- **Requires Java 1.5 to run:** Fanurio no longer works with Java 1.4. This shouldn't be a problem for most people. Only Mac OS X 10.3.9 has Java 1.4. The other platforms (Mac OS X 10.4+, Windows or Linux) have Java 1.5 or later.

  See this section for more details on how to install the latest version of Java on your computer.
- **QuickBooks integration:** Fanurio can export time to an .iif file that can be imported by QuickBooks Pro. See this section for more details.
- **Tags for time entries:** When you add a new time entry you can also enter one or more tags that have to be separated by commas (eg: tag1,tag2,tag3). See this section for more details.
- **TeX/LaTeX templates:** Fanurio recognizes TeX/LaTeX (.tex) files as templates.
- **Configurable folders:** Application folders like the database, backups and templates folders can be changed from the application. This could be helpful if you want to store data somewhere else than the default locations.

Improvements

- **UI Improvements**
  - The services and expenses are ordered by date in the New Invoice dialog.
  - Popup menu for text components. Right-click a text component to access actions like Cut, Copy and Paste.
  - The background color for gray table rows (paid invoices, invoiced items, invoiced time entries) is now gray instead of blue to help figure out the status of the entry.
- **Global hotkeys:** The list of global hotkeys can now be changed from the Options/Settings dialog.

Bug Fixes

- Cannot add item categories in Services and Rates from Business » My Business Details.
- Failed to select dates correctly when using a Wacom pen.
- The Preferences menu action was no longer available after upgrading to Java 1.6.0_22 (Java for Mac OS X 10.6 Update 3).
- The reminders are dismissed if the timer changes its state. Previously, if a reminder was displayed to announce that the timer wasn't started lately and then the timer was started from the tray icon menu, the reminder wasn't dismissed.

6.16. Version 2.3 (August 17, 2010)

This version has a redesigned templates module that can handle template files in several formats: HTML, Microsoft Word 2007, OpenOffice OpenDocument and others. **Templates can now be edited using well-known visual editors like Adobe Dreamweaver, Microsoft Word or OpenOffice.** It's recommended that you uninstall the previous version before upgrading to remove old default template files. Here's a list with all the changes:

New Features

- **Redesigned templates module**
  - *New file formats (.html, .docx, .odt):* Fanurio can now process templates saved in several file formats that can easily be edited using a visual editor like Adobe Dreamweaver, Microsoft Word and OpenOffice.

    There's a new template guide section that explains how to create and edit templates. It also contains a list of all supported file formats with comments on each one of them.
  - *Old format (.ftl):* Old templates (.ftl files) are still recognized by Fanurio and they will work just fine. If you have custom invoice templates, you will still be able to use them. However, they cannot be edited using a visual editor since they were not designed for that.

    Old .ftl templates are using the angle bracket syntax for directives while the new .html templates use the square bracket syntax [http://freemarker.sourceforge.net/docs/dgui_misc_alternativesyntax.html].
  - *Templates location:* The default templates location has changed. The invoice templates folder (templates/invoices) and the project reports templates folder (templates/project_reports) are now located under the same folder (templates). See the troubleshooting section to learn where the templates are kept on Windows, Linux and Mac OS X.

    Exiting templates are moved automatically by the new version to the new location.
- **Export payments:** Payments can now be exported just like invoices. This is useful if you need to create payment receipts. Fanurio comes with a default template that can be used to export payments to HTML or PDF but it can also use templates in other formats.
- **New item field - unit of measure:** Both services and expenses have a new field to specify the name of the unit of measure. This is very useful if you need to specify what you're actually billing on your invoices.

  For instance, if you are providing translation services you may want to bill words or pages and not just units.
- **New client and business fields:** The business and the clients have new fields to keep various identification numbers. These numbers together with the name are now grouped in a new section called company.
  - *business number*: All businesses are assigned a number when they are created. This number has different names around the world like: Codice Fiscale (Italy), Business Number (Canada), Australian Business Number (Australia), CUI (Romania), SIRET (France).
  - *tax number*: Some countries use the business number for tax purposes while others require a different number. Use this field only if it's required by the law in your country. For instance, Italian companies use Codice Fiscale for the business number and Partita IVA for the tax number.
  - *other number*: Use this field if your company has other registration numbers. You could use this field if your company has a trade register number. French companies are registered with

"Registre du commerce et des sociétés" or RCS while German companies are registered with Handelsregister.

- **New invoice field:** Invoices have a new field called period. This field can be used to specify the date or period when the services were provided. This field is optional but in some countries like Germany this information needs to be displayed on invoices.

- **Filter clients by their status:** Clients have a new status field that can help to control whether they are active or not.

  To hide inactive clients, just click the small arrow icon above the clients tree and select Active. Fanurio will display only active clients.

- **Filter invoices by payment date:** Invoices can also be filtered by payment date if the status filter is set to Paid.

  If you keep track of your business's income using the cash method, you can use this filter to see how much money you received during a certain time period. If you are using the accrual method, you can use the Date filter to see invoices created during a certain time period.

- **Third-party timesheet import:** Fanurio can also import time from other applications, not just from another instance of Fanurio.

  Now it can import time from iTimeSheet [http://itimesheet.free.fr], TimeLogger [http://www.costmosoft.com/], Time Tracker [http://time-tracker.avh4.net/] (iPhone) and BizTrackIt [http://www.shrunkenhead.biz/biztrackit.html] (Blackberry) but it can be extended to support other applications as well.

- **Search time entries by description:** Time entries listed in the Timesheet view can now be searched by their description.

- **New languages:** Fanurio is now available in Dutch and Portuguese. Go to Options/Preferences and then to the Locale section to change the language.

Improvements

- **User interface**

  - The General and Backup tabs from the Settings/Options dialog were replaced by the System and Paths tabs.

  - The New Invoice dialog shows the date of the items in gray. This helps distinguishing between items with the same name but different dates.

  - If the application is closed when the timer is running, one can now cancel this operation and not just decide whether the time recorded by the timer is saved to a service or not.

  - Remembers the size of the Timer Info window.

  - Warns if a time entry is empty (time is zero).

- **Timesheet export**

  - Changed the XML file used to export the timesheet to indicate whether a service is billable or not and the category it belongs to.

  - The CSV and Excel files used to export the timesheet display time in hour format instead of displaying the number of seconds.

- **Backup:** Fanurio creates a backup copy every time it starts and it no longer creates backup copies when it is closed. On Windows and Linux, the backup copies created when the computer is shut

down are not always complete. Fanurio doesn't get enough time from the operating system to finish them. If a backup copy is not valid, Fanurio lets you skip that copy.

- **Multiple instances:** If Fanurio is already running and it is started again, the current instance is made visible. Previous versions displayed a message that Fanurio was already running.

