



frePPLe

A free Production Planning Library

**USER MANUAL
VERSION 0.2.3
JUNE 2007**

Frepple 0.2.3

This document is made available under the terms of the GNU Free Documentation Licence. See the appendix in this document for details.

You may:

1. make and distribute *verbatim copies* of these pages, provided that the copyright notice and this permission notice are preserved on all copies
2. copy and distribute *modified versions* of these pages under the conditions for verbatim copies, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one
3. copy and distribute *translations* of these pages into another language, under the above conditions for modified versions

This document is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

* * *

CONTENTS

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Features | 2 |
| 1.2 | Architecture | 2 |
| 1.2.1 | Command line application | 2 |
| 1.2.2 | Command line application with embedded Python scripting | 3 |
| 1.2.3 | Excel frontend for the command line application | 4 |
| 1.2.4 | Your C or C++ application links with Frepple | 4 |
| 1.2.5 | Your java/perl/ruby/VB/.NET application accesses the Frepple shared library | 6 |
| 1.2.6 | Django frontend for Frepple | 7 |
| 1.2.7 | Frepple as a service on the network | 8 |
| 2 | Download and install | 10 |
| 2.1 | Installing on Windows | 11 |
| 2.1.1 | Windows Installer | 11 |
| 2.1.2 | Installation of web user interface | 12 |
| 2.1.3 | Compiling under windows | 20 |
| 2.2 | Installing on Linux, Unix and CygWin | 21 |
| 2.2.1 | Build instructions | 21 |
| 2.2.2 | Compiling from the Subversion repository | 23 |
| 2.2.3 | VMware virtual machine | 24 |
| 2.3 | Other platforms | 24 |
| 3 | Modeling | 25 |
| 3.1 | Domain model | 26 |
| 3.2 | Global Parameters | 27 |
| 3.2.1 | Fields | 27 |
| 3.2.2 | Example XML structures: | 27 |
| 3.3 | Buffer | 27 |
| 3.3.1 | Fields | 27 |
| 3.3.2 | BUFFER_DEFAULT | 29 |

| | | |
|--------|-----------------------------------|----|
| 3.3.3 | BUFFER_PROCURE | 29 |
| 3.3.4 | BUFFER_INFINITE | 30 |
| 3.3.5 | Example XML structures: | 30 |
| 3.4 | Calendar | 31 |
| 3.4.1 | Calendar Fields | 31 |
| 3.4.2 | Bucket Fields | 32 |
| 3.4.3 | Example XML structures: | 32 |
| 3.5 | Command | 33 |
| 3.5.1 | COMMAND_LIST | 33 |
| 3.5.2 | COMMAND_LOADLIB | 34 |
| 3.5.3 | COMMAND_SYSTEM | 35 |
| 3.5.4 | COMMAND_READXML | 35 |
| 3.5.5 | COMMAND_READXMLSTRING | 36 |
| 3.5.6 | COMMAND_READXMLURL | 36 |
| 3.5.7 | COMMAND_SETENV | 37 |
| 3.5.8 | COMMAND_IF | 38 |
| 3.5.9 | COMMAND_ERASE | 38 |
| 3.5.10 | COMMAND_SAVE | 39 |
| 3.5.11 | COMMAND_SAVEPLAN | 40 |
| 3.5.12 | COMMAND_SIZE | 40 |
| 3.5.13 | COMMAND_SOLVE | 40 |
| 3.6 | Customer | 41 |
| 3.6.1 | Fields | 41 |
| 3.6.2 | Example XML structures: | 42 |
| 3.7 | Demand | 42 |
| 3.7.1 | Fields | 42 |
| 3.7.2 | Example XML structures: | 43 |
| 3.8 | Flow | 44 |
| 3.8.1 | Fields | 44 |
| 3.8.2 | Example XML structures: | 44 |
| 3.9 | Item | 46 |
| 3.9.1 | Fields | 46 |
| 3.9.2 | Example XML structures: | 47 |
| 3.10 | Load | 47 |
| 3.10.1 | Fields | 47 |
| 3.10.2 | Example XML structures: | 48 |
| 3.11 | Location | 49 |
| 3.11.1 | Fields | 49 |
| 3.11.2 | Example XML structures: | 49 |
| 3.12 | Operation | 50 |
| 3.12.1 | Fields | 50 |
| 3.12.2 | OPERATION_FIXED_TIME | 52 |
| 3.12.3 | OPERATION_TIME_PER | 52 |
| 3.12.4 | OPERATION_ALTERNATE | 52 |
| 3.12.5 | OPERATION_ROUTING | 53 |
| 3.12.6 | Example XML structures: | 53 |
| 3.13 | OperationPlan | 54 |
| 3.13.1 | Fields | 54 |
| 3.13.2 | Example XML structures: | 55 |

| | | |
|----------|--|-----------|
| 3.14 | Problem | 55 |
| 3.14.1 | Types | 55 |
| 3.14.2 | Fields | 56 |
| 3.15 | Resource | 56 |
| 3.15.1 | Fields | 57 |
| 3.15.2 | RESOURCE_DEFAULT | 57 |
| 3.15.3 | RESOURCE_INFINITE | 58 |
| 3.15.4 | Example XML structures: | 58 |
| 3.16 | Solver | 58 |
| 3.16.1 | Fields | 58 |
| 3.16.2 | MRP Solver | 59 |
| 3.16.3 | Example XML structures: | 59 |
| 4 | Solver Algorithm | 60 |
| 4.1 | Solver Features | 61 |
| 4.1.1 | Solver | 61 |
| 4.1.2 | Demand | 61 |
| 4.1.3 | Operation | 61 |
| 4.1.4 | Resource | 61 |
| 4.1.5 | Buffer | 61 |
| 4.2 | Implementation details | 62 |
| 4.2.1 | Top level loop | 64 |
| 4.2.2 | Demand solver | 64 |
| 4.2.3 | Buffer solver | 65 |
| 4.2.4 | Operation solver | 66 |
| 4.2.5 | Flow solver | 67 |
| 4.2.6 | Load solver | 67 |
| 4.2.7 | Resource Solver | 67 |
| 4.3 | Cluster and level algorithm | 68 |
| 5 | Modules | 71 |
| 5.1 | Python Module | 71 |
| 5.1.1 | COMMAND_PYTHON | 72 |
| 5.1.2 | Processing instruction PYTHON | 72 |
| 5.2 | Forecast Module | 73 |
| 5.3 | Linear Programming Solver Module | 73 |
| 6 | Developer | 74 |
| 6.1 | Code structure | 74 |
| 6.1.1 | Object | 75 |
| 6.1.2 | MetaData | 75 |
| 6.1.3 | Date - DateRange - TimePeriod | 75 |
| 6.1.4 | Timer | 76 |
| 6.1.5 | Exception | 76 |
| 6.1.6 | XML Serialization | 76 |
| 6.1.7 | Command | 77 |
| 6.1.8 | Mutex and LockManager | 77 |
| 6.1.9 | HasName and Tree | 77 |
| 6.1.10 | HasHierarchy | 77 |
| 6.1.11 | Leveled | 77 |

| | | |
|----------|---|-----------|
| 6.2 | Class diagram | 79 |
| 6.3 | Extension modules | 81 |
| 6.4 | Portability | 82 |
| 6.5 | Version control | 82 |
| 6.6 | Style guide | 82 |
| 6.7 | Security | 83 |
| 7 | Samples and tests | 84 |
| 7.1 | Test Callback | 85 |
| 7.2 | Test Cluster | 85 |
| 7.3 | Test Command 1 | 85 |
| 7.4 | Test Command 2 | 85 |
| 7.5 | Test Command 3 | 85 |
| 7.6 | Test Constraints Leadtime 1 | 85 |
| 7.7 | Test Constraints Material 1 | 86 |
| 7.8 | Test Constraints Material 2 | 86 |
| 7.9 | Test Constraints Material 3 | 86 |
| 7.10 | Test Constraints Resource 1 | 86 |
| 7.11 | Test Constraints Resource 2 | 87 |
| 7.12 | Test Constraints Resource 3 | 87 |
| 7.13 | Test CSV | 87 |
| 7.14 | Test Datetime | 87 |
| 7.15 | Test Deletion | 87 |
| 7.16 | Test Demand Policy | 87 |
| 7.17 | Test Demo 1 | 88 |
| 7.18 | Test Forecast | 88 |
| 7.19 | Test LP Solver 1 | 88 |
| 7.20 | Test Name | 88 |
| 7.21 | Test Operation Effective | 88 |
| 7.22 | Test Operation Pre Op | 89 |
| 7.23 | Test Pegging | 89 |
| 7.24 | Test Python 1 | 90 |
| 7.25 | Test Python 2 | 90 |
| 7.26 | Test Problems | 90 |
| 7.27 | Test Procure 1 | 90 |
| 7.28 | Test Safety Stock | 90 |
| 7.29 | Test Sample Module | 91 |
| 7.30 | Test Scalability 1 | 91 |
| 7.31 | Test Scalability 2 | 91 |
| 7.32 | Test Scalability 3 | 91 |
| 7.33 | Test Sizeof | 92 |
| 7.34 | Test XML | 92 |
| 7.35 | Test XML Remote | 92 |
| 8 | Appendices | 93 |
| 8.1 | GNU Lesser General Public License | 93 |
| 8.2 | GNU Free Documentation License | 100 |

CHAPTER

1

Introduction

Frepple aims at building a lightweight open source framework that easily and quickly delivers a solution for production planning problems. The initial focus is on small to medium sized planning problems for the discrete manufacturing industry.

Production planning software traditionally has been an area with plenty of home-grown, extremely specialised and/or very primitive solutions.

Strangely enough, while creative and innovative open source solutions pop up in all computing areas, production planning software still tends to be a very closed world full of academic, proprietary and expensive solutions. Till now. . .

Frepple is the first open source production planning toolkit for your day-to-day planning problems.

For the developer community, the project is also trying to establish a common ground framework for planning applications. Rather than rebuilding the basic foundation from scratch over and over again, developers can now leverage a proven framework to extend with their own extension modules.

New workflows and functionality can now be built much quicker and easier.

The word “free” in the project name refers to liberty, not price. Think of “freedom of speech” rather than “free beer”: see *the free software definition*¹.

The software is still young. The functionality and API are enhanced very quickly.

Plenty of feedback is required at this stage to move the project forward. So, browse the site, have a look at the software and please mail *us* [info@frepple.com] your feedback.

While there’s plenty of work left, the software is quickly getting into a state where real-life problems can be handled efficiently.

1. Features
2. Architecture

¹ www.gnu.org/philosophy/free-sw.html

1.1 Features

Key features of Frepple are:

- **Modeling and solving framework for discrete manufacturing environments.**
Key modeling constructs are:
 - Item
 - Buffer
 - Resource
 - Operation
 - Demand
- **Heuristic “MRP-like” solving algorithm** respecting capacity, material and leadtime constraints.
- **XML-based data input and output.**
- **Extensible and customizable architecture.**
New modeling constructs and solving algorithms can be developed in C++ and loaded as a plugin module.
- Frepple comes as a **‘library’ developed in C++.**
It has no graphical user interface as such and requires to be deployed in a ‘application’.
Different applications are envisioned:
 - Standalone application for use at the command line
 - Microsoft Excel frontend
 - Accessable from programming languages such as java, python, perl or visual basic
 - Can be linked into your own C or C++ application
 - A reference **browser-based user interface** and **database schema** are based on the powerful Django framework.
Both can very easily be extended and customized.
- Embeds **Python** as scripting language.
Allows for flexible and easy integration and customization.
- Supported on **linux and Windows** environments.
- Licenced under the **GNU lesser general public license**.

1.2 Architecture

The Frepple binaries are a collection of shared libraries: a core library frepple.so (frepple.dll on Windows) and an additional shared library for each extension module.

The extension modules are loaded dynamically as plugins by Frepple.

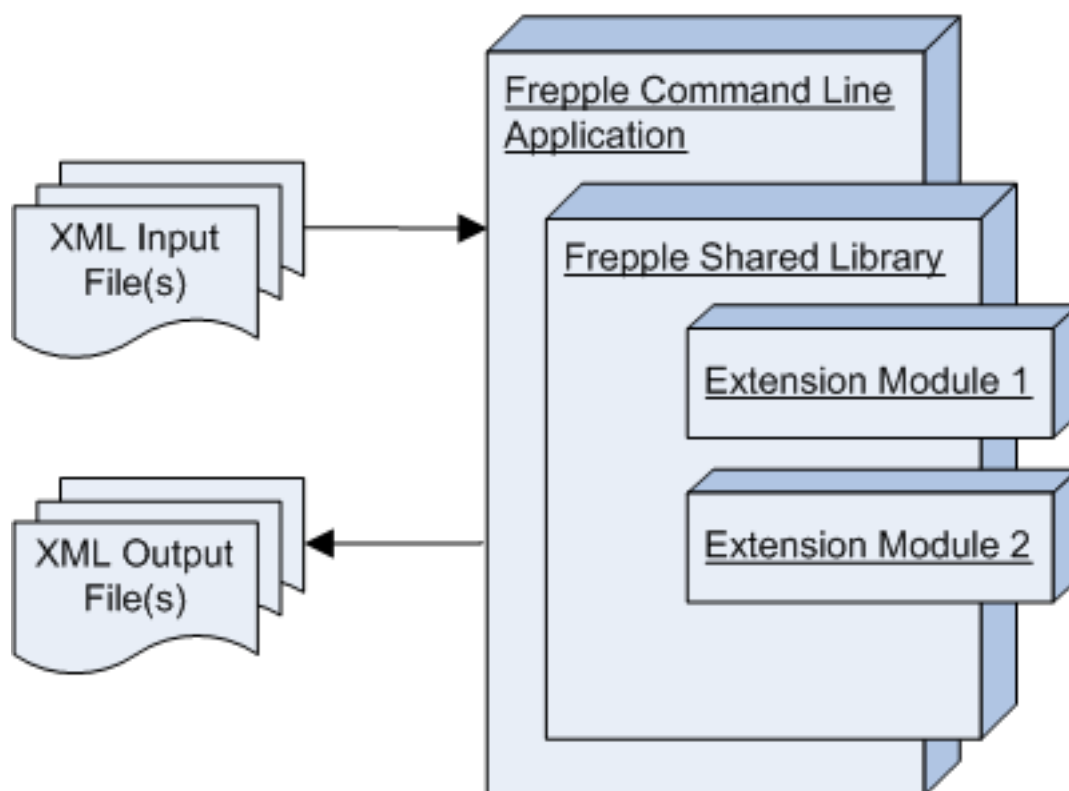
The Frepple shared library can be used in different ways by applications.

Below is a list of some common ways to deploy Frepple, but additional scenarios are definately feasible.

The main development efforts are currently focussed on the first and the last two scenarios.

1.2.1 Command line application

A simple command-line application is available.



The application reads a set of XML files or from the standard input.

It executes all commands defined in the inputdata (which typically involves writing the results back into flatfiles or a database) and then exits.

The program exit code reflects any processing errors.

Example usage:

```
frepple file1.xml
frepple file2.xml file3.xml
frepple dir_with_xml_files
command | frepple
```

Use the option “-help” or “-?” to get a list of possible flags that can be passed on the command line.

This command line application is used for all test cases.