Bug Fixes

- Global hotkeys are not working on 64-bit computers running Linux.
- Files and folders are not opened on Linux computers using KDE.
- Idle time was no longer detected on Mac OS X Snow Leopard.
- Couldn't send emails on non-SSL connections.
- Select the project in the projects tree when adding time to a service item.

6.17. Version 2.2 (October 14, 2009)

This version focuses on making the backup process more reliable. Backup copies are now checked for integrity to prevent corruption and they are created automatically even when the application is running. An additional audit file is used to save the data entered recently. Here's a list with all the changes:

New Features

- **More reliable backups**
  - *Mandatory backups:* Backups are no longer an option, they are mandatory. We removed the two options that controlled this behavior (whether they are enabled and the maximum allowed number of backup copies). Fanurio keeps at most 25 backup copies now.
  - *Writable backup folder:* Fanurio checks on startup and when the backup folder is changed if it is writable. This prevents situations when the backup folder is not writable. For instance, if the backup folder is located on a USB stick that's not plugged in.
  - *Automatic backups:* Fanurio created a backup copy when it was closed but now it also does this when the application is running. Automatic backup copies are made every six hours (1:00, 7:00, 13:00, 19:00). This is very useful if a computer runs for days without being shut down.
  - *Backup integrity:* Fanurio checks whether the last backup is valid each time it starts. It also does this after each automatic backup.
  - *Audit file:* An audit file is used to record the latest database operations. Depending on how much you use the application, it should record at least the last 5 days. The audit file can be used to recover data just in case there is no backup copy and the database is corrupted.
  - *Clean shutdown on Windows:* Fanurio exits correctly on Windows if it is running when the user logs out or shuts down the computer. This fix addresses an existing issue for the Windows platform. It has never been a problem on Mac OS X but it still is on Linux.
- **Global hotkeys:** Global hotkeys are keyboard shortcuts that can be used from within any running application. Only the Windows and Linux versions of Fanurio have support for global hotkeys.

  Global hotkeys are very useful if you need to control the timer without clicking the tray icon or making the main window visible. Please see the global hotkeys section for the actual list of shortcuts.
- **New settings**
  - *Date format:* The date and time format depends on the region you choose from the Options/Preferences dialog. The new date format option lets you use an explicit format.
  - *Confirm exit:* Just to make sure you don't close the application accidentally, you can now confirm if you want to close it. The new setting is disabled by default and must be enabled from the Options/Preferences dialog.
- **Search invoices by reference:** New column for the invoices table to display the reference field which usually is the purchase order. The search field above the invoices table can now be used to search invoices by their reference number.
- **The return of - default date for new time entries:** The default date of a new time entry is **today** again. It was changed in version 2.1 to be the last date when time was added to the service but it didn't prove to be a good idea.

  To enter time for another date than today, go to the Timesheet view and use the Date filter to specify a date. Then, when you'll use the Add Time button from the Timesheet view, the default date will be the date specified by the Date filter.

Improvements

- **Template:** New invoice.overdue placeholder to know whether an invoice is overdue or not.
- **Check for updates:** Fanurio automatically checks for updates even if it is not restarted for many days. Previously, it only checked when started.
- **Improved mini timer**
  - New minimize button in the mini-timer window.
  - New toolbar button that changes the view to the mini timer window.
  - Fixed the bug that maximized the mini-timer window in some cases.
  - Renamed the **Switch View** action to **Switch to Mini Timer** and **Switch to Fanurio Window**. The old name wasn't that obvious.
- **User interface improvements**
  - The Delete key works on all important tables.
  - Text areas lose focus when the tab key is pressed.
  - Show Contents is no longer an option in the project popup menu.

Bug Fixes

- Price and cost currency mismatch is no longer checked if cost is not used.
- Invoices with zero total are marked automatically as paid.
- Fixed the name of email attachments.
- The following placeholders were not available in templates: business.other, client.other, business.mobile and client.mobile.
- Tables that span on multiple pages don't display the bottom border.

6.18. Version 2.1.1 (July 22, 2009)

This is a patch for version 2.1 that contains bug fixes and small improvements. Here's a list with all the changes:

Improvements

- Default name for backups to save time.
- Close the currently opened project when a client is selected. It can be confusing sometimes.
- Toggle buttons from the reminder dialogs are disabled when their section is visible (Snooze... and Snooze are different).

Bug Fixes

- The application freezes when the idle reminder is triggered
- Failed to export date and time objects to CSV
- Pause/resume timer tooltip doesn't change
- The view toggle buttons are not quite visible when selected on Vista
- Projects are opened on mouse click instead of mouse pressed

6.19. Version 2.1 (July 9, 2009)

This version opens up Fanurio for users who speak other languages than English. It is now available in German, Romanian and Spanish but it can be translated to any language. Other than that it has many improvements that make it more usable. Here's a list with all the changes:

New Features

- **New setting - language:** There's a new setting in the Locale section of the Options/Preferences dialog that lets you specify the language of the user interface of the application.

  The current version can be used in **English**, **German**, **Romanian** and **Spanish**. Support for other languages will be added in future releases.

  We also have partial translations for **French**, **Italian**, **Dutch**, **Danish** and **Swedish** but they will not be distributed officially until they are completed. If you are interested in using Fanurio in any other language than the official ones, just let us know.

- **New setting - time format:** The date and time format depends on the region you choose from the Options/Preferences dialog. The new time format option lets you use an explicit format.

  For instance, if your region is English (US) then time is formatted using the 12-hour format. The new option lets you override it and use a 24-hour format.

- **Printing support:** Invoices can be printed directly from the View invoice dialog. To print an invoice, double-click it in the Invoices table then click the Print button at the top.

  Instead of printing an invoice directly from Fanurio, you may want to export it to PDF and then print it. Read this section for more details.

- **Email support:** Invoices can be emailed as PDF attachments directly from Fanurio.

  Since Fanurio already knows the email addresses of your clients, you can send invoices directly from the application. Read this section for more details.

- **Default billing settings:** Projects now have a billing section where you can specify default billing settings for their items. These settings are used when a new service or expense is created for that project.

  For instance you can specify a default hourly rate and a default rule for rounding time. Each time a new service item is created, it will have the default rate and rounding specified at project level.

- **Menu bar icon on Mac OS X:** Fanurio adds an icon to the menu bar on Macs running Java 6 (Mac Intels running Leopard). The menu bar icon displays the status of the timer (running, paused) and its menu has actions to control the timer.

  This menu bar icon menu makes it even easier to control the timer than the iTunes-like mini timer.

- **Idle time detection on Linux:** Idle time detection happens when you start the timer but at some point you must leave the computer. This option is not enabled by default and must be configured from Edit > Preferences > Timer.

  Until now, idle time detection only worked on Mac OS X and Windows. Now, it is also available on Linux for both 32-bit and 64-bit computers.