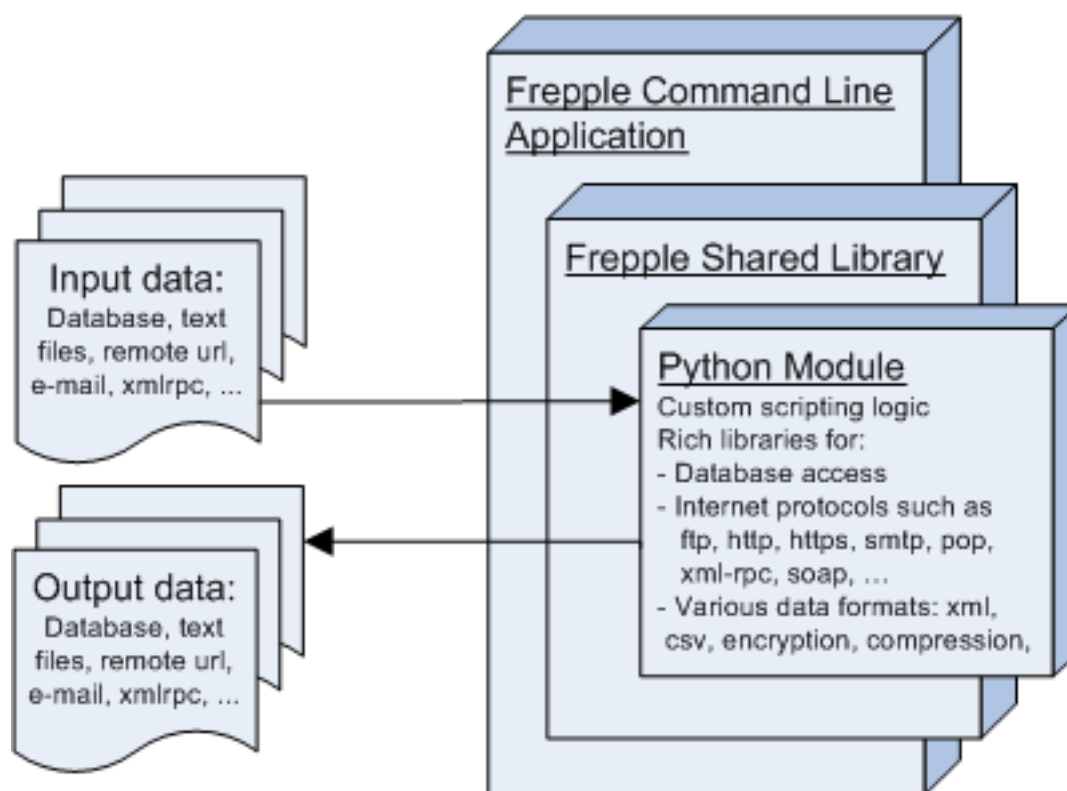
1.2.2 Command line application with embedded Python scripting

In the previous setup the XML input and output files are supplied externally.

Frepple comes with a module that embeds a interpreter for the *Python*² language.

Python is a dynamic object-oriented programming language. It comes with extensive standard libraries for database access, a wide range of internet protocols (such as ftp, http, https, smtp, pop, xml-rpc, soap, ...), various data formats (such as xml, csv, compression, encryption), ...

² www.python.org



Frepple's Python module allows custom logic to be implemented in an easy and flexible way, with full access to the rich standard libraries.

The combination of the rich functionality of the Python library and the direct access to the Frepple objects in memory make this a really powerful combination.

For a majority of applications this will be the most viable solution.

1.2.3 Excel frontend for the command line application

A variation of the first setup worth mentioning is where Excel is as a front-end for Frepple.

The Excel XML-mapping features allow easy generation and manipulation of XML-files. A button in the Excel workbook is used to trigger running the frepple command line application.

A sample of such a spreadsheet is included in the installation.

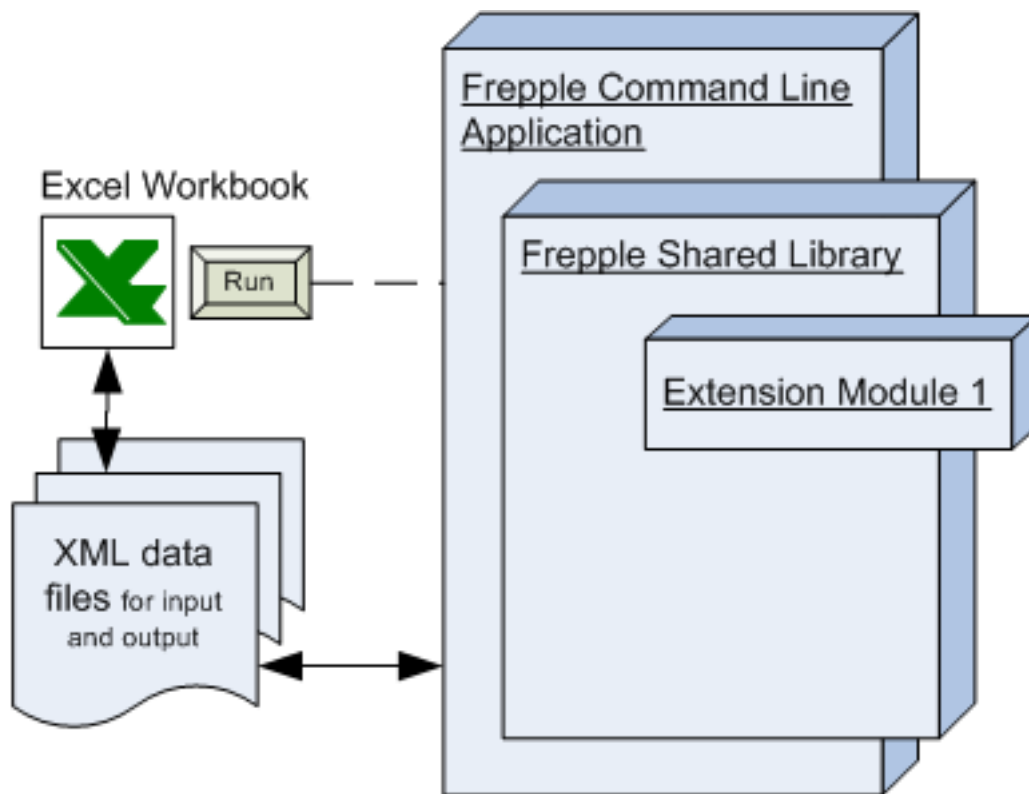
1.2.4 Your C or C++ application links with Frepple

Your application can be link with the Frepple shared library.

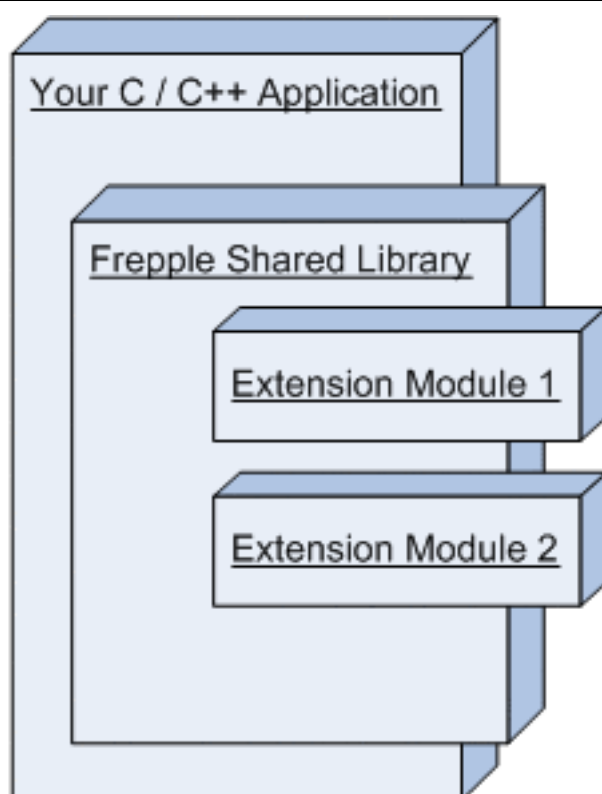
Use the header file `plannerinterface.h` for the high-level interface declarations.

Use header file `frepple.h` when you need low-level access.

Since Frepple is coded in C++:



- C applications will need some wrapper code to catch exceptions correctly and assure C linkage.
- Because of the C++ name mangling Frepple and your application will need to be compiled by the same compiler.



1.2.5 Your java/perl/ruby/VB/.NET application accesses the Frepple shared library

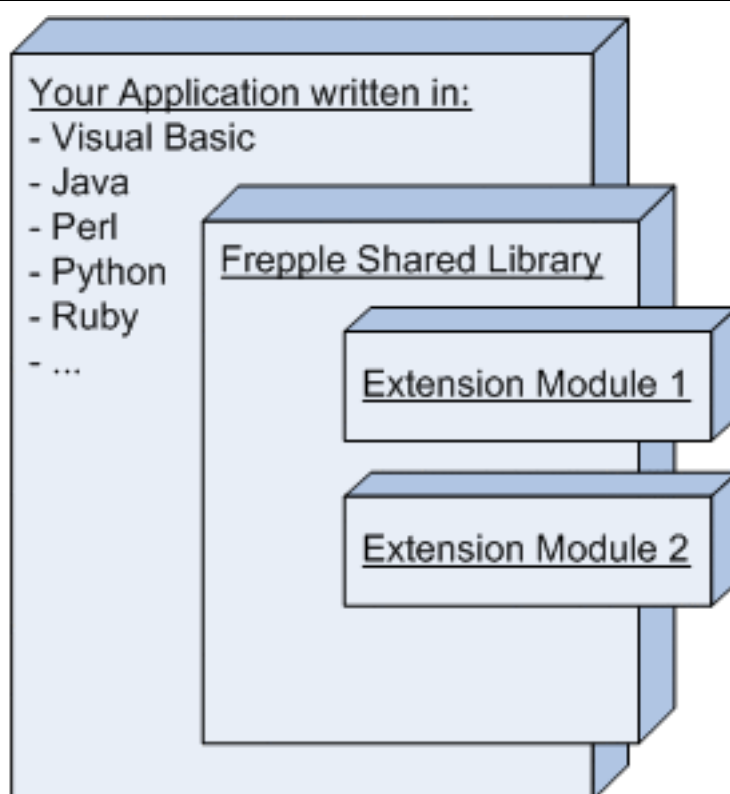
Most modern languages and tools have the capability to access functions in shared libraries.

*SWIG*³ is a tool that can help to generate the integration code with a wide range of high-level languages, such as Java, Ruby, Perl, Tcl, PHP, ...

An example setup is provided in the subdirectory *contrib/scripting*.

When building applications in this way, remember that the scripting language will load the frepple shared library and all memory allocated by frepple (which can be quite a lot!) will be owned by the scripting language process. For large models this is not be a very appropriate integration method.

³ www.swig.org



1.2.6 Django frontend for Frepple

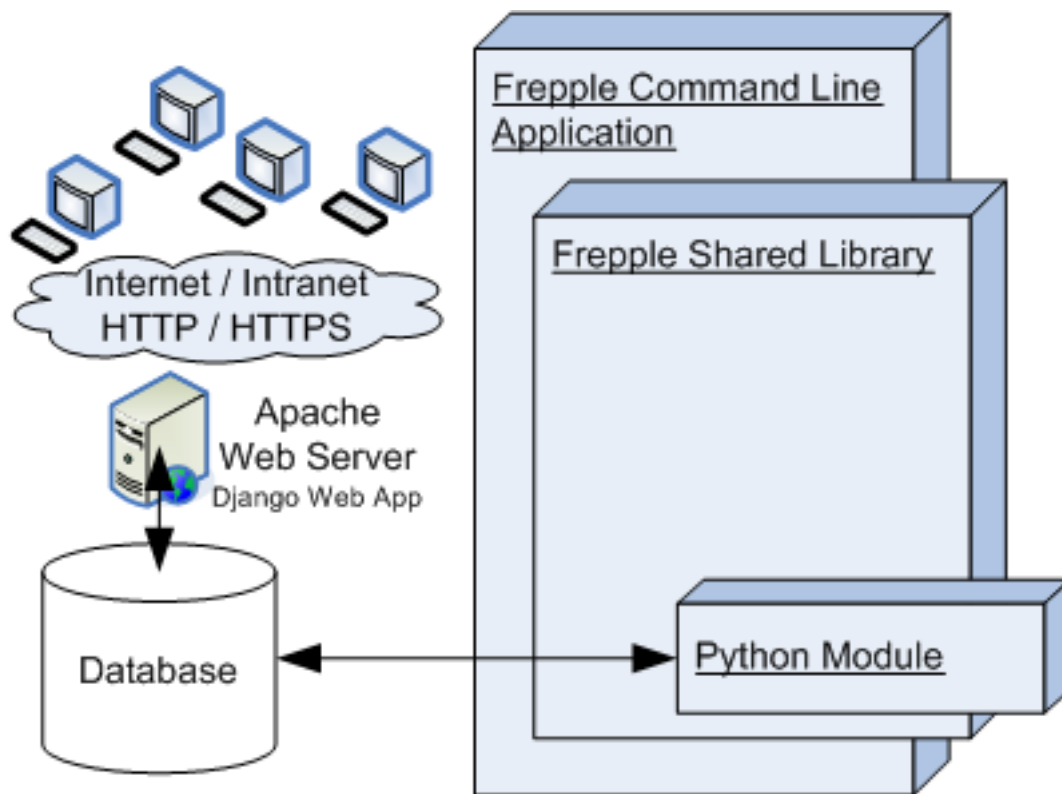
*Django*⁴ is an impressive web application framework written in the Python language. It allows quick and easy definition of the data model, automatically creates a administration user interface and allows you to construct performant and scalable web sites.

Frepple then reads from and writes into this Django database.

The subdirectory *contrib/django* provides a reference Django model for Frepple.

In a real-life implementation you will typically develop your own data model. You'll build web pages to support the user's workflows, and then write the appropriate mapping between your data model and the Frepple internal data structures.