- **Third-party timesheet import:** Fanurio can also import time from other applications, not just from another instance of Fanurio.

  Now it can import time from iTimeSheet [http://itimesheet.free.fr] and TimeLogger [http://www.costmosoft.com/] (two iPhone applications) but it can be extended to support other applications as well.

Improvements

- **The return of - move time:** The Edit Service Item dialog has a Move button that can move a time entry to a different service item. Time entries can also be moved to a different service item from the Timesheet view by editing them and changing their service.

  This feature was available in version 1.11.3.

- **The return of - toggle project completion:** Right-click on a project from the projects tree and you can toggle its finished state using the "Toggle Project Completion" action.

  This feature was available in version 1.11.3 under a different name, "Mark Finished/Unfinished".

- **Snooze reminders:** The reminders have been improved to let the user override the snooze interval.

  At the end of the day when you don't want to be reminded to start the timer just override the snooze interval to a big value (300 minutes) and it won't bother you to start it.

- **Accidental timer stop:** The timer menu no longer has the Recover Time action that was used to recover time when the timer was stopped by mistake.

  If you stop the timer, Fanurio will display the Add Time dialog. If you stopped it by mistake, just click Cancel and Fanurio will let you resume the timer.

- **Default date for new time entries:** When adding time manually to a service using the Add Time action, the default date and time is no longer "now" but the last date and time when time was recorded for that service.

  This is useful if you enter a lot of time entries manually after a few days or at the end of the week because you don't have to change the date so often from now to a past date.

- **User interface improvements**

  - Improved the New Invoice and Create Template dialogs to be less crowded.
  - The mini timer view also shows the service.
  - Items and expenses now have a check box to indicate they are non-billable instead of two radio buttons
  - The drag corner is now displayed on all windows on Mac OS X
  - Time columns are aligned to the right

- **Others**

  - Renamed contract.clientNumber to contract.reference.
  - A **mobile** and an **other** field have been added to both business and clients.
  - Title and meta tags from the head section of the template are converted to document properties when an invoice is exported to PDF.

Bug Fixes

- Spelling errors in the Options/Preferences dialog.
- The dock icon was updated correctly if the timer was stopped after pause.
- Using Quaqua 5.2.1 to work with Java 6.
- Date filters weren't updated if the application ran after midnight

- Idle detection works on both 32 and 64-bit computers
- The Add Time dialog didn't recognize the start time in some cases on Mac OS X.

6.20. Version 2.0 (February 26, 2009)

This version provides a more intuitive interface to enter services and expenses, can handle multiple payments for an invoice and can import time from other users.

New Features

- **Merged items with expenses:** This is an important change in Fanurio. Please read the entire explanation to understand why we've done this and why it's useful.

  Up until this version, Fanurio used items to represent services and expenses to represent billable, reimbursable expenses. Services were billable by the hour or using a flat rate but one could also mark them as non-billable. The distinction between items and expenses was also made in the user interface where a project had two sections, one for items and the other for expenses. This design has worked quite well since version 1.0 but it has some limitations.

  Here's a list with the most important limitations of the **old design**:

  - *Language:* The term item was misleading because it actually meant a service item.
  - *Project totals:* There was no way to see the total value of a project because the project was split in two (items and expenses).
  - *Non-billable expenses:* There was no way to mark certain expenses as non-billable.
  - *Item dialog:* The dialog used to create or edit items required an extra mouse click to change the billing details. These details were not visible right away.
  - *Taxes:* Items were taxable but expenses were not. This is a real problem in certain countries (Australia, Sweden, UK, etc) where both services and expenses are taxable.

  To address these limitations, we've merged items with expenses and we've made a few other changes. Here's **what's new** and how it compares to the old design:

  - *Type:* Items have a new field called type that indicates whether an item is a service or an expense. Expenses recorded in previous versions of Fanurio are converted automatically to items of the type expense. Attachments from old expenses can now be found in the notes field of new expense items.
  - *Billing:* Any item can now be either billable or non-billable. Previously, only items had this property while expenses were always billable. Billable expenses add nothing to the profit of a project while non-billable expenses add a negative profit.
  - *Measure:* This notion didn't exist explicitly in previous versions. Hourly-rated items were measured in hours while flat rate items where measured in units. The current version makes measure an explicit property. Services can be measured in hours or in units while expenses are always measured in units.
  - *Taxes:* Expenses and services can now be marked as taxable or non-taxable (tax exempt). Until now, expenses were non-taxable and services were taxable.
  - *Category:* Categories help to create and classify services and expenses easier. You can control item categories from Business > My Business Details > Services and Rates.
  - *User interface:* Projects have a single section instead of two (Items and Expenses). The section contains a table that displays both services and expenses. Above the table there are a few filters that control which items are visible while below the table you can see totals for time, cost, profit and total.
  - *New names:* Rate was renamed to price and amount was renamed to total.

- **Item notes:** Items have a notes field where you can enter anything that cannot be specified using the other fields, especially description. You can use the notes field to make notes for yourself and the description field to enter client notes.

- **Payments and statements:** Invoices can now be paid in multiple rounds by using payments. Payments can be seen, managed and filtered in the Payments view (Command-Shift-A or Ctrl-Shift-A). An invoice is considered paid when its balance is zero.

  Invoice templates can also display a statement of all open invoices of a client. To support this, the following template placeholders have been added: client.invoices, client.balance, invoice.balance, invoice.paymentsTotal, invoice.payments, invoice.paid, invoice.paymentDate.

- **Timesheet view:** Added a new Timesheet view to make it easier to access recorded time. In the Timesheet view you can filter time by client, project or service item, by status (invoiced or not invoiced) and by date. The Timesheet view is very useful if you want to review the time recorded for a service item or project or if you want to review what you've done at the end of a day.

  Ctrl-click the Time column of a service item from a project and you will see its time in the Timesheet view.

- **Import timesheet:** Fanurio can help you manage time recorded on multiple computers. You can export time recorded in one instance of Fanurio running on a certain computer and then import it in another instance of Fanurio running on a different computer. Read this section for more details.

- **Invoice age:** The invoices table has a new column that indicates the age of an invoice in days. For unpaid invoices, the age represents the time since it has been created. For paid invoices, the age represents the time between when it was created and when it was paid in full.

- **Invoice notes:** Invoices now have a notes field where you can enter additional notes about the invoice. You can even display these notes on the invoice for the client to see them.

Improvements

- **Invoice Templates:** The File » Create Template dialog has lots of options to configure the default invoice template. You can specify which columns and subtotals are visible, whether the template uses a logo or page numbering and many other options. This is useful for new users who want to create their first invoice template.

- **Locale:** There's a new section called Locale in the Options/Preferences window for region and language. Future versions will support other languages than English.

- **Project reports:** The projects report dialog has been updated to support the new item properties. Items can now be filtered by type (Service or Expense), billing (billable or not) and status (invoiced or not).

- **Exchange rates:** If using multiple currencies, the invoice template can display the exchange rates. See the invoice.exchangeRates placeholder for more details.

- **Exporting data:** Time, payments and invoices can be exported from File » Export.

- **Always on top:** The mini timer view should be on top of all windows on any computer running Java 1.5 or later. Until now it worked only on Windows.

Migration

- Project report configurations are lost and must be created again.

- Existing invoice templates are changed automatically to work with the new version. A copy is saved in the same folder.

- Old expenses are automatically converted to expense items.

- A payment is added for each invoiced marked as paid.

6.21. Version 1.11.3 (October 9, 2008)

This is a patch for version 1.11 that contains bug fixes and small improvements. Here's a list with all the changes:

Improvements

- **Edit project:** Project properties like its name, the notes or whether it is finished or not, can be edited using the Edit Project action. To use this action, right-click on a project in the projects tree and select Edit Project.
- **Copy project items:** Project items can be duplicated using the Copy action after a project is opened. When duplicating an item, its time entries are not duplicated.
- **New shortcuts:** Ctrl-Shift-M (Command-Shift-M on Mac) instead of Ctrl-M to switch to the iTunes-like mini timer.

  Ctrl-Shift-I (Command-Shift-I on Mac) changes to the Invoices view while Ctrl-Shift-P (Command-Shift-P on Mac) changes to the Projects view.
- **Database validation:** When started, Fanurio checks if it can connect to the database to prevent any accidental data loss.
- **Logging:** Fanurio is now configured to keep at most 4MB of logs (previously it used only 0.5MB). In case anything is going wrong with it, we'll have lots of information to investigate and fix the problem.

Bug Fixes

- The fanurio.lck file that is used by Fanurio to know if it was already started is no longer created in the user's home folder but in the settings folder. Fanurio fails to create this file on some Macs.

6.22. Version 1.11 (July 29, 2008)

This version makes possible to track costs and profits. Here's a list with all the changes:

New Features

- **Costs and profits:** Project items can now track both the purchase cost and the selling price (rate). Tracking costs is very useful especially if you subcontract or resell some of your services. By recording the cost, you can get accurate reports of your profits.
- **Money reports:** Project reports can now show both time and money.
- **Start recent:** The Start Recent list contains the latest items where time has been recorded. It can be accessed from the Timer menu, the tray menu, the toolbar and the iTunes-like mini timer. By clicking on one of the items, the timer is started for that item.

  The recent items are grouped by client and project for easier access.
- **Payment date for invoices:** You can now specify the date when an invoice is marked as paid.
- **Check for Updates:** Fanurio can now check automatically for updates. This option is enabled by default and can be disabled from the Preferences dialog.

Improvements

- **Money totals:** The items table footer displayed the total time recorded for the project but now it also displays the total money earned (profit) for that project. To easily track profits and taxes, the invoices table footer displays totals too.
- **Reports location and format:** Fanurio now remembers where reports have been exported and the format used to export them. This can save you a few clicks every time a report is exported.
- **Report configurations:** If you use project reports frequently then you can create your own report configurations. These configurations will remember the report period and the other settings so that you don't have to specify them again each time.
- **Expand/collapse clients tree nodes:** The clients tree nodes are now collapsed by default. You can expand or collapse them from the popup menu at the top of the clients tree.
- **Suspend auto-numbering:** If auto-numbering is enabled, it can be disabled when an invoice is created. This is helpful if you create and delete an invoice several times in a row. You may not want the counter to be increment since you are creating the same invoice.
- **Preview invoices:** Invoices can be previewed right in the "Create Invoice" dialog. You can now see how an invoice looks like without actually creating it.

Bug Fixes

- Idle time was reported inaccurately on some Macs.
- Doesn't select all contacts when importing them from Address Book.
- The Start (F5) timer action didn't start the timer for the current item.
- The link that indicates an item in the Time Entry dialogs doesn't wrap if there isn't enough space.
- Table columns weren't always sorted correctly.
- Auto-backup is enabled by default.

- Clients were not initialized correctly when created from the New Project dialog.
- Duplicate taxes appeared in tax groups when created but they were discarded once the application was restarted.

6.23. Version 1.10 (April 22, 2008)

This version adds smart timing, better Linux integration and many other improvements. Here's a list with all the changes:

New Features

- **Runs on Linux:** Fanurio now integrates with and runs on Linux. We've tested the application on Ubuntu 7.10 with Sun Java 1.5.

  Although a Linux version was available right after we released version 1.9, this release also has a Debian installer and can integrate with the tray. You can read more about the Linux version on our blog [http://ateliersoftware.wordpress.com/2008/02/19/fanurio-runs-on-linux/].

- **Smart Timing:** Smart timing is how Fanurio figures out what you are doing in order to record time accurately. It uses idle time detection and a set of reminders to do that.

  Until now, Fanurio had only idle time notification and one reminder to start the timer. We've added two more reminders, one to resume the timer if it is paused for too long and one to stop it. You can read more about smart timing in this section.

  We've also made the start timer reminder dialog easier to use.

- **New Timer menu:** We've added several new timer actions (start, discard, transfer, recover and timer info) and so we've decided to create a Timer menu to hold all timer actions. The new actions make time tracking more flexible.

  - **Start...** lets you start the timer for a specified project item and attach a description for the task that you are doing. When the timer is stopped, Fanurio will add time to this item with the specified description. The other **Start** action will simply start the timer without asking where time will be added.

  - **Discard** lets you discard some of the time recorded by the timer. Lets assume the timer is running for 30 minutes but for 10 minutes you've been on a break. This action lets you discard those 10 minutes.

  - **Transfer** lets you move some of the time recorded by the timer to a project item. Lets assume the timer is running for 30 minutes but for 10 minutes you've been doing something else. This action lets you discard those 10 minutes from the timer and move them to the project item you've been working on.

  - **Info** shows some information about the timer when it is running. It also lets you change the project item and the description that will be used to add time when the timer is stopped.

  - **Recover** lets you recover the last time interval recorded by the timer that wasn't added to a project. If you start the timer and at some point you stop it but you don't add the recorded time to a project by mistake, you'll be able to recover it. Fanurio remembers only the last time interval that wasn't added to a project.

- **View invoiced item:** Double-clicking on an invoiced item shows its description and the time log. Previously it wasn't possible to view an invoiced item.

Improvements

- **Unicode support:** All exported invoices (HTML or PDF) use the UTF-8 character encoding (Unicode). See this section for more details.

- **Improved Add Time dialog:** The Add Time dialog is used to add time to a project item. Until now, you had to select an item and then add time to it. The new Add Time dialog makes it possible to specify both time and a project item for that time.

You can now use the Business > Add Time menu action to add time to a project item without opening a project first.