⁴ www.djangoproject.com

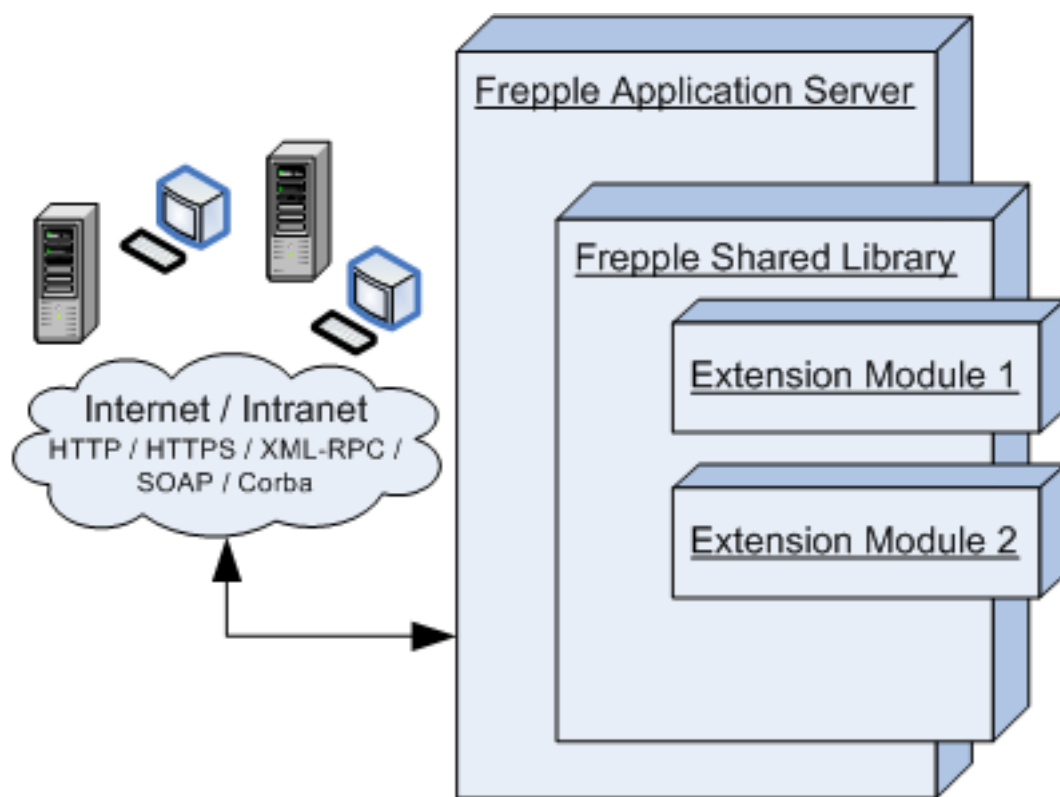


1.2.7 Frepple as a service on the network

A future development will implement a Frepple backend service.

The Frepple server will communicate over the network with other applications, systems or users.

It is not sure yet what technology would be used: SOAP webservice, Corba or XML-RPC.



CHAPTER

2

Download and install

The Frepple project lives on <http://sourceforge.net/projects/frepple> where all release files and the source code are hosted.

The software can be downloaded from http://sourceforge.net/project/showfiles.php?group_id=166214

The project distributes the following formats:

- Installer for 32-bit Windows platforms
- Source code tar-file for all platforms
- A VMware virtual machine with a fully configured demo and development environment on Linux
- Access to the Subversion source code repository for the ‘freshest’ developments

1. Installing on Windows
 - 1.1. Windows Installer
 - 1.2. Installation of web user interface
 - 1.2.1. Installing Python 2.5
 - 1.2.2. Installing PostgreSQL 8.2
 - 1.2.3. Installing Psycopg2 2.0.5.1
 - 1.2.4. Installing Django 0.96
 - 1.2.5. Putting it all together
 - 1.3. Compiling under windows
2. Installing on Linux, Unix and CygWin
 - 2.1. Build instructions
 - 2.2. Compiling from the Subversion repository
 - 2.3. VMware virtual machine
3. Other platforms

2.1 Installing on Windows

The first section describes the installer which provides all documentation, command line application, excel user interface and the development libraries.

A next section describes the installation of the complete Django application stack, which includes python, postgresql database, django and the apache web server.

1. Windows Installer
2. Installation of web user interface
 - 2.1. Installing Python 2.5
 - 2.2. Installing PostgreSQL 8.2
 - 2.3. Installing Psycogp2 2.0.5.1
 - 2.4. Installing Django 0.96
 - 2.5. Putting it all together
3. Compiling under windows

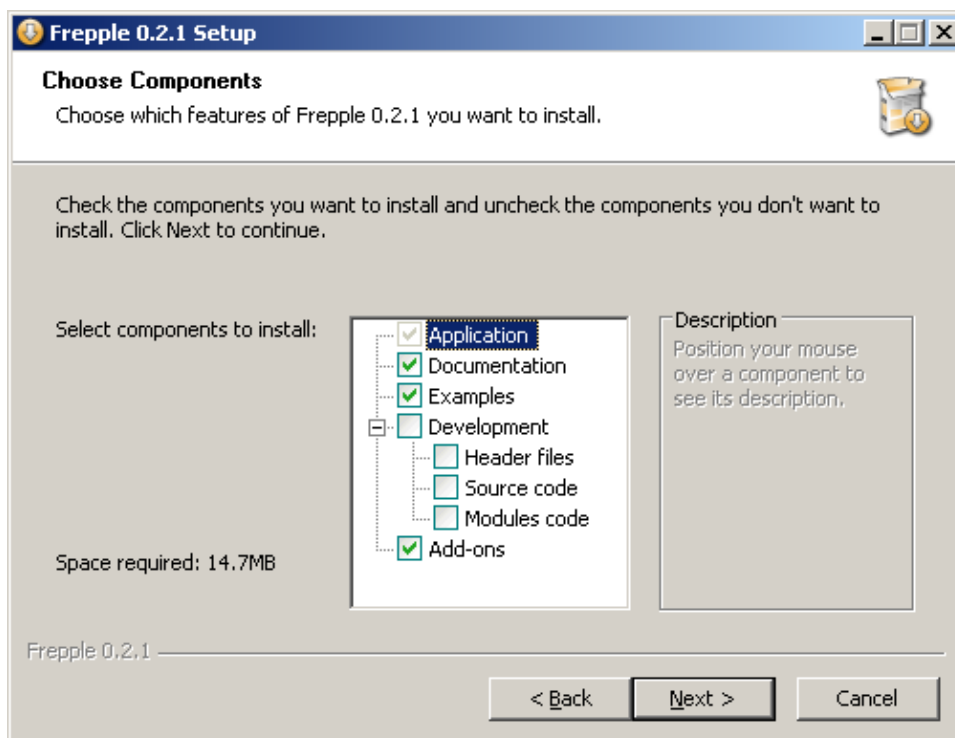
2.1.1 Windows Installer

Installing and uninstalling Frepple is straightforward, and follows the normal Windows conventions.

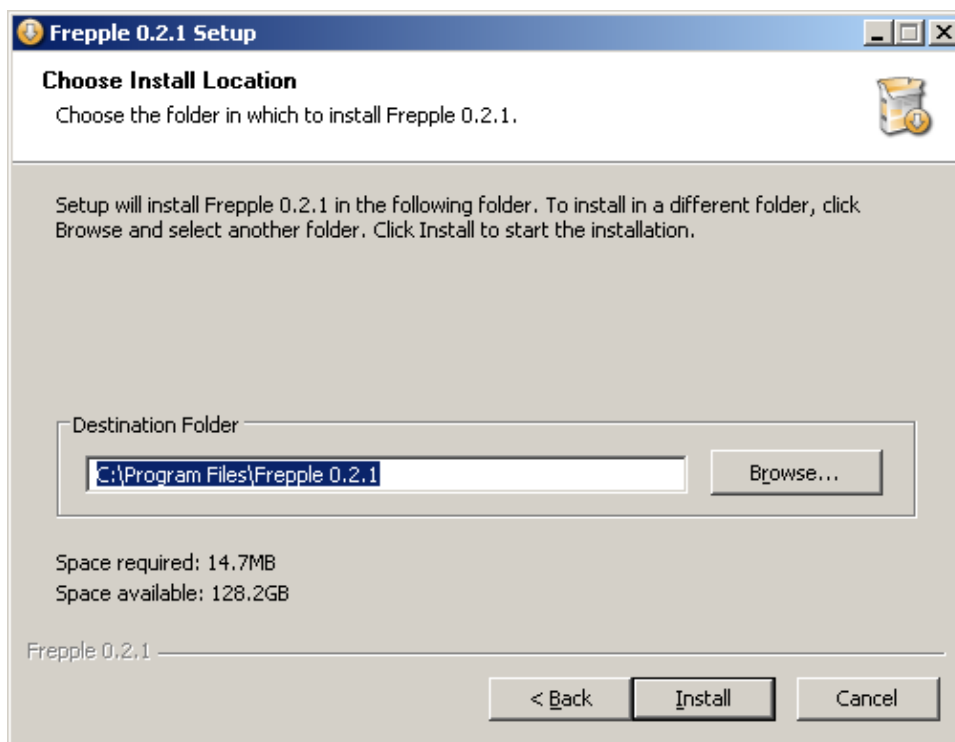
After accepting the licence agreement, the installer will allow you to select:

- The components to install
- The installation directory

With all options included the installation still requires less than 20MB of disk space.



The installer provides all documentation, command line application, excel user interface and the development libraries.



The installer provides **NOT** include the Python language or the django user interface. See the following section on the installation of those components.

It is perfectly possible to have multiple installations in parallel on the same computer. They need to be installed in different directories, and you need to set the environment variable FREPPLE_HOME to point to the directory with the version you want to run.

2.1.2 Installation of web user interface

The installation of the web user interface involves quite a few components. Don't let it frighten you - it's all pretty easy and straightforward...

The web user interface is based on the *Django*¹ framework. The Django installation is documented at <http://www.djangoproject.com/documentation/install/>.

The instructions in this section are only a convenience for first time Django users on windows.

When installing different versions of the Python and/or PostgreSQL than the ones mentioned here, make sure a Psycopg installer is available for your combination.

1. Installing Python 2.5
2. Installing PostgreSQL 8.2
3. Installing Psycopg2 2.0.5.1
4. Installing Django 0.96
5. Putting it all together

¹ www.djangoproject.com

Installing Python 2.5

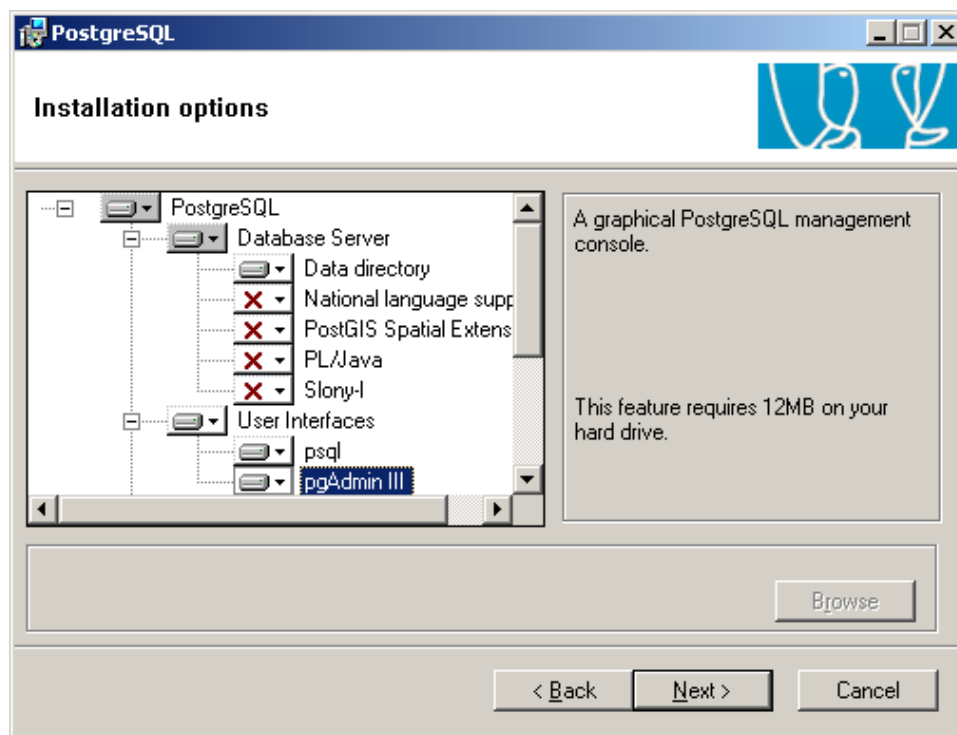
1. Download the installer from <http://www.python.org/download/>
2. Double-click the msi file and follow the instructions.

You will be prompted for an installation directory and can select different components to be installed.

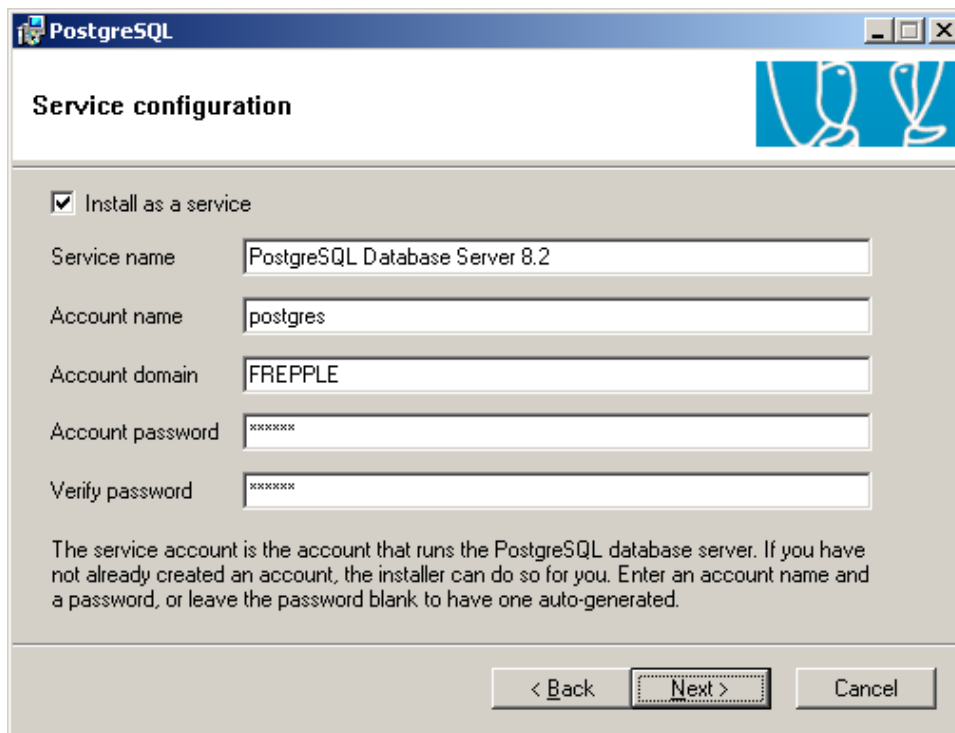


Installing PostgreSQL 8.2

1. Django supports a few different databases including PostgreSQL, MySQL, and SQLite. PostgreSQL is the recommended database by the Django team.
The installers can be downloaded from <http://www.postgresql.org/download/>.
2. Unpack the zip in a temporary folder and you'll have three files (including a README.TXT). Double click the msi file and then walk through the installer. Note that double-clicking the msi from winzip will not work.
Make sure that you have psql and pgAdmin III selected to be installed.



1. When asked if you want to run Postgres as a service say yes. It will automatically start the database and run it in the background so it's there when you need it.
In case you use the database only occasionally and want to keep the boot time of your PC short, you can configure the database service to start manually. This can be done in the windows control panel after the installation has completed.
The installer can automatically create a new user account to run the database service. Otherwise use an existing user.



The screenshot shows the 'Service configuration' window of the PostgreSQL installer. The window has a title bar with the PostgreSQL logo and standard window controls. The main area contains a checkbox labeled 'Install as a service' which is checked. Below it are five input fields: 'Service name' (containing 'PostgreSQL Database Server 8.2'), 'Account name' (containing 'postgres'), 'Account domain' (containing 'FREPPLE'), 'Account password' (masked with 'xxxxxxx'), and 'Verify password' (masked with 'xxxxxxx'). A paragraph of text explains that the service account is the account that runs the database server and that the installer can create one if needed. At the bottom are three buttons: '< Back', 'Next >', and 'Cancel'.