- **Copy time:** Added a new Copy action in the Time section of the New Item and Edit Item dialogs. This action lets you duplicate a time entry.
- **Replaced the Assign Time dialog:** The Assign Time dialog that was used to assign time to a project item when the timer was stopped is no longer used. We are using the new Add Time dialog instead.

  This approach is better for two reasons. 1) we are using the same dialog to assign time whether it is manually or if the timer is stopped and 2) the new Add Time dialog makes it possible to assign time to new items, projects or clients.
- **Redesigned the idle notification dialog:** The idle notification dialog had too many options so we've made it simpler. If you are away from the computer for a while and it becomes idle, Fanurio will ask you what you've been doing while away. You have three options:

  1. You've been on a break (coffee, for instance). In this case you should tell Fanurio to **discard** the away time. This means the idle time will be considered pause time and it will be discarded from the timer.
  2. You've been working on the same thing but not on the computer. In this case you should tell Fanurio to **keep** the away time.
  3. You've been working on something else (a client called and you had to suspend what you've been doing, for instance). In this case you should tell Fanurio to **transfer** the away time to some other project item.
- **Timer keyboard shortcuts:** The start, pause/resume and stop timer actions no longer have the same keyboard shortcuts. The following table shows the new shortcuts for the timer actions. The new shortcuts will hopefully be easier to remember since F5 is usually used to start something (see Microsoft Powerpoint).

  For more details about the keyboard shortcuts available in Fanurio, please read this section.

Table 6.1. Timer Keyboard Shortcuts

Action	Old Shortcut	New Shortcut
Start	CTRL-D	F5
Pause/Resume	CTRL-T	F6
Stop	CTRL-D	F7

- **Items table sorting:** The items table can remember the sorted column between successive launches of the application.
- **Changed Window menu on Mac:** Moved the Zoom action from Window > Zoom to View > Switch View for two reasons.

  First, it wasn't obvious that Zoom switches to the iTunes-like mini timer although that's where iTunes puts it and second, the Zoom action should do something else for a Mac application.
- **New invoice date field:** The date of the invoice can be specified when an invoice is created. Until now, Fanurio created an invoice only for the current date.
- **New item name:** In the New Item dialog, one can double click the name field to select the name of one of the other project items for the new item. Use this feature if you want to create a new item with a name similar to the other items.
- **Default service:** You can define a service as default in the Services and Rates section from Business » My Business Details.

- **Create similar project:** You can create a project similar to an existing one if you right click on it in the projects tree and select New Similar Project from the popup menu. When you create a similar project, Fanurio will copy all the items without their time log.

Bug Fixes

- Not displayed correctly when using multiple screens.
- Fixed Mac OS X 10.5 bug that prevented Fanurio from starting.

6.24. Version 1.9 (February 4, 2008)

This version adds many features and improvements that make Fanurio even more easier to use. Here's a list with all the changes:

New Features

- **More project reports:** The current project reports have been improved and others have been added.

  Project reports can be accessed from the Reports menu. Users can now see project time by date, project, week, month or year.

- **Import Address Book contacts on Mac OS X:** Fanurio can import contacts from Address Book. Go to File > Import > Address Book Contacts to choose which contacts should be imported as clients in Fanurio.

  You can also synchronize your Fanurio business contact information with that from Address Book. Go to Business > My Business Details > Contact and click the Import from Address Book button. Your Address Book contact information will be copied to Fanurio.

  Fanurio can now import contacts from Address Book and from a CSV (comma-separated values) file. All major applications (especially e-mail applications) that keep a list of contacts can export them to CSV.

- **Detect idle time on Mac OS X:** Idle time detection happens when you start the timer but at some point you must leave the computer. This option is not enabled by default and must be configured from Preferences > Timer.

  Until now, idle time detection only worked on Windows. Now, it is also available for Intel and Power-PC Macs that run Tiger (Mac OS X 10.4) or Leopard (Mac OS X 10.5). Previous versions of Mac OS X don't have a reliable method to detect idle time.

- **Move time entries:** Time entries can now be moved from one item to another. To do it, open the edit dialog for the item that contains the time entry, go to the Time section and then use the Move button to move it to another item.

- **Filter project items:** The list of items from a project can be filtered to show invoiced items, uninvoiced items or both. This feature is useful if your project has many invoiced items and you want to hide them.

  To specify which items should be displayed, click on the top-right button from the items table and select from the popup menu one of the options: All Items, Invoiced or Uninvoiced.

- **Purchase order number:** Every time an invoice is created, it is associated with a contract. Fanurio lets you specify both your contract number and your client's number. The client's number is usually a purchase order number.

  A new set of placeholders can be used to access these values from an invoice template. Read this section for more information.

- **Foreign clients:** Many freelancers work with both domestic and foreign clients. When working with foreign clients, they may want invoices in their own currency with numbers displayed according to their country rules. For instance, some countries use comma as decimal separator while others use a dot. Dates may also be formatted differently. Some countries display the month first while others display the day of month.

  Fanurio lets you specify if a client is domestic or foreign. If clients are foreign, you can also specify the client's country so that when invoices will be created for them, numbers, dates and currency will be formatted correctly.

Let's say that you are from Scotland and most of your clients are from the UK but you also have a few clients from the USA. When dealing with clients from UK, you will be invoicing in GBP while for the American clients the invoices will be in USD. By marking a client as being from the USA, all invoices created for him or her will show numbers, dates and currency formatted using American rules.

Here's how to mark a client as foreign:

1. Go to the projects view
2. Select the client from the projects tree
3. Edit the client (right-click on the tree node and select Edit or double-click on the node)
4. Go to the Billing tab, check the "This is a foreign client..." option and then select a locale to indicate the country

Improvements

- **Preview Invoice:** Invoices are displayed like they will appear on paper, if printed. A black border is used to show the page limits.
- **Add time to item:** You can now add time to an item by specifying only the total time. Select the **none** option from the **Add Time** dialog and you will not be asked to specify when the activity started or ended. This is useful if you are not interested when you've worked during a day.

  The Add Time dialog also remembers how users like to enter time: by start time, by end time, by both or none.
- **Minimize to tray bar on Windows:** The Windows version of Fanurio can be configured to minimize to the tray bar from Tools > Options. By default, it is minimized to tray if the application window is closed. Other options are: **never**, **on minimize** and **on minimize or close**.
- **Higher precision on tax calculation:** Taxes are calculated using a 4 decimals precision.
- **Delete older backups:** If auto-backup is enabled, you can also tell Fanurio how many backups to keep. The older ones will be automatically deleted.

  To specify how many backups to keep, go to Preferences (on Mac) or Tools > Options (on Windows) and enter a number.

Bug Fixes

- When exporting invoices to PDF, some fonts where not rendered. To fix this problem, copy the font file to the templates folder.