Service configuration

☒ Install as a service

Service name: PostgreSQL Database Server 8.2

Account name: postgres

Account domain: FREPPLE

Account password: xxxxxxx

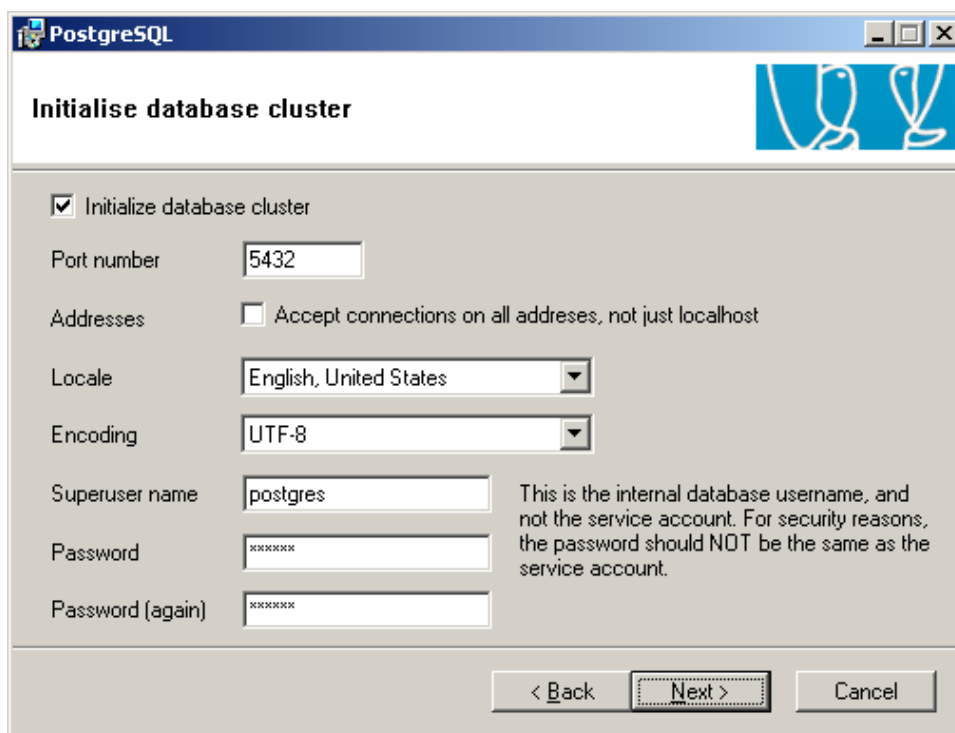
Verify password: xxxxxxx

The service account is the account that runs the PostgreSQL database server. If you have not already created an account, the installer can do so for you. Enter an account name and a password, or leave the password blank to have one auto-generated.

< Back Next > Cancel

1. When configuring your cluster, the default port of 5432/tcp should be fine unless you have a compelling reason to use something else. Select the locale, such as 'English, United States' and 'UTF-8' encoding.

You'll also need to pick your Postgres superuser name and password. This is not the same as the Windows account the service will be running as, and in fact it is not recommended to have the same password for both.



The screenshot shows the 'Initialise database cluster' window of the PostgreSQL installer. The window has a title bar with the PostgreSQL logo and standard window controls. The main area contains a checkbox labeled 'Initialize database cluster' which is checked. Below it are several fields and options: 'Port number' (5432), 'Addresses' (a checkbox for 'Accept connections on all addresses, not just localhost' which is unchecked), 'Locale' (a dropdown menu showing 'English, United States'), 'Encoding' (a dropdown menu showing 'UTF-8'), 'Superuser name' (postgres), 'Password' (masked with 'xxxxxxx'), and 'Password (again)' (masked with 'xxxxxxx'). A text box on the right explains that the superuser name and password are for the internal database and should not be the same as the service account. At the bottom are three buttons: '< Back', 'Next >', and 'Cancel'.

Initialise database cluster

☒ Initialize database cluster

Port number: 5432

Addresses: ☐ Accept connections on all addresses, not just localhost

Locale: English, United States

Encoding: UTF-8

Superuser name: postgres

Password: xxxxxxx

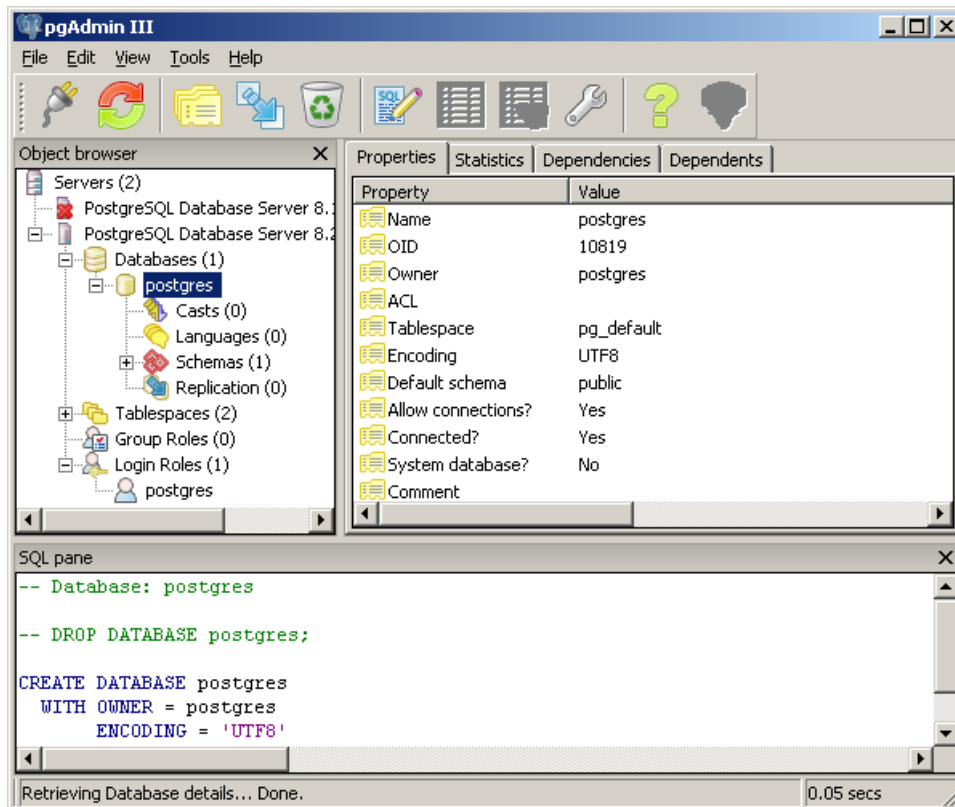
Password (again): xxxxxxx

This is the internal database username, and not the service account. For security reasons, the password should NOT be the same as the service account.

< Back Next > Cancel

1. Click through the final screens and finish. You should now have PostgreSQL installed and running as a service.

To try it out, go to Start | Programs | PostgreSQL | pgAdmin III and run it. If all is well you should be able to double click on 'PostgreSQL Database Server', type your password and see something like the following:

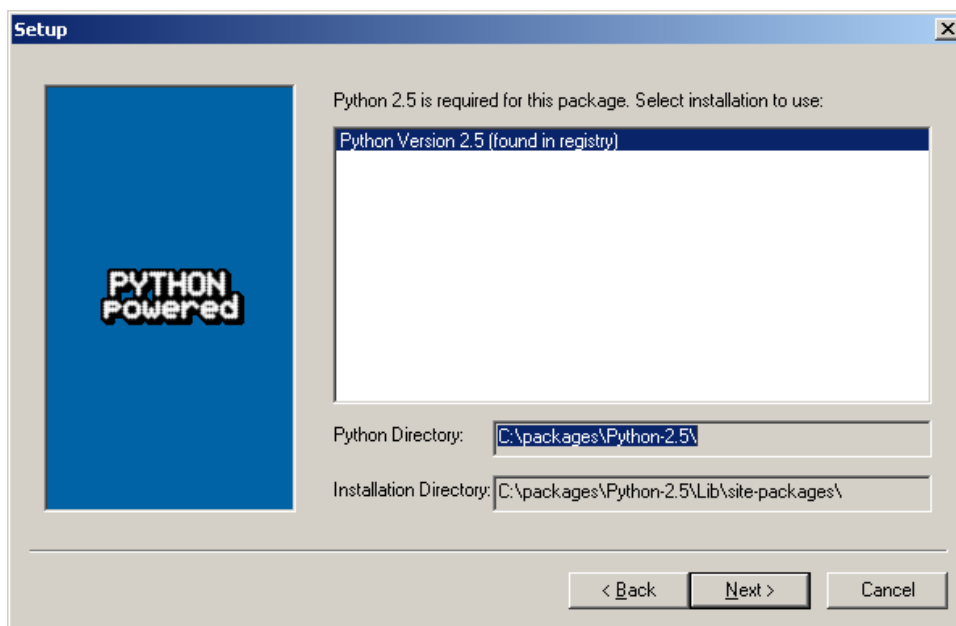


1. For use as a production environment it is highly recommended to create a dedicated tablespace, database and login role for frepple. It assures the best flexibility, eases database maintenance, and allows proper security settings to be applied.

Creating the dedicated entities is easily done with pgAdmin by a database administrator. When creating a new tablespace you need to make sure the user account running the database has full control over the database folder. You may need to use the Windows explorer to update this: right click on the folder, select 'security' and add 'postgres' to the list of users. For now, we'll go with the default user 'postgres' with full permissions on the default database.

Installing Psycopg2 2.0.5.1

1. Psycopg implements an interface to the postgresql database from Python. More information can be found on <http://initd.org/tracker/psycopg/wiki/PsycopgTwo>
Compiled binaries can be downloaded from <http://www.stickpeople.com/projects/python/win-psycopg/index.html>
Make sure that you grab the version that matches both your Python version and your PostgreSQL version!
2. The installer is small and the installation is quick and easy.
It will automatically detect the Python installation folder, and put the library files in the folder Lib/site-packages.



Installing Django 0.96

1. Download the distribution from <http://www.djangoproject.com/download/> and unpack the compressed tar.gz file in a temporary folder. The installation process is described on <http://www.djangoproject.com/documentation/install/> and the steps below are a summary of the steps described there.
If you are familiar with the Subversion version control tool I encourage you to install the development version of Django.
2. In the temporary folder you'll find a subdirectory django.
Copy this folder to the site-packages directory of your python installation. When you installed python in the folder c:\python25 the site-packages directory will be located at c:\python25\Lib\site-packages.
You'll find the psycopg2 library already installed in the same folder.
3. In the temporary folder you'll also find a file django\bin\django-admin.py. Copy this file to the Scripts folder of your python installation. When you installed python in the folder c:\python25 the site-packages directory will be located at c:\python25\Scripts
This step simply lets you type django-admin.py from within any directory, rather than having to qualify the command with the full path to the file.

Putting it all together

With all components installed, it's time to verify the installation and to configure frepple. In this step we update the django settings, create the database schema and start the Django development server with the frepple application.

1. Open a command prompt and navigate to the folder where you installed frepple.
Navigate to the subdirectory **contrib\django\freppledb**.
2. Open the file settings.py in a text editor and edit the following settings. Detailed documentation on all settings can be found on the Django website.
 - **DATABASE_ENGINE** = 'postgresql_psycopg2'
 Leave at the default unless you installed a different database than PostgreSQL.

- **DATABASE_NAME** = 'postgres'
Leave at this default value unless you created a dedicated database for frepple, as .
 - **DATABASE_USER** = 'postgres'
The database superuser as you selected it in step 4 of the database installation.
 - **DATABASE_PASSWORD** = 'yoursecret'
The password of the database user as chosen in step 4 of the database installation.
 - **DATABASE_HOST** = ''
Leave at the default unless the database is running on a different machine.
 - **DATABASE_PORT** = ''
Leave at the default unless you choose a non-default port number for your database in step 4 of the PostgreSQL installation.
 - **FREPPLE_HOME** = os.environ['FREPPLE_HOME']
Edit this line if you the environment variable FREPPLE_HOME doesn't point to a valid bin directory of a Frepple installation.
 - **FREPPLE_APP** = ...
Edit this line to point to directory where Frepple's django application is installed. The default expression builds a path relative from the FREPPLE_HOME directory.
3. Issue the command **.manage.py shell** A command prompt appears and we will type some commands to verify the correctness of the installation.

In a correct setup no error messages will be displayed:

- Type **print "hello world"** to check whether python works fine.
- Type **import psycopg2** to verify the correct installation of the database interface.
- Type **import django** to verify Django is installed correctly.
- Type **import freppledb** to verify the Frepple's Django application.

```
C:\develop\frepple\contrib\django\freppledb>cd C:\develop\frepple\contrib\django\freppledb
C:\develop\frepple\contrib\django\freppledb>.\manage.py shell
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> print 'hello world'
hello world
>>> import psycopg2
>>> import django
>>> import freppledb
>>> ^Z
C:\develop\frepple\contrib\django\freppledb>
```

1. Issue the command **.manage.py syncdb**

This will create the database tables for frepple.

You will also be prompted to create an administrator user. Select 'yes' and provide a user name, email and password. I usually use 'frepple' as user name and password.

When completed you can see the tables in the pgAdmin user interface.


```

C:\develop\Frepple\contrib\django\Freppledb>
C:\develop\Frepple\contrib\django\Freppledb>.\manage.py syncdb
Creating table auth_message
Creating table auth_group
Creating table auth_user
Creating table auth_permission
Creating table django_content_type
Creating table django_session
Creating table django_admin_log
Creating table input_customer
Creating table input_load
Creating table input_dates
Creating table input_resource
Creating table input_suboperation
Creating table input_buffer
Creating table input_flow
Creating table input_bucket
Creating table input_operationplan
Creating table input_item
Creating table input_plan
Creating table input_demand
Creating table input_calendar
Creating table input_operation
Creating table input_location
Creating table output_flowplan
Creating table output_problem
Creating table output_loadplan
Creating table output_operationplan
Creating table execute_execute

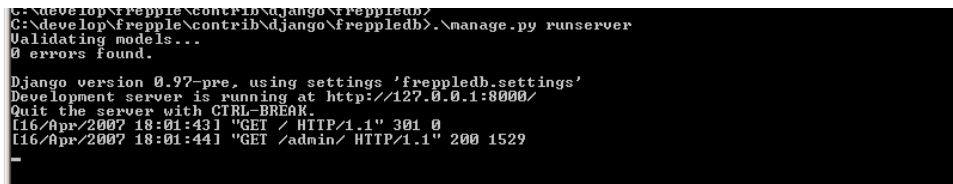
You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username: frepple
E-mail address: frepple@yoursite.com
Password:
Password (again):
Superuser created successfully.
Installing custom SQL for execute.Plan model
Installing index for auth.Message model
Installing index for auth.Permission model
Installing index for admin.LogEntry model
Installing index for input.Customer model
Installing index for input.Load model
Installing index for input.Dates model
Installing index for input.Resource model
Installing index for input.SubOperation model
Installing index for input.Buffer model
Installing index for input.Flow model
Installing index for input.Bucket model
Installing index for input.OperationPlan model
Installing index for input.Item model
Installing index for input.Plan model
Installing index for input.Demand model
Installing index for input.Calendar model
Installing index for input.Operation model
Installing index for input.Location model
Installing index for output.FlowPlan model
Installing index for output.Problem model
Installing index for output.LoadPlan model
Installing index for output.OperationPlan model
Loading 'initial_data' fixtures...
Installing json fixture 'initial_data' from 'C:\develop\Frepple\contrib\django\Freppledb\..Freppledb\input\fixtures'.
Installed 2 object(s) from 1 fixture(s)

C:\develop\Frepple\contrib\django\Freppledb>

```

1. Issue the command `.manage.py runserver`

This command starts a simple web server written in Python. You're now ready and can connect your browser to `http://127.0.0.1:8000/` and use the user created in the previous step to log in.



```
C:\develop\Frepple\contrib\django\Freppledb>.manage.py runserver
Validating models...
0 errors found.

Django version 0.97-pre, using settings 'freppledb.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[16/Apr/2007 18:01:43] "GET / HTTP/1.1" 301 0
[16/Apr/2007 18:01:44] "GET /admin/ HTTP/1.1" 200 1529
```

1. The web server used above is suitable only for development and demoing purposes. In a production environment you'll need to install the Apache web server and the `mod_python` module.

See <http://www.djangoproject.com/documentation/modpython/> for the details of the configuration steps.

2.1.3 Compiling under windows

Different options exist to compile Frepple under windows:

- Microsoft Visual C++ Compiler (p 20)
- Cygwin Compiler (p 20)
- Borland Compiler (p 21)

Note that executables created by these compilers are not compatible with each other. A module compiled with compiler A can't be loaded by the executable compiled with compiler B.

Compiling using Microsoft Visual C++ compiler

Frepple comes with Microsoft Visual C++ projects and workspaces to build Frepple.

The solution file is **contrib/vc/frepple.sln** and more detailed build instructions are provided in the README.txt file in this directory.

The project configuration files are generated with version 8 of Visual C++ and unfortunately are not compatible with earlier releases :- (:- (:- (

A free version of the compiler and the IDE, called "Visual C++ 2005 Express Edition", can be downloaded from the Microsoft website.

Make sure the include and library directories of your Python installation can be found by the compiler.

Compiling using the Cygwin compiler

Cygwin is a Linux-like environment for Windows. The Cygwin environment can be downloaded free of charge from the *Cygwin website*².

The build instructions on Cygwin are identical to the Linux and Unix platforms.

Cygwin does provide all the packages listed above. However, the xerces-c version is older and you'll

² www.cygwin.com

need to recompile it.

To build the latest xerces library for cygwin, remove the eventual previous xerces release in your cygwin directory, download the source from the xerces web page and then follow the build instructions. (Hint: Use the following to configure the build: `./runConfigure -p cygwin -c gcc -x g++ -P /usr`)

Compared to the other platforms and compilers, the Cygwin executables are considerably slower, break more often, and take longer to compile. That's the price for an emulation layer. . . . As a result, consider the Cygwin environment as a test and development environment for a *nix environment and not as a production system.

Compiling using Borland C++ compiler

The Frepple source distribution comes with a Borland C++ compiler make file and detailed compilation instructions in a README.txt file. Both are located in the subdirectory *contrib/borland*.

The Borland compiler can be downloaded free of charge from the *Borland website*³.

The borland build hasn't been kept up to date for a long time now and some work is required to get it ready again. All helping hands welcome. . .

2.2 Installing on Linux, Unix and CygWin

1. Build instructions
2. Compiling from the Subversion repository
3. VMware virtual machine

2.2.1 Build instructions

The following describes the steps you need to build Frepple.

1. Update your system with the development software packages.
 - **gcc**, v3.2 or v4.0
Front end for the GNU compiler suite.
 - **gcc-c++**, compatible with gcc release
GNU C++ compiler.
 - **xerces-c**, v2.7 or later
Xerces is a validating XML parser provided by the Apache Foundation.
You need to install the libraries as well as the development libraries.
 - **python** v2.4 or higher
Python is a modern, easy to learn interpreted programming language.
See <http://www.python.org> for more information. The language is used to a) run the test suite, b) script custom logic in frepple and c) to run the web application framework django.
You need to install the language as well as the development libraries.
2. Change to the frepple installation directory.

³ www.borland.com

3. Issue the command `./configure --prefix=<dir> --with-python` to specify the build options and detect the specifics of your platform.
Use the command `./configure --help` to see the list of available options.
It is **recommended** to use the `--prefix=<dir>` configuration option to set the installation directory. Installing Frepple in its own folder with normal user permissions fits better with the typical usage. Only use the standard `/usr` or `/usr/local` installation prefixes if you are planning to link Frepple into your own application.
4. Issue the command **‘make all’** to compile the code.
5. Optionally, issue the command **‘make check’** to run the test suite.
Not all tests are currently passing, so you shouldn’t be worried about a couple of failures. :-)
6. Issue the command **‘make install’** to install the files.
7. You can issue the command **‘make clean’** to free the disk space used during the build and test phases.
8. Optionally, if you are interested in some of the add-ons in the `contrib` subdirectory, follow the instructions in the `README.txt` file in each of the add-on directory.
You may need to install additional software components for a certain add-on. As a reference, here is a brief summary list of those components:
 - **Django**, v0.96
A web application framework written in python.
Django supports a number of databases, but Frepple (currently) works only with PostgreSQL as the underlying database.
In addition Django needs *psycpg2*⁴ as the database driver, the *apache web server*⁵ and *mod_python*⁶.
Visit the *django website*⁷ for full details.
 - **swig**, any version should do
SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is used with different types of languages including common scripting languages such as Perl, PHP, Python, Tcl, Ruby and PHP.
 - **GLPK**, any version should do
The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP) and other related problems. It can be downloaded from <http://www.gnu.org/software/glpk/glpk.html>
 - **NSIS**, version greater or equal to 2.07
NSIS, which stands for “Nullsoft Scriptable Installation System”, is a free scriptable win32 installer/uninstaller system that doesn’t suck and isn’t huge.
This program can be downloaded from <http://nsis.sf.net> and you’ll only need it if you are planning to create a windows installation package.
9. Frepple uses the environment variable `FREPPLE_HOME` to point to its configuration files. It typically points to the `bin` directory under the `prefix` directory of the folder chosen in step 3. You would set the variable in your login script of your shell or a wrapper shell script.
For bash/ksh/sh Bourne shells, add the following line to the file `.profile` in your home directory:
‘export FREPPLE_HOME=“/home/me/frepple/bin”’
For csh/tcsh C shells, add the following line to the file `.cshrc` in your home directory:

⁴ www.initd.org/tracker/psycpg/wiki/PyiscpgTwo

⁵ httpd.apache.org

⁶ www.modpython.org

⁷ www.djangoproject.com

'setenv FREPPLE_HOME "/home/me/frepple/bin"' Frepple expects to find the XML-schema file `frepple.xsd` in this directory. The file is used to validate the XML data. If a file called `init.xml` is found in this directory, it will be automatically executed when frepple starts up.

2.2.2 Compiling from the Subversion repository

To work with the code from the repository, follow the steps below.

Step 3 is the main difference with the build process from a distribution.

1. Your machine will need the following software components in addition to the ones listed for compiling from a distribution file:
 - **autoconf**, v2.59 or later
Gnu Autoconf produces shell scripts to automatically configure software source code packages. This makes the source code easier to port across the different *nix flavors.
 - **automake**, v1.9.5 or later
Gnu Automake is a tool for automatically generating make-files.
 - **libtool**, v1.5 or later
Libtool hides the complexity of developing and using shared libraries for different platforms behind a consistent and portable interface.
 - **gdb**, compatible with gcc release
Debugger for the gcc environment.
 - **doxygen**, any version should do
Extracts documentation from the C++ source code.
 - **subversion**, any version should do
Excellent version control tool.
2. Pick up the latest code from the repository with the command:
svn checkout https://frepple.svn.sourceforge.net/svnroot/trunk <project_directory>
 More information on working with the Sourceforge svn repositories can be found on http://sourceforge.net/docman/display_doc.php?docid=31070&group_id=1
 The repository allows anonymous connections for checkouts and it is also possible to browse it online from <http://frepple.svn.sourceforge.net/viewvc/frepple/>
3. Initialize the automake/autoconf/libtool scripts:
cd <project_directory>
make -f Makefile.dist prep
 If the command fails you can try the following. It re-initializes all automake/autoconf/libtool scripts to the version you have available on your machine.
cd <project_directory>
make -f Makefile.dist prep_force
4. Now the configure script is up to date and you can follow the same steps as in the section Compiling from a distribution to compile the code.
5. To refresh your environment with the changes from the repository:
cd <project_directory>
svn update
make -f Makefile.dist prep
 The last command is optional, but still recommended.

2.2.3 VMware virtual machine

A VMware virtual machine is available with a complete model and a development environment. It is not intended to be used a production environment.

The setup is based on a Fedora 5 Linux distribution and has the following main software packages are:

- Linux kernel 2.6.18
- g++ compiler 4.1.1
- xerces-c 2.7.0
- mysql 5.0.27
- python 2.4.3.9
- apache httpd 2.2
- django 0.95.1
- mod_python 3.2.8
- vmware tools are installed

To get up and running:

1. Download and install the VMWare server from <http://www.vmware.com/>.
2. Download and unzip the virtual machine from the sourceforge site.
3. Using the VMware console open the virtual machine “frepple.vmx” and start it.
4. When started the login screen will display the URL where you can browse the demo environment.
5. Instructions about login details, user accounts, database instance, etc are included in the README.txt file included with the virtual machine.

2.3 Other platforms

Frepple hasn't been compiled on any other platforms.

If you succeed in porting the code to another platform, please let us know and give us a hand in updating this document.

In the developer documentation a section is included listing some potential portability issues.

CHAPTER

3

Modeling

This chapter describe the Frepple data entities, their fields and relationships.

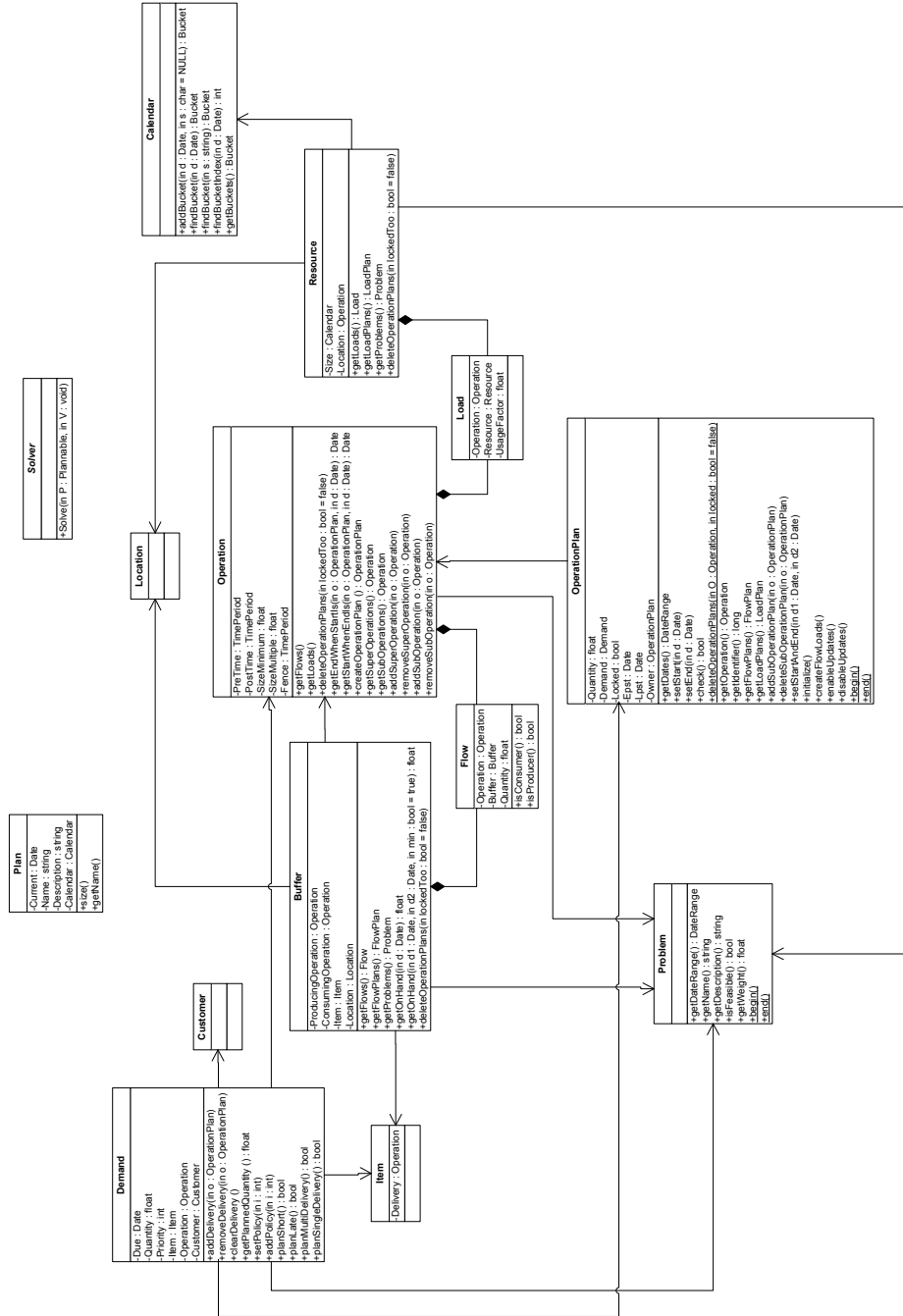
A couple of remarks to keep in mind:

- Frepple limits itself to the data fields that are relevant for planning.
An ERP or similar systems are more transaction-oriented and contain plenty of more detailed information.
- The Frepple model is designed to be pretty “atomic” in order to be as generic as possible.
Quite often an entity in a source system will map into a collection or sequence of Frepple entities.

Before diving into the details, have a look at the domain model diagram. It shows clear and simple the main entities and their relationships.

1. Domain model
2. Global Parameters
3. Buffer
4. Calendar
5. Command
6. Customer
7. Demand
8. Flow
9. Item
10. Load
11. Location
12. Operation
13. OperationPlan
14. Problem
15. Resource
16. Solver

3.1 Domain model



3.2 Global Parameters

A number of global settings and parameters are described here.

3.2.1 Fields

| Field | Type | Description |
|-------------|------------------|--|
| NAME | normalizedString | Model name. Default is null. |
| DESCRIPTION | normalizedString | Free format description. |
| CURRENT | date | The ‘now’ date for the plan. It distinguishes the past from the future. |
| LOGFILE | normalizedString | File name where all output will be sent to. If left unspecified, the output appears on the standard output. |

3.2.2 Example XML structures:

- Global initialization section

```
<PLAN>
  <NAME>Demo model</NAME>
  <DESCRIPTION>A demo model demonstrating frepple</DESCRIPTION>
  <CURRENT>2007-01-01T00:00:00</CURRENT>
  <LOGFILE>frepple.log</LOGFILE>
</PLAN>
```

3.3 Buffer

A buffer is a storage for a item.

Normally they represent a place where inventory of an item is kept.

Different types of buffers exist:

- BUFFER_DEFAULT (p 29): The default buffer uses an “producing” operation to replenish it with additional material.
- BUFFER_PROCURE (p 29): A buffer that is replenished by a supplier. A number of parameters control the re-ordering policy: classic re-order point, fixed time ordering, fixed quantity ordering, etc. . .
- BUFFER_INFINITE (p 30): An infinite buffer has an infinite supply of the material is available.