  This folder can be accessed from the menu File > Show Templates.
- Fanurio crashed on some Macs because it didn't update the dock icon badge correctly.
- Fanurio failed to display invoices if it was set to use the first template from the list of templates.
- Fanurio failed to start if it used a language only locale.
- Failed to display images when viewing invoices.
- The today date from the status bar and the date filters were not updated if the application ran after midnight.
- The View menu was not synchronized correctly.

- Spinners didn't always accept new typed values on Mac OS X.

6.25. Version 1.8 (November 12, 2007)

The most important feature of this version is the discounts feature. Users can discount individual items or the whole invoice. Here's a list with all the changes:

New Features

- **Discounts:** With Fanurio you can discount each project item or you can discount the total invoice value. In either case, invoices can show both the discounted values and the regular ones. Discounts can be specified as a percentage or as a fixed value.

- **Time report for projects:** A time report can be displayed for each project. It shows the time recorded for each item and the total time recorded for the whole project.

  There are two ways to display the time report of a project. You can either go to the projects section, right-click on a project and select "Show Time Report" or you can click the link label under the items table that shows the total recorded time.

- **Export invoices directly to PDF:** Invoices can now be exported directly to HTML and PDF. Previous versions allowed HTML export only.

  Read this section to learn how to customize an invoice template to export nicely to PDF.

- **Drag items to other projects:** The items of a project can be dragged to a different project in the projects tree. Dragging an item can either copy or move it to the new project. Invoiced items can only be copied. They cannot be moved as that would alter existing invoices.

  A simple mouse drag represents a move operation. To copy an item, you have to press the control key on Windows or the option key on Mac and then drag the mouse.

- **Export the list of invoices and expenses:** The list of invoices can be exported to Excel and CSV. Just click on the top-right button from the invoices table and it will show a popup menu that allows to make this export.

  This operation is useful if you want to create tax reports outside Fanurio.

- **Use custom template to view invoices:** If you have a custom invoice template, you can use it to both export and view an invoice. This makes it easier to view an invoice before it is exported to HTML or PDF.

  The previous version restricted invoice viewing to a predefined template only.

- **Reminder to start the timer:** There's a new option available that let's you configure Fanurio to remind you to start the timer if it wasn't started lately. This option is not enabled by default.

- **Region setting to indicate format for numbers, currency and time:** The new region setting allows you to specify how data should be formated in Fanurio. Fanurio detects your region automatically from your computer settings but in some cases this information is not available.

  If Fanurio doesn't display numbers, currency or time in your format, make sure you change its region setting.

Improvements

- **Projects popup:** The section that contains the clients and projects tree has been renamed from Clients to Projects. The popup that controls which projects are visible has been placed to the right to make it more obvious.

  The popup contains a new command that highlights which projects have been invoiced.

- **Specify a time entry in different ways:** Previous versions allowed to specify only the start time and the duration of a time entry. This version makes it possible to also define a time entry by its start time and finish time or by duration and finish time.
- **Improved idle notifier:** The idle notifier gives you more choices for handling idle time.
- **New time log filter:** The time log can now show or hide billable time. To access the time log, click on the today date from the status bar.
- **New placeholders:** Added new placeholders like: project.notes, client.code, client.website, business.website and invoice.taxable.
- **New templates extension:** The template files now have the ftl extension instead of itl. The old extension conflicted with iTunes.
- **New website field:** A website field has been added to both business and clients.
- **Mac OS X Window menu:** Added a Window menu for Mac OS X to contain the Minimize Window and Zoom commands. The Zoom command changes the main window to the iTunes-like mini timer.

Bug Fixes

- **Crash recovery:** The time wasn't recovered correctly if the application crashed and the timer was running
- **Same name clients:** The application failed to start if there were two clients with the same name.
- **Time format:** Time is displayed using locale specific settings. In some countries it is displayed as "02:30 PM" while in others as "14:30".
- **View invoice:** The view invoice operation failed when one of the invoice fields contained invalid XHTML characters like &, < or >. The bug fix escapes such characters so that any invoice field can contain any characters.
- **Mac OS X bugs**
  - The application hanged on some computers when the timer was started due to a font problem
  - The main window didn't move correctly
  - The time field editor didn't validate the values correctly due to a focus problem
- **UI bugs**
  - No mnemonics or accelerators in the clients tree popups
  - In the project view, moved the project toolbar under the project title. If the project title was too long, the main application window didn't resize properly.
  - The tree node editor was enabled when clicking on a node from the clients tree. The editor should start only when pressing F2 on a node.
  - Deleting all time entries of an existing item didn't reset its time to zero
  - Time was not updated in the items table when the timer was stopped

6.26. Version 1.7 (September 3, 2007)

This version brings Mac OS X integration and an improved timer. To make time tracking easier, we've added a pause button and an iTunes-like mini timer. The mini timer is a small window that lets you control the timer easier. Here's a list with all the changes:

New Features

- **Runs on Mac OS X:** Fanurio now integrates with and runs on Mac OS X. We've tested the application on Mac OS X 10.3.9 with Java 1.4 but it should also work with later versions too.

  Fanurio has many integration features to make it look and feel like a Mac application but one that you will notice is the badged dock icon. The dock icon is badged to show the elapsed time down to the minute when the timer is running or it is paused.

- **Mini timer:** The mini timer window gives quick access to the timer controls and you can also see how much time has elapsed since it was started. You can switch to the mini timer using the View > Switch View command from the menu bar.

  On Windows, you can also switch the view from the tray icon menu.

- **New pause button:** The pause button can be of great help when you start working on something and must take frequent breaks. When the timer is paused, the mini timer display blinks.

  On Windows, the tray icon blinks too while on Mac it is the dock icon badge that blinks.

- **Build the invoice template visually:** When it comes to invoice templates, you could create them yourself or you could ask us to create them for you. In case you want to create them yourself, we've just made things easier.

  Just go to the invoices view and click to view an invoice. Once it is displayed, you can configure its appearance by specifying the information you want to be included. Click the Configure Template button to see what options are available, make your choices and then click Apply Settings to see how the invoice looks. You'll have quick visual feed-back on your options.

  Once you've decided on a look, click Save Template to save the template to the templates folder. Later, when you'll want to export an invoice to HTML, you can use this template.

  Note: Previous versions had a New Template dialog that helped to create an invoice template. It has been replaced by the more intuitive visual method.

Improvements

- **Show the time log for a date range:** The time log dialog is displayed whenever you click the today date from the status bar. In previous versions, this dialog only showed the log for a single date. This version can also display the time log for a date range. You can choose a predefined date range like **This Week**, **This Month**, **Last Week**, **Last Month** or you can specify a custom date range.

  In addition to that, the time log can be exported to HTML, CSV and Excel. The previous version only exported to HTML.

- **More detailed invoice previews:** The view invoice dialog has been redone visually to display the invoice contents more intuitively. The new view also shows when time has been recorded for each item by date.