3.3.1 Fields

| Field | Type | Description |
|----------------|-------------------|--|
| NAME | non-empty string | Name of the buffer. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | BUFFER | Buffers can be organized in a hierarchical tree. This field defines the parent buffer. No specific planning behavior are currently linked to such a hierarchy. |
| MEMBERS | list of BUFFER | Buffers can be organized in a hierarchical tree. This field defines a list of child buffers. |
| LOCATION | LOCATION | Location of the buffer. Default is null. |
| ITEM | ITEM | Item being stored in the buffer. Default is null. |
| ONHAND | float | Inventory level at the start of the time horizon. Default is 0. |
| MINIMUM | CALENDAR | Refers to a calendar storing the desired minimum inventory level, aka safety stock. The solver treats this as a soft constraint, ie it tries to meet this inventory level but will go below the minimum level if required to meet the demand. A problem is reported when the inventory drops below this level. The safety stock target is expressed as a quantity. If you want to define a safety stock target as a time value, you can set a post-operation time on the producing operation of a buffer. |
| MAXIMUM | CALENDAR | Refers to a calendar storing the maximum inventory level. This field is not used by the solver. A problem is reported when the inventory level is higher than this limit. |
| PRODUCING | OPERATION | This operation will be instantiated by the solver to replenish the buffer with additional material. |
| DETECTPROBLEMS | boolean | Set this field to false to suppress problem detection on this buffer. Default is true. |
| FLows | list of FLOW | Defines material flows consuming from or producing into this buffer. |
| FLOW_PLANS | list of FLOW_PLAN | This field is populated during an export with the plan results for this buffer. It shows all the inventory profile. |

| | | |
|--------|-----------------------------|---|
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |
|--------|-----------------------------|---|

3.3.2 BUFFER_DEFAULT

The default buffer uses an “producing” operation to replenish it.

No fields are defined in addition to the ones listed above.

3.3.3 BUFFER_PROCURE

A procurement buffer is replenished by a supplier.

A number of parameters control the re-ordering policy: classic re-order point, fixed time ordering, fixed quantity ordering, etc. . .

The parameters LEADTIME, MININVENTORY and MAXINVENTORY define a replenishment with a classical re-orderpoint policy. The inventory profile will show the typical sawtooth shape.

The parameters MININTERVAL and MAXINTERVAL put limits on the frequency of replenishments. The inventory profile will have “teeth” of variable size but with a controlled interval.

The parameters SIZE_MINIMUM, SIZE_MAXIMUM and SIZE_MULTIPLE put limits on the size of the replenishments. The inventory profile will have “teeth” of controlled size but with variable intervals.

Playing with these parameters allows flexible and smart procurement policies to be modelled.

Note that frepple doesn't include any logic to set these parameters in an optimal way. The parameters are to be generated externally and frepple only executes based on the parameter settings.

At a later stage frepple may include an add-on to compute these parameters.

The PRODUCING field is unused for this buffer type.

Propagation through a bill of material will be stopped at a procurement buffer.

| Field | Type | Description |
|----------|------------|--|
| LEADTIME | TimePeriod | Time taken between placing the purchase order with the supplier and the delivery of the material. When the “LEADTIME” constraint is enabled in the solver, it won't create any new procurement orders that would need to start in the past. |

| | | |
|---------------|----------------|--|
| FENCE | TimePeriod | Time window (from the current date of the plan) during which procurement orders are expected to be released. When the “FENCE” constraint is enabled in the solver, it won’t create any new operation plans in this time fence. Only the externally supplied existing procurement plans will then exist in this time window. |
| MININVENTORY | Positive float | Inventory level triggering a new replenishment. The actual inventory can drop below this value. |
| MAXINVENTORY | Positive float | Inventory level to which we try to replenish. The actual inventory can exceed this value. |
| MININTERVAL | TimePeriod | Minimum time between replenishments. The order quantity will be increased such that it covers at least the demand in the minimum interval period. The actual inventory can exceed the target set by the MinimumInventory parameter. |
| MAXINTERVAL | TimePeriod | Maximum time between replenishments. The order quantity will replenish to an inventory value less than the maximum when this maximum interval is reached. |
| SIZE_MINIMUM | Positive float | Minimum quantity for a replenishment. This parameter can cause the actual inventory to exceed the target set by the MinimumInventory parameter. |
| SIZE_MAXIMUM | Positive float | Maximum quantity for a replenishment. This parameter can cause the maximum inventory target never to be reached. |
| SIZE_MULTIPLE | Positive float | All replenishments are rounded up to a multiple of this value. |

3.3.4 BUFFER_INFINITE

An infinite buffer has an infinite supply of the material is available.

The PRODUCING field is unused for this buffer type.

Propagation through a bill of material will be stopped at an infinite buffer.

3.3.5 Example XML structures:

- Adding a buffer

```
<PLAN>
  <BUFFERS>
    <BUFFER NAME="item a @ location b">
      <ITEM NAME="item a" />
      <LOCATION NAME="location b" />
      <ONHAND>10</ONHAND>
    </BUFFER>
  </BUFFERS>
```

```
</PLAN>
```

- Update the current inventory information of an existing buffer

```
<PLAN>
```

```
<BUFFERS>
```

```
<BUFFER NAME="item a @ location b" ONHAND="100" ACTION="C" />
```

```
</BUFFERS>
```

```
</PLAN>
```

- Deleting a buffer

```
<PLAN>
```

```
<BUFFERS>
```

```
<BUFFER NAME="item a @ location b" ACTION="R"/>
```

```
</BUFFERS>
```

```
</PLAN>
```

3.4 Calendar

A calendar represents a value that is varying over time.

Calendars can be linked to multiple entities: a maximum capacity limit of a resource, a minimum capacity usage of a resource, a minimum or maximum inventory limit of a buffer, etc...

Different types of calendar exist:

- CALENDAR_VOID: A calendar without any value in its buckets.
- CALENDAR_FLOAT: A calendar storing float values.
- CALENDAR_INTEGER: A calendar storing integer values.
- CALENDAR_BOOLEAN: A calendar storing boolean values.
- CALENDAR_STRING: A calendar storing string values.
- CALENDAR_OPERATION: A calendar storing operation values.

3.4.1 Calendar Fields

| Field | Type | Description |
|---------|------------------|--|
| NAME | non-empty string | Name of the calendar. This is the key field and a required attribute. |
| BUCKETS | List of BUCKET | A list of a buckets. |

| | | | |
|--------|--------------|--------------------------------|---|
| ACTION | A | Type of action to be executed: | <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |
| | C | | |
| | AC (default) | | |
| | R | | |

3.4.2 Bucket Fields

| Field | Type | Description |
|--------|-------------------------------|---|
| START | Date | Start date of the validity of this bucket. The value is effective till the start date of the next bucket. This is the key field and a required attribute. |
| NAME | non-empty string | Optional name of the bucket. |
| VALUE | Varies with the calendar type | The actual time-varying value. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none">• A: Add an new entity, and report an error if the entity already exists.• C: Change an existing entity, and report an error if the entity doesn't exist yet.• AC: Change an entity or create a new one if it doesn't exist yet.• R: Remove an entity, and report an error if the entity doesn't exist. |

3.4.3 Example XML structures:

- Adding a calendar and its buckets

```

<PLAN>
  <CALENDARS>
    <CALENDAR NAME="cal" xsi:type="CALENDAR_FLOAT">
      <BUCKETS>
        <BUCKET START="2007-01-01T00:00:00" VALUE="10"/>
        <BUCKET START="2007-02-01T00:00:00" VALUE="20"/>
        <BUCKET START="2007-03-01T00:00:00" VALUE="10"/>
      </BUCKETS>
    </CALENDAR>
  </CALENDARS>
</PLAN>

```

- Removing a calendar

```
<PLAN>
  <CALENDARS>
    <CALENDAR NAME="cal" ACTION="R"/>
  </CALENDARS>
</PLAN>
```

3.5 Command

All state changes in frepple are modeled as commands.

Commands are read from XML input, and executed **at the end** of parsing/processing all input. Commands are read and executed, but are never exported or saved again.

A wide range of commands exists to control the application:

- **COMMAND_LIST** (p 33) groups a number of commands, which can be executed in sequence or in parallel.
- **COMMAND_LOADLIB** (p 34) enables Frepple to dynamically load extension modules.
- **COMMAND_SYSTEM** (p 35) executes a operating system command.
- **COMMAND_READXML** (p 35) processes a XML-file from the local file system.
- **COMMAND_READXMLSTRING** (p 36) processes a XML-formatted string.
- **COMMAND_READXMLURL** (p 36) retrieves remote XML-data over an HTTP connection and processes it.
- **COMMAND_SETENV** (p 37) updates an environment variable.
- **COMMAND_IF** (p 38) allows for conditional execution of commands.
- **COMMAND_ERASE** (p 38) removes part of the model or plan from memory.
- **COMMAND_SAVE** (p 39) saves the model to an XML-formatted file.
- **COMMAND_SAVEPLAN** (p 39) saves the most important plan information to a file.
- **COMMAND_SIZE** (p 40) prints information about the memory size of the model and other state parameters.
- **COMMAND_SOLVE** (p 40) runs a solver.

3.5.1 COMMAND_LIST

This command groups a number of commands, which can be executed in sequence or in parallel.

| Field | Type | Description |
|---------|---------|---|
| COMMAND | COMMAND | The sub-commands part of this list. Multiple sub-commands can be defined. |

| | | |
|--------------|------------------|--|
| ABORTONERROR | Boolean | <p>When executing commands sequentially, this field specifies the behavior in the case of an error:</p> <ul style="list-style-type: none"> • When set to false, the execution will simply continue with the next command. • When set to true, the execution of the list will be aborted. <p>The default is true.</p> |
| MAXPARALLEL | Positive integer | Maximum number of commands to be executed in parallel. The default value is 1, ie sequential execution. |
| VERBOSE | Boolean | Echo information about the command execution in the log. This field is inherited by the sub-commands. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <VERBOSE>true</VERBOSE>
    <COMMAND xsi-type="COMMAND_LIST" MAXPARALLEL="100">
      <COMMAND xsi:type="COMMAND_SYSTEM"
        CMDLINE="sleep 1 && echo  after 1 second" />
      <COMMAND xsi:type="COMMAND_SYSTEM"
        CMDLINE="sleep 2 && echo  after 2 second" />
    </COMMAND>
  </COMMANDS>
</PLAN>
```

3.5.2 COMMAND_LOADLIB

This command enables Frepple to dynamically load extension modules.

| Field | Type | Description |
|-----------|-----------|---|
| FILENAME | String | <p>Name of the shared library file to be loaded. The operating system should allow frepple to locate the file. The directories listed in the following environment variable should include the module shared library.</p> <ul style="list-style-type: none"> • LD_LIBRARY_PATH variable for Linux, Solaris • LIBPATH for AIX • SHLIB_PATH for HP-UX • PATH for windows and cygwin |
| PARAMETER | Parameter | Initialization and configuration values that are passed to the module's initialization routine. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <VERBOSE>true</VERBOSE>
    <COMMAND xsi:type="COMMAND_LOADLIB" FILENAME="mod_python.so" />
    <COMMAND xsi:type="COMMAND_LOADLIB" FILENAME="your_module.so">
      <PARAMETER NAME="test1" VALUE="val1"/>
      <PARAMETER>
        <NAME>test2</NAME>
        <VALUE>val2</VALUE>
      </PARAMETER>
    </COMMAND>
  </COMMANDS>
</PLAN>
```

3.5.3 COMMAND_SYSTEM

executes a operating system command.

| Field | Type | Description |
|---------|---------|--|
| CMDLINE | String | Command line to be executed in an operating shell. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <VERBOSE>true</VERBOSE>
    <COMMAND xsi:type="COMMAND_SYSTEM" CMDLINE="sleep 1" />
    <COMMAND xsi:type="COMMAND_SYSTEM" CMDLINE="do_something.sh" />
  </COMMANDS>
</PLAN>
```

3.5.4 COMMAND_READXML

This command reads and processes a XML-file from the local file system.

| Field | Type | Description |
|----------|---------|---|
| FILENAME | String | Name of the data file to be loaded. |
| VALIDATE | Boolean | When set to true, the XML data are validated against the XML-schema. The default value is true, for security reasons. When parsing large files with a trusted structure setting this field to false will speed up the import. |

| | | |
|---------|---------|--|
| VERBOSE | Boolean | Echo information about the command execution in the log. |
|---------|---------|--|

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_READXML" FILENAME="input.xml" />
  </COMMANDS>
</PLAN>
```

3.5.5 COMMAND_READXMLSTRING

This command processes a XML-formatted data string.

| Field | Type | Description |
|----------|---------|--|
| DATA | String | XML-formatted data to be processed. |
| VALIDATE | Boolean | When set to true, the XML data are validated against the XML-schema. The default value is true, for security reasons. When processing large data strings with a trusted structure setting this field to false will speed up the execution. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_READXMLSTRING">
      <DATA>
        <![CDATA[
          <PLAN xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <LOCATIONS>
              <LOCATION NAME="Location 1" ACTION="R"/>
            </LOCATIONS>
          </PLAN>
        ]]>
      </DATA>
    </COMMAND>
  </COMMANDS>
</PLAN>
```

3.5.6 COMMAND_READXMLURL

This command is used for reading XML input over an HTTP connection.

The Xerces parser used to implement this feature supports only the simplest possible setup: no proxy, no caching, no ssl, no authentication, etc. . .

In more complex setup the following alternatives are available:

- Use a sytem command to retrieve the data first with eg a command line utility such as curl, wget, lynx, rcp, scp, ftp, sftp, rsynch, . . .
After the download read in the data file with a readXML command.
- Use the embedded Python interpreter and its rich library of functionality to download the data.
The test 'xml_remote' shows some simple code for this.

Given the serious limitations and the availability of better alternatives, this command may well be removed in a future release.

| Field | Type | Description |
|----------|---------|--|
| URL | String | URL where the XML data can be downloaded from. |
| VALIDATE | Boolean | When set to true, the XML data are validated against the XML-schema. The default value is true, for security reasons. When processing large data volumes with a trusted structure setting this field to false will speed up the execution. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_READXMLURL"
      URL="http://frepple.sourceforge.net/test/xml_remote.xml " />
  </COMMANDS>
</PLAN>
```

3.5.7 COMMAND_SETENV

This command updates an environment variable.

| Field | Type | Description |
|----------|---------|--|
| VARIABLE | String | Environment variable to be updated. |
| VALUE | String | New value of the variable. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
```

```

    <COMMAND xsi:type="COMMAND_SETENV" VARIABLE="VAR1" VALUE="VAL1" />
    <!-- Showing the variables in a shell command. -->
    <COMMAND xsi:type="COMMAND_SYSTEM" CMDLINE="echo ${VAR1}" />
  </COMMANDS>
</PLAN>

```

3.5.8 COMMAND_IF

This command allows for conditional execution of commands.

| Field | Type | Description |
|-----------|---------|---|
| CONDITION | String | Expression that is evaluated in an operating system shell. |
| THEN | COMMAND | Command to be executed when the condition evaluates to true. |
| ELSE | COMMAND | Command to be executed when the condition evaluates to false. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```

<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_IF">
      <CONDITION>test -f yourfile</CONDITION>
      <THEN xsi:type="COMMAND_SYSTEM"
        CMDLINE="echo 'the file exists'" />
      <ELSE xsi:type="COMMAND_SYSTEM"
        CMDLINE="echo 'the file doesnt exist'" />
    </COMMAND>
  </COMMANDS>
</PLAN>

```

3.5.9 COMMAND_ERASE

This command allows for conditional execution of commands.

| Field | Type | Description |
|---------|---------------|--|
| MODE | Plan Model | When set to “model” the complete model is erased. You will again have a completely empty model. When set to “plan” only the plan information is erased, ie all operationplans with their load- and flowplans are removed (except the ones that are locked). |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_ERASE" MODE="plan" />
  </COMMANDS>
</PLAN>
```

3.5.10 COMMAND_SAVE

This commands saves the model into an XML-formatted file.

| Field | Type | Description |
|-------------|--------------------------------|---|
| FILENAME | String | Name of the output file. |
| CONTENT | STANDARD PLAN PLANDETAIL | Controls the level of detail in the output: <ul style="list-style-type: none"> STANDARD plan information is sufficient for restoring the model from the output file.\\ This is the default mode. PLAN adds more detail about its plan with each entity. A buffer will report on its flowplans, a resource reports on its loadplans, and a demand on its delivery operationplans. PLANDETAIL goes even further and includes full pegging information the output. A buffer will report how the material is supplied and which demands it satisfies, a resource will report on how the capacity used links to the demands, and a demand shows the complete supply path used to meet it. |
| HEADERSTART | String | The first line of the XML output. The default value is: <?xml version="1.0" encoding="UTF-8"?> |
| HEADERATTS | String | Predefined attributes of the XML root-element. The default value is: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_SAVE" FILENAME="output.xml" />
  </COMMANDS>
</PLAN>
```

3.5.11 COMMAND_SAVEPLAN

This command saves the most important plan information to a file.