- **Export the application log file:** If the application is not working correctly, you can always export its log file to a folder of your choice and then send it to us for further investigation.

  Go to Help > About Fanurio to display the about dialog and then click the **Export Log File...** button.

- **Other improvements**
  - New timer icons
  - Renamed Generate to Export: The old Generate Invoice command has been renamed to Export as we already use this term in a few other places. We want to keep command names consistent across the application.
  - Larger address field: The address field has been replaced with a multiline address field.

Bug Fixes

- **Invoice date filter:** The date filter was not accurate and invoices created before noon were not displayed.
- **Wrap text in text areas:** The editable text areas (like project description, item description, etc) wrap the text when the line is too long.

6.27. Version 1.6 (July 2, 2007)

The most important features of this version are a new timer that replaces the previous one and support for marking invoices as paid. Here's a list with all the changes:

New Features

- **New timer:** This version of the application replaces the old timer with a more usable one. The main drawback of the old timer was that it required to select a project item first in order to start it. This approach wasn't very intuitive or usable.

  The new timer can be started **anytime** (you don't have to select an item first) and from **anywhere** (menu, toolbar, tray menu, shortcut). When the timer is stopped, the user is asked to specify the item for which the time is recorded. If none is available, a new item can be created.

  This new timer can also be started or stopped from the tray menu or using the **CTRL**+**T** shortcut.

- **Mark projects as finished or unfinished:** Knowing which projects are finished and which aren't can be of great help when you only want to deal with one kind of projects. If you want to focus on your current work, you can choose to see only unfinished projects. But if you want to go over past projects, you can choose to see only finished projects.

  To change the finished state of a project, right click on a project node from the tree and mark it as finished or unfinished. Please note that finished projects cannot be modified, they are read-only. This means that you cannot create an invoice, add-edit-delete an item, add-edit-delete an expense or record time (manually or using the timer).

  To choose what projects are displayed in the clients tree, click on the Clients label right above the tree.

- **Mark invoices as paid:** Invoices can be marked as paid or unpaid. Unpaid invoices are displayed as overdue if they are not paid on time.

  To help you differentiate between different kinds of invoices, Fanurio paints paid, unpaid and overdue invoices in distinct colors. You can also use a filter to see only one kind of invoices.

- **Invoice search and filtering:** This feature is really helpful when there are many invoices and you want to see only some of them. To do that, you can display invoices from a certain date range, you can search them by number or client name and you can also filter them by their paid status. The paid filter makes it easy to see unpaid and overdue invoices.

  Searching and filtering can be done in the Invoices View, right above the invoices table.

Improvements

- **Keyboard actions for the clients tree:** The clients tree can be controlled using the keyboard. Use:
  - INSERT to add a new project
  - ENTER to edit a client or a project
  - DELETE to delete a project or a client
- **Confirm project delete:** The application will prompt the user when a project is deleted.
- **Log all errors:** If the application fails to complete any task during execution, that error will automatically be logged. This helps us identify and fix possible problems faster.

Bug Fixes

- **Vista idle time detection:** Idle time detection is now available for Windows Vista too.

- **Application settings:** The application settings are saved while the application is running so that if the operating system quits unexpectedly, they are not lost.

  This problem occurs on some Windows machines when the OS is shut down while Fanurio is running. Fanurio doesn't have the time to save its settings. This is more an OS problem than a Fanurio problem and that's why we are trying to save the settings before it is closed.

- **User guide fixed for Internet Explorer:** The html pages from the user guide didn't look properly in Internet Explorer.

6.28. Version 1.5 (May 28, 2007)

The most important features of this version are time logging support for project items and more billing options like invoice auto-numbering and time rounding. Here's a list with all the changes:

New Features

- **Time log for each item:** Up until this version, Fanurio could only remember how much time was recorded for an item. It couldn't remember when that time was recorded. For instance, if I recorded 4 hours for an item during a week, Fanurio was only able to say that 4 hours were recorded. It wasn't able to say when (exact day and time) that time was recorded.

  Knowing exactly when time was recorded for an item can be of great help when it comes to creating time reports. The first report that we are adding is that of a daily time log. This report is useful if you want to have a quick look at what you've done throughout a day.

- **Time log report:** The application can display a daily time log with a summary of all the time recoded during a day.

- **Time rounding:** The time spent on hourly-rated items can be rounded for billing purposes. To specify a default time rounding rule, go to Business > My Business Details > Billing. All hourly-rated items will use this rule when they are created. Time can be rounded up, down or to the nearest for any time interval.

  For instance, you can tell the application to round time to the nearest 15 minutes interval or to round it up to 6 minutes.

- **Invoice auto-numbering:** Fanurio can number invoices automatically. This option must be enabled from Business > My Business Details > Billing.

  If you use global numbering, the invoice counter will be incremented with each invoice you create. But you can also number invoices for each client separately. If this option is enabled, the invoice counter of each client is incremented only when an invoice is created for them.

  You can also define a format for the invoice number that can include besides the counter, the client code, the client name, the year or the date.

- **Added due date for invoices:** For each invoice, one can specify the number of days they are due. To make things easier, a default value can be specified in the business configuration dialog. To set a default value go to Business > My Business Details > Billing.

Improvements

- **Responsive user interface:** The application no longer leaves the impression that it freezes on some operations.

- **Same number invoices are not allowed:** The application doesn't allow to create two invoices with the same number because its purpose is to identify only one invoice.

- **Improved currency display format:** Money amounts are displayed according to the system settings.

  Swiss francs centimes are displayed as .-- instead of .00 if the amount of money has zero centimes.

- **New placeholders:** Added a few more placeholders that can help to create better invoice templates:

  - business.taxLiable indicates whether taxes are enabled or not. It can be used to detect whether the business is registered for taxes or not.

  - invoice.dueDays indicates the number of days when an invoice is due.

- **New placeholders to indicate total time:** Added a few more placeholders that can help to create better invoice templates by reporting the total time spent on an item, a project or an invoice. You can use the following placeholders to get the total recorded time in *decimal format*. For instance 1 hour and 30 minutes is represented as 1.50.

  - invoice.billableTimeAsDecimal indicates the total time recorded for an invoice.
  - project.billableTimeAsDecimal indicates the total time recorded for a project.
  - item.billableTimeAsDecimal indicates the total time recorded for an item.

  You can use the following placeholders to get the total recorded time in *hour format.* For instance 1 hour and 30 minutes is represented as 1:30.

  - invoice.elapsedTimeAsHour indicates the total time recorded for an invoice.
  - project.elapsedTimeAsHour indicates the total time recorded for a project.
  - item.elapsedTimeAsHour indicates the total time recorded for an item.

- **Total time displayed for each project:** When a project is opened, you can see its total time displayed in the lower left corner.
- **Total time displayed for each invoice:** When an invoice is viewed from the application, it shows its total time and the total time spent on each project.