It is used for the unit tests, but its' usefulness in a real-life implementation is probably limited.

| Field | Type | Description |
|----------|---------|--|
| FILENAME | String | Name of the output file. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_SAVEPLAN" />
      <FILENAME>output.xml</FILENAME>
    </COMMAND>
  </COMMANDS>
</PLAN>
```

3.5.12 COMMAND_SIZE

This command prints information about the memory size of the model and other state parameters.

| Field | Type | Description |
|---------|---------|--|
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi:type="COMMAND_SIZE" />
  </COMMANDS>
</PLAN>
```

3.5.13 COMMAND_SOLVE

This command will execute a solver.

| Field | Type | Description |
|---------|---------|--|
| SOLVER | SOLVER | Points to the solver to execute. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <VERBOSE>true</VERBOSE>
    <COMMAND xsi:type="COMMAND_SOLVE">
      <SOLVER xsi:type="SOLVER_MRP" NAME="MRP"
        CONSTRAINTS="7" VERBOSE="false" />
    </COMMAND>
  </COMMANDS>
</PLAN>
```

3.6 Customer

Demands are associated with a customer.

Customers can be organized in a hierarchical tree to represent the sales organization's structure.

Frepple uses customers only from reporting purposes, no real planning logic is currently linked to them.

3.6.1 Fields

| Field | Type | Description |
|-------------|-----------------------------|--|
| NAME | non-empty string | Name of the customer. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | CUSTOMER | Customers are organized in a hierarchical tree. This field defines the parent customer. |
| MEMBERS | list of CUSTOMER | Customers are organized in a hierarchical tree. This field defines a list of child customer. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.6.2 Example XML structures:

- Adding a customer

```
<PLAN>
  <CUSTOMERS>
    <CUSTOMER NAME="customer A" CATEGORY="Direct"/>
  </CUSTOMERS>
</PLAN>
```

- Deleting a customer

```
<PLAN>
  <CUSTOMERS>
    <CUSTOMER NAME="customer A" ACTION="R"/>
  </CUSTOMERS>
</PLAN>
```

3.7 Demand

Define independent demands for items.

These can be actual customer orders, or forecasted demands.

3.7.1 Fields

| Field | Type | Description |
|-------------|------------------|---|
| NAME | non-empty string | Name of the demand. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | DEMAND | Demands are organized in a hierarchical tree. This field defines the parent demand. |
| MEMBERS | list of DEMAND | Demands are organized in a hierarchical tree. This field defines a list of child demand. |
| QUANTITY | Float | Requested quantity. |
| ITEM | ITEM | Requested item. |
| DUE | Date | Due date of the demand. |
| PRIORITY | Integer | Priority of the demand relative to the other demands. A lower number indicates higher priority. The default value is 0. |

| | | |
|----------------|--|---|
| OPERATION | OPERATION | Operation to be used to satisfy the demand. If left unspecified the operation on the item will be used. |
| CUSTOMER | CUSTOMER | Customer placing the demand. |
| POLICY | Space delimited sequence of policy names | <p>The demand policy controls how the demand needs to be plan in case of constraints:</p> <ul style="list-style-type: none"> • PLANLATE: Allows a demand to be satisfied late. • PLANSHORT: Disallows satisfying the demand late, and plans it short on the due date instead. This is the opposite of the PLANLATE policy. • MULTIDELIVERY: Allows a demand to be satisfied with multiple partial deliveries. • SINGLEDELIVERY: Forces a demand to be satisfied in full in a single delivery, or remain unplanned. This is the opposite of the MULTIDELIVERY policy. <p>The default policy is PLANLATE MULTIDELIVERY.</p> |
| DETECTPROBLEMS | boolean | Set this field to false to suppress problem detection on this demand. Default is true. |
| ACTION | A C AC (default) R | <p>Type of action to be executed:</p> <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.7.2 Example XML structures:

- Adding a demand

```
<PLAN>
```

```
<DEMANDS>
```

```

<DEMAND NAME="order ABC">
  <QUANTITY>10</QUANTITY>
  <DUE>2007-01-10T00:00:00</DUE>
  <PRIORITY>1</PRIORITY>
  <ITEM NAME="item 1" />
</DEMAND>

```

```

<DEMAND NAME="order DEF" QUANTITY="10" DUE="2007-01-10T00:00:00" PRIORITY="1" >
  <ITEM NAME="item 1" />

```

```

    </DEMAND>
  </DEMANDS>
</PLAN>

```

- Removing a demand

```

<PLAN>
  <DEMANDS>
    <DEMAND NAME="order ABC" ACTION="R"/>
  </DEMANDS>
</PLAN>

```

3.8 Flow

Flows are used to model the consumption and production of material from buffers.

Two types of flows exist:

- **FLOW_START**: Flows that consume material at the start of an operationplan.
- **FLOW_END**: Flows that produce material at the end of an operationplan.

3.8.1 Fields

| Field | Type | Description |
|-----------|-----------------------------|--|
| BUFFER | BUFFER | Buffer from which material will be moved or transferred into. This is a required field. |
| OPERATION | OPERATION | Operation to which the material flow is associated. This is a required field. |
| QUANTITY | Float | Material quantity being consumed or produced per unit of the operationplan. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.8.2 Example XML structures:

- Defining a flow

```

<PLAN>

```

```

    <FLOWS>
      <FLOW xsi:type="FLOW_START">
        <BUFFER NAME="buffer component"/>
        <OPERATION NAME="operation B"/>
        <QUANTITY>-2</QUANTITY>
      </FLOW>
    </FLOWS>
  </PLAN>

```

- Defining a flow nested in an operation structure

```

<PLAN>
  <OPERATIONS>
    <OPERATION NAME="operation B">
      <FLOWS>
        <FLOW xsi:type="FLOW_START">
          <BUFFER NAME="buffer component"/>
          <QUANTITY>-2</QUANTITY>
        </FLOW>
        <FLOW xsi:type="FLOW_END">
          <BUFFER NAME="buffer end item"/>
          <QUANTITY>1</QUANTITY>
        </FLOW>
      </FLOWS>
    </OPERATION>
  </OPERATIONS>
</PLAN>

```

- Defining a flow nested in a buffer structure

```

<PLAN>
  <BUFFERS>
    <BUFFER NAME="buffer component">
      <FLOWS>
        <FLOW xsi:type="FLOW_START">
          <OPERATION NAME="operation A"/>
          <QUANTITY>-2</QUANTITY>
        </FLOW>
        <FLOW xsi:type="FLOW_START">
          <OPERATION NAME="operation B"/>
          <QUANTITY>-1</QUANTITY>
        </FLOW>
      </FLOWS>
    </BUFFER>
  </BUFFERS>
</PLAN>

```

- Deleting a flow

```

<PLAN>
  <FLOWS>
    <FLOW ACTION="R">

```

```

        <BUFFER NAME="buffer component"/>
        <OPERATION NAME="operation B"/>
    </FLOW>
</FLOWS>
</PLAN>

```

3.9 Item

An item represents a end product, intermediate product or a raw material.

Each demand is associated with an item.

A buffer is also associated with an item: it represents a storage of the item.

3.9.1 Fields

| Field | Type | Description |
|-------------|-----------------------------|--|
| NAME | non-empty string | Name of the item. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | ITEM | Items are organized in a hierarchical tree. This field defines the parent item. |
| MEMBERS | list of ITEM | Items are organized in a hierarchical tree. This field defines a list of child items. |
| OPERATION | OPERATION | This is the operation used to satisfy a demand for this item. If left unspecified the value is inherited from the parent item. See also the OPERATION field on the DEMAND. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.9.2 Example XML structures:

- Adding an item

```
<PLAN>
  <ITEMS>
    <ITEM NAME="item A">
      <OPERATION NAME="Delivery of item A"
        xsi:type="OPERATION_FIXED_TIME">
        <DURATION>24:00:00</DURATION>
      </OPERATION>
      <OWNER NAME="Item class A"/>
    </ITEM>
  </ITEMS>
</PLAN>
```

- Deleting an item

```
<PLAN>
  <ITEMS>
    <ITEM NAME="item A" ACTION="R"/>
  </ITEMS>
</PLAN>
```

3.10 Load

Loads are used to model the capacity consumption of an operation.

3.10.1 Fields

| Field | Type | Description |
|-----------|-----------------------------|--|
| RESOURCE | RESOURCE | Resource being loaded. This is a required field. |
| OPERATION | OPERATION | Operation loading the resource. This is a required field. |
| USAGE | Float | Load factor of the resource. The default value is 1.0. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.10.2 Example XML structures:

- Defining a load

```
<PLAN>
  <LOADS>
    <LOAD>
      <RESOURCE NAME="machine A"/>
      <OPERATION NAME="operation B"/>
    </LOAD>
  </LOADS>
</PLAN>
```

- Defining a load nested in an operation structure

```
<PLAN>
  <OPERATIONS>
    <OPERATION NAME="operation B">
      <LOADS>
        <LOAD>
          <RESOURCE NAME="machine A"/>
          <USAGE>1</USAGE>
        </LOAD>
      </LOADS>
    </OPERATION>
  </OPERATIONS>
</PLAN>
```

- Defining a load nested in a resource structure

```
<PLAN>
  <BUFFERS>
    <RESOURCE NAME="machine A">
      <LOADS>
        <LOAD>
          <OPERATION NAME="operation B"/>
          <USAGE>2</USAGE>
        </LOAD >
        <LOAD>
          <OPERATION NAME="operation C"/>
          <USAGE>1</USAGE>
        </LOAD>
      </LOADS>
    </BUFFER>
  </BUFFERS>
</PLAN>
```

- Deleting a load

```
<PLAN>
  <LOADS>
```

```

    <LOAD ACTION="R">
      <RESOURCE NAME="machine A"/>
      <OPERATION NAME="operation B"/>
    </LOAD>
  </LOADS>
</PLAN>

```

3.11 Location

A location is a (physical or logical) place where resources and buffers are located.

Frepple uses locations only from reporting purposes, no planning logic is currently linked to them.

3.11.1 Fields

| Field | Type | Description |
|-------------|-----------------------------|--|
| NAME | non-empty string | Name of the location. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | LOCATION | Locations are organized in a hierarchical tree. This field defines the parent location. |
| MEMBERS | list of LOCATION | Locations are organized in a hierarchical tree. This field defines a list of child locations. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.11.2 Example XML structures:

- Adding a location

```

<PLAN>
  <LOCATIONS>

```

```

    <LOCATION NAME="site A">
      <CATEGORY>cat A</CATEGORY>
      <OWNER NAME="Manufacturing sites"/>
    </LOCATION>
  </LOCATIONS>
</PLAN>

```

- Alternate format of the previous example

```

<PLAN>
  <LOCATIONS>
    <LOCATION NAME="Manufacturing sites">
      <MEMBERS>
        <LOCATION NAME="site A" CATEGORY="cat A"/>
      </MEMBERS>
    </LOCATION>
  </LOCATIONS>
</PLAN>

```

- Deleting a location

```

<PLAN>
  <LOCATIONS>
    <LOCATION NAME="site A" ACTION="R"/>
  </LOCATIONS>
</PLAN>

```

3.12 Operation

An operation represents an activity: these consume and produce material, take time and also require capacity.

An operation consumes and produces material, modeled through flows.

An operation requires capacity, modeled through loads.

Different operation types exist:

- OPERATION_FIXED_TIME (p 52): Models an operation with a fixed duration regardless of the quantity.
E.g. a transport operation.
- OPERATION_TIME_PER (p 52): Models an operation where the duration is linear with the quantity.
E.g. a production operation.
- OPERATION_ALTERNATE (p 52): Models a choice between different operations.
- OPERATION_ROUTING (p 53): Models a existing of a number of 'step' sub-operations, to be executed in sequence.

3.12.1 Fields

| Field | Type | Description |
|----------------|------------------|---|
| NAME | non-empty string | Name of the operation. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | OPERATION | Operation can be organized in a hierarchical tree. This field defines the parent operation. |
| FENCE | Time | Time window from the current date of the plan during which all operationplans are expected to be frozen / released. When the “FENCE” constraint is enabled in the solver, it won’t create any new operation plans in this time fence. Only the externally supplied operationplans will then exist in this time window. |
| SIZE_MINIMUM | Positive float | A minimum size for operationplans. A request for a lower quantity will be rounded up. |
| SIZE_MULTIPLE | Positive float | A lotsize quantity for operationplans. |
| PRETIME | Time | A pre-operation time, used as a buffer for uncertain material supply. The solver will try to position material supply for operation plans early by the time specified here. This is a soft constraint, ie it can be violated if required to meet the demand in time. |
| POSTTIME | Time | A post-operation time, used as a buffer for uncertain capacity or operation duration. The solver will try to respect this time as a soft constraint. Ie when required to meet demand on time the post-operation time can be violated. This field is used to model time-based safety stock targets. It is typically set for the producing operation of a certain buffer. If you want to model a safety stock quantity, you can use the minimum field on the buffer. |
| DETECTPROBLEMS | Boolean | Set this field to false to skip problem detection on this operation. The default value is true. |
| LOADS | List of LOAD | A list of all resources loaded by this operation. |
| FLows | List of FLOW | A list of all buffers where material is consumed from or produced into. |

| | | | |
|--------|--------------|--------------------------------|---|
| ACTION | A | Type of action to be executed: | <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |
| | C | | |
| | AC (default) | | |
| | R | | |

3.12.2 OPERATION_FIXED_TIME

Models an operation with a fixed duration regardless of the quantity.

E.g. a transport operation.

This is the default operation type.

| Field | Type | Description |
|----------|------|---|
| DURATION | Time | Duration of the operation. The default value is 0. |

3.12.3 OPERATION_TIME_PER

Models an operation where the duration changes linear with the quantity.

E.g. a production operation.

The total duration of the operation plan is the sum of:

- A fixed DURATION.
- A variable duration, computed as the operationplan quantity multiplied by a DURATION_PER.

| Field | Type | Description |
|--------------|------|---|
| DURATION | Time | Fixed component of the duration of the operationplan. The default value is 0. |
| DURATION_PER | Time | Variable component of the duration of the operationplan. The default value is 0. |

3.12.4 OPERATION_ALTERNATE

Models a choice between different operations.

It has a list of alternate sub-operations listed, each with a priority.

| Field | Type | Description |
|------------|-------------------|---|
| ALTERNATES | List of ALTERNATE | List of alternate sub-operations, each with their priority. |

ALTERNATE fields:

| Field | Type | Description(:table border=1 width="100%":) |
|-----------|-----------|---|
| OPERATION | OPERATION | Sub-operation. |
| Priority | Float | Sub-operation. Lower numbers indicate higher priority. |

3.12.5 OPERATION_ROUTING

Models a existing of a number of ‘step’ sub-operations, to be executed in sequence.

| Field | Type | Description |
|-------|-------------------|---|
| STEPS | List of OPERATION | Lists all sub-operations in the order of execution. |

3.12.6 Example XML structures:

- Adding operations