Bug Fixes

- **Sorted invoices:** All the columns from the invoices table are now sorted correctly.
- **Sorted items in invoice:** The items and expenses displayed in an invoice are kept sorted by the date they have been created. Previously, their order was random.

6.29. Version 1.4 (April 18, 2007)

This version contains some important features and improvements that it almost deserves to be a 2.0 release. Tax support is perhaps the most important feature because it helps to create invoices with tax information. Here's a list with all the changes:

New Features

- **Tax support:** Fanurio can create complete invoices with tax information. Tax support is not enabled by default, you have to enable it from Business > My Business Details. After you enable it, you can:
  - define one or more taxes that you need to use for your invoices
  - specify a tax number for your business (this is especially useful if you need to include a tax number on an invoice)
  - specify a tax number for a client
  - specify if a client is exempt from taxes or not

  To quickly define your taxes, we highly recommend you to use the **Tax Wizard** that is available from the Business > My Business Details > Taxes section.
- **Manual:** Starting with this version, Fanurio has a manual that covers all its functionality. The quick-start guide from previous versions has been expanded to a full-blown tutorial.
- **Import clients from CSV:** The application can import clients from an external CSV file. If you keep your contacts in an e-mail application, you can first export them to a CSV file (most such applications can do it) and then use the import wizard to add them to Fanurio.

  Go to File > Import to import your clients to Fanurio.
- **Expense attachment:** Each expense can have a file attached to it. This is helpful especially if you have the bills scanned as files and you want to associate them with an expense.

  To open the file attached to an expense or to locate it on disk, right click on the expense to show a popup menu.

Improvements

- **Improved templates support:** Creating or customizing invoice templates is not an easy thing to do. To help you with it, we've added a few actions and some documentation.

  Go to Invoice > Show Templates if you want to quickly access all invoice templates.

  Go to Invoice > New Template if you want to create a template by just checking some options. This is the fastest way to create a template.

  Read the template guide to learn how to create a template from scratch.
- **Improved clients tree:** Projects and clients are kept sorted in the tree to locate them faster. Their names can be edited by pressing the F2 key on the tree node.
- **Nicer user interface:** Each section (clients, projects, invoices) has its own title. For instance, there is a Clients title above the clients tree.
- **Improved new item button:** The new item button displays a popup so that you can quickly specify what kind of service will be used when the new item is created.
- **Autoresize table columns:** Almost all tables have a button in the top-right corner that displays a table popup menu. You can use it to autoresize the columns of the table.

Bug Fixes

- **Remember the selected client:** When a new project was created from the menu or by clicking the toolbar button, it wasn't created for the client that was selected in the tree.
- **Create Invoice dialog scrollbar:** The tree with items to invoice from the Create Invoice dialog didn't show the scrollbar if it had many items.

6.30. Version 1.3 (March 1, 2007)

New Features

- **Backup/Restore support:** The application can create backup copies of its data and then restore them. This new feature will give users more control over their own data since it is very important and valuable.

  The File > Backup and File > Restore commands can be used to create and restore backup copies. When a backup is created, users can give it a name and a description.

  The File > Show Backups command can be used to go directly to the backup folder and manually delete unwanted backups.

New Options

- **Create backup on exit:** If checked, the application will create a backup copy every time it is closed. This setting is not enabled by default.
- **Path to backup copies:** This path is used by the application to determine where backup copies are located.

Improvements

- **Enhanced tables:** The tables used in the application are sortable and the width of their columns is adjustable.

6.31. Version 1.2 (February 12, 2007)

New Features

- **Currency support:** Money is no longer represented as a plain number but as a number plus currency. Usually this is not a problem for people working with a single currency but it is helpful for those who have to deal with multiple currencies.

  When invoicing a project with multiple currencies, users can specify an invoice currency and exchange rates to this currency.

  To manage the currencies that you can use, go to Business > My Business Details and click on the Currency tab.

- **New application data folder:** The application no longer keeps its data in the installation folder but in each user's profile folder. Users can find the data created by the application in *Documents and Settings\<UserName>\Application Data\Fanurio*.

  When the new version is started, it asks if old data should be copied to the new location.

- **New date field:** Added a date field for both items and expenses. Users can specify when a project item has been started and when an expense has occurred.

New Options

- **Run on startup:** If checked, the application will be started automatically when the computer is started. This option is not enabled by default.

  When started automatically, the application doesn't display the splash screen and it is minimized in the tray bar.

- **On idle action:** If the timer is running and users leave the computer, the application can decide what to do if the computer becomes idle. It can either stop the timer directly or ask the user if the idle period should be discarded.

- **Path to invoice templates:** This path is used by the application to load the templates that are used to export HTML invoices. Users can change this path if they want another location than the default one.

  This is also useful because users can quickly find out where the templates are located and add their own.

Improvements

- **Improved project editing:** Projects are easier to edit because they can be opened using a single mouse click instead of two. The application will also display only one project at a time. Tabs are not used anymore to represent multiple open projects.

- **Improved client editing:** Clients can be edited by double clicking on them.

- **New Business menu:** The business menu contains data-related actions. It is from this menu that users can create new clients, projects, start the timer or describe their business.

- **Export expenses to Excel:** When exporting the application data (File > Export), the Excel file has two sheets: one contains all the items and the other all the expenses.

Bug Fixes

- Failed to save invoices sometimes.

6.32. Version 1.1 (January 16, 2007)

This version adds capability to export application data to Excel. Users have now the possibility to export data to both CSV and Excel. Here's a list with all the changes:

New Features

- Application data can be exported to Excel.
- Added a "View Invoice" action; invoices can now be viewed from the application.
- Added a "Check for Updates" action in the Help menu.

Improvements

- The items and expenses tables can be sorted using mouse-clicks.
- The application remembers its window location and position.
- The idle timer is now optional.
- Minimize the application to tray using the ESC key.

Bug Fixes

- A gray window appeared if the application was minimized to tray and an action was triggered from the Quick-Start Guide window.
- Failed to save new invoices.

6.33. Version 1.0.2 (December 5, 2006)

This version fixes a few bugs, most of them related to the CSV export of the application data. Here's a list with all the changes:

- Using comma as the delimiter for CSV export instead of semicolon.
- Exporting all the fields of a project item to CSV.
- Using double quote instead of single quote when exporting to CSV.
- Ask if file should be overwritten when exporting to CSV.
- Project name is saved when the application is closed.
- The running project item wasn't displayed correctly if the project was closed and reopened.
- The timer's state wasn't updated correctly if an item was invoiced or it's invoice was deleted.

6.34. Version 1.0.1 (November 24, 2006)

This version fixes a database bug that prevented the application from creating more project items that used the same service.

6.35. Version 1.0 (November 10, 2006)

Initial release.