```
<PLAN>
  <OPERATIONS>
    <OPERATION NAME="buy item X from supplier" xsi:type="OPERATION_FIXED_TIME">
      <DURATION>24:00:00</DURATION>
    </OPERATION>
    <OPERATION NAME="make item X" xsi:type="OPERATION_TIME_PER">
      <DURATION>1:00:00</DURATION>
      <DURATION_PER>5:00</DURATION>
    </OPERATION>
    <OPERATION NAME="make or buy item X" xsi:type="OPERATION_ALTERNATE">
      <ALTERNATES>
        <ALTERNATE>
          <OPERATION NAME="make item X" />
          <PRIORITY>1</PRIORITY>
        </ALTERNATE>
        <ALTERNATE>
          <OPERATION NAME="buy item X from supplier" />
          <PRIORITY>2</PRIORITY>
        </ALTERNATE>
      </ALTERNATES>
    </OPERATION>
  </OPERATIONS>
</PLAN>
```

```

    </OPERATION>
    <OPERATION NAME="make subassembly" xsi:type="OPERATION_ROUTING">
      <STEPS>
        <OPERATION NAME="make subassembly step 1" DURATION="1:00:00"/>
        <OPERATION NAME="make subassembly step 2" DURATION="5:00:00"/>
      </STEPS>
    </OPERATION>
  </OPERATIONS>
</PLAN>

```

- Deleting an operation

```

<PLAN>
  <OPERATIONS>
    <OPERATION NAME="make item X" ACTION="R"/>
  </OPERATIONS>
</PLAN>

```

3.13 OperationPlan

Used to model an existing or planned activity.

This can represent work-in-progress, in-transit shipments, planned material receipts, frozen manufacturing plans, etc...

3.13.1 Fields

| Field | Type | Description |
|-----------|------------------|---|
| OPERATION | non-empty string | Name of the operation. This field is required when no identifier is provided. |
| ID | Unsigned long | Unique identifier of the operationplan. If left unspecified an identifier will be automatically generated. This field is required when updating existing instances. |
| START | Date | Start date. |
| END | Date | End date. |
| DEMAND | DEMAND | Points to the demand being satisfied with this operationplan. This field is only non-null for the actual delivery operationplans. |
| QUANTITY | Float | Quantity being planned. |
| LOCKED | Boolean | A locked operation plan is not allowed to be changed any more by any solver algorithm. |
| OWNER | OPERATION_PLAN | Points to a parent operationplan. The default is NULL. |

| | | |
|--------|-----------------------------|--|
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |
|--------|-----------------------------|--|

3.13.2 Example XML structures:

- Adding an operationplan to represent a planned receipt of material

```
<PLAN>
  <OPERATION_PLANS>
    <OPERATION_PLAN OPERATION="Purchase component A">
      <QUANTITY>100</QUANTITY>
      <START>2007-01-10T00:00:00</START>
      <LOCKED>true</LOCKED>
    </OPERATION_PLAN>
  </OPERATION_PLANS>
</PLAN>
```

- Deleting an operationplan

```
<PLAN>
  <OPERATION_PLANS>
    <OPERATION_PLAN ID="1020" ACTION="R"/>
  </OPERATION_PLANS>
</PLAN>
```

3.14 Problem

Frepple will automatically detect problems and inconsistencies in the plan.

Problem detection can optionally be disabled on entities by setting the field “DETECTPROBLEMS” to false.

3.14.1 Types

| Problem Entity | Problem Category | Description |
|----------------|------------------|---|
| Demand | unplanned | No plan exists yet to satisfy this demand. |
| Demand | excess | A demand is planned for more than the requested quantity. |
| Demand | short | A demand is planned for less than the requested quantity. |

| | | |
|---------------|-------------------|--|
| Demand | late | A demand is satisfied later than the accepted tolerance after its due date |
| Demand | early | A demand is planned earlier than the accepted tolerance before its due date. |
| Resource | overload | A resource is being overloaded during a certain period of time. |
| Resource | underload | A resource is loaded below its minimum during a certain period of time. |
| Buffer | material excess | A buffer is carrying too much material during a certain period of time. |
| Buffer | material shortage | A buffer is having a material shortage during a certain period of time. |
| Operationplan | before current | Flagged when an operationplan is being planned in the past, i.e. it starts before the current date of the plan. |
| Operationplan | before fence | Flagged when an operationplan is being planned before its fence date, i.e. it starts 1) before the current date of the plan plus the release fence of the operation and 2) after the current date of the plan. |
| Operationplan | precedence | Flagged when the sequence of two operationplans in a routing isn't respected. |

3.14.2 Fields

| Field | Type | Description |
|-------------|-----------|-------------------------------------|
| NAME | String | Problem type. |
| DESCRIPTION | String | Description of the problem. |
| DATES | Daterange | Dates over which the problem spans. |

3.15 Resource

Resources represent capacity.

They represent a machine, a worker or a group of workers, or some logical limits.

A calendar refers to a time-phased maximum limit of the resource usage.

Operations will consume capacity using loads.

Different types of resources exist:

- **RESOURCE_DEFAULT** (p 57): A default resource is constrained with a maximum available capacity.

- RESOURCE_INFINITE (p 58): An infinite resource has no capacity limit.

3.15.1 Fields

| Field | Type | Description |
|----------------|-----------------------------|--|
| NAME | non-empty string | Name of the resource. This is the key field and a required attribute. |
| DESCRIPTION | normalizedString | Free format description. |
| CATEGORY | normalizedString | Free format category. |
| SUBCATEGORY | normalizedString | Free format subcategory. |
| OWNER | RESOURCE | Resources can be organized in a hierarchical tree. This field defines the parent resource. No specific planning behavior is currently linked to such a hierarchy. |
| MEMBERS | list of RESOURCE | Resources can be organized in a hierarchical tree. This field defines a list of child resources. |
| LOCATION | LOCATION | Location of the resource. Default is null. |
| MAXIMUM | CALENDAR | Refers to a calendar storing the available capacity. A problem is reported when the resource load exceeds than this limit. |
| DETECTPROBLEMS | boolean | Set this field to false to suppress problem detection on this resource. Default is true. |
| LOADS | list of LOAD | Defines the capacity of the operations. |
| LOAD_PLANS | list of LOAD_PLAN | This field is populated during an export with the plan results for this resource. It shows all the resource load profile. |
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |

3.15.2 RESOURCE_DEFAULT

A default resource is constrained with a maximum available capacity.

No fields are defined in addition to the ones listed above.

3.15.3 RESOURCE_INFINITE

An infinite resource has no capacity limit.

It is useful to monitor the loading or usage.

The MAXIMUM field is unused for this resource type.

3.15.4 Example XML structures:

- Adding a resource

```
<PLAN>
  <RESOURCES>
    <RESOURCE NAME="machine X">
      <MAXIMUM NAME="capacity calendar for machine X" />
    </RESOURCE>
  </RESOURCES>
</PLAN>
```

- Deleting a resource

```
<PLAN>
  <RESOURCES>
    <RESOURCE NAME="machine X" ACTION="R"/>
  </RESOURCES>
</PLAN>
```

3.16 Solver

A solver represents modules of functionality that manipulate the model.

Examples are solvers to generate a plan, solvers to compute safety stocks, solvers to create production or purchase orders, etc. . .

Currently only 1 solver is available: SOLVER_MRP (p 59), which uses a heuristic algorithm to generate plans.

3.16.1 Fields

| Field | Type | Description |
|---------|------------------|--|
| NAME | non-empty string | Name of the location. This is the key field and a required attribute. |
| VERBOSE | Boolean | When set to true the solver echoes more logging information. |

| | | |
|--------|-----------------------------|---|
| ACTION | A C AC (default) R | Type of action to be executed: <ul style="list-style-type: none"> • A: Add an new entity, and report an error if the entity already exists. • C: Change an existing entity, and report an error if the entity doesn't exist yet. • AC: Change an entity or create a new one if it doesn't exist yet. • R: Remove an entity, and report an error if the entity doesn't exist. |
|--------|-----------------------------|---|

3.16.2 MRP Solver

| Field | Type | Description |
|-------------|------------------|--|
| CONSTRAINTS | Unsigned integer | Sum up the values of the constraints you want to enable in the solver: <ul style="list-style-type: none"> • 1: Lead times, ie don't plan in the past • 2: Material supply, ie don't allow inventory values to go negative • 4: Capacity, ie don't allow to overload resources • 8: Operation fences, ie don't allow to create plans in the frozen fence of operations |

3.16.3 Example XML structures:

- Adding a solver

```
<PLAN>
  <SOLVERS>
    <SOLVER NAME="MRP" xsi-type="SOLVER_MRP">
      <CONSTRAINTS>7</CONSTRAINTS>
    </SOLVER >
  </SOLVERS>
</PLAN>
```

- Deleting a solver

```
<PLAN>
  <SOLVERS>
    <SOLVER NAME="MRP" ACTION="R"/>
  </SOLVERS>
</PLAN>
```

CHAPTER

4

Solver Algorithm

The default solver is based on a heuristic algorithm. It is structured in a clear ask-reply pattern between the different entities.

The algorithm can create different types of plans. With the following three flags, a total of 8 combinations are possible:

- Material constrained or not:
Supply of raw material can be treated as finite or infinite.
- Capacity constrained or not:
Production capacity can be treated as finite or infinite.
- Leadtime constrained or not:
Allow or disallow plans to be created in the past.

It is possible to build create extensions to the solver, or to create a completely new solver altogether. The solvers can be loaded as plugin modules without touching or recompiling the main application.

1. Solver Features
2. Implementation details
 - 2.1. Top level loop
 - 2.2. Demand solver
 - 2.3. Buffer solver
 - 2.4. Operation solver
 - 2.5. Flow solver
 - 2.6. Load solver
 - 2.7. Resource Solver
3. Cluster and level algorithm

4.1 Solver Features

In brief, here are the main features of the solver:

4.1.1 Solver

- Ability to create **unconstrained plans**.
- Ability to respect following **constraints: material supply, available capacity, leadtime, re-lease time fence**.
- Ability to run in **multi-threaded** mode. Different threads are solving independent sub-problems.

4.1.2 Demand

- **Demand priorities** are recognized, such that constraints impact the lowest ranking demands only. The default ranking is based on the priority attribute and the due date.
- Ability to respect different **demand policies**: In case of a constraint a demand can be allowed to be satisfied late or not. Satisfying the demand in multiple parts can be allowed or not.

4.1.3 Operation

- Models **multiple operation types**:
 - Operations with fixed duration.
 - Operations with variable duration, depending on quantity.
 - Alternate operations: When a demand can't be met from the primary operation the solver will plan on alternative operations.
 - Date-effective operations: Depending on the start date (or end date) different operations are used.
 - Multi-step operations: An operation can have multiple sub-operations that need to be executed in sequence.
- The operations can be planned as a multiple of the **lot-size** quantity.
- A **minimum size** can be enforced when planning an operation.
- **Pre- and post-operation times** used as soft constraints (ie they are respected when feasible but will be reduced when required to meet the demand in time).

4.1.4 Resource

- Resources loaded during the complete duration of an operation.
- Resources with **finite or infinite capacity**.
- Capacity shortages are solved by **moving operations early**.

4.1.5 Buffer

- Material consumption or production happens at the start or at the end of operations.
- Buffers with **finite or infinite material supply**.

- Ability to specify a desired minimum inventory level, aka **safety stock**. The minimum level can be time dependent and is treated as a soft constraint (ie will be respected when feasible, but will be violated when constraints prevent meeting it).

4.2 Implementation details

The algorithm solves demand per demand. The demand is thus sorted in descending order of priority, and next these demands are planned one after the other.

When planning a single demand, the algorithm basically consists of a set of recursive functions structured in a ask-reply pattern, as illustrated in the example below. The indentation is such that the ask and its matching reply are represented at the same level.

Every demand has a certain delivery operation associated with it, either directly or indirectly by specifying a delivery operation for the requested item. The demand **asks** this **operation** for the requested quantity on the due date of the demand.

(*) The operation first checks for the lead time constraints.

The operation will **ask** each of the **loads** to verify the capacity availability.

The operation will **ask** each of the **flows** to check the availability of consumed materials.

A load passes on the question and **asks** the **resource**.

The **resource reply** indicates whether the capacity is available or not.

The **load** uses the resource reply to **reply** to the operation.

A flow passes on the question too and **asks** the **buffer**.

The buffer checks the inventory situation.

If material is available no further recursion is required.

If the required material isn't available the buffer will **ask** an **operation** for a new replenishment. Each buffer has a field indicating which operation is to be used to generate replenishments.

Depending on the buffer inventory profile, safety stock requirements, etc... the operation may be asked for different quantities and on different dates than the original demand.

When an operation is asked to generate a replenishment it evaluates the leadtime, material and capacity constraints. This results in a nested ask-sequence similar as the one described earlier - marked with (*)

...

The maximum recursion depth will be the same as the number of levels in the bill-of-material of the end item.

In some cases the iteration can be stopped at an intermediate level. Eg. When sufficient inventory is found in a buffer and no replenishment needs to be asked: a positive reply can be returned immediately.

Eg. When an operation would need to be planned in the past (ie leadtime constraint violated) a negative reply can be returned immediately.

...

The operation collects the replies from all its flows, loads and -indirectly- from all entities nested at the deeper recursion levels. A final **reply** of the **operation** is generated.

Based on the reply of the replenishing operation the **buffer** evaluates whether or not the replenishments are possible, and **replies** back to the flow. Sometimes a buffer may need to ask multiple times for a replenishment before an answer can be returned.

The **flow** picks up the buffer reply and **replies** to the operation.

From the reply of all its loads and flows the **operation** compiles a **reply** and returns it to the demand. The interaction of material, leadtime and capacity constraint are pretty complex and an operation may require several ask-reply iterations over its flows and loads before a final answer can be returned.

The answer of the operation indicates how much of the requested quantity can be satisfied on the requested date.

Depending on the planning result and the demand parameters (such as allow/disallow satisfying the demand late or in multiple deliveries) we can now decide to commit all operation plans created during the whole ask-reply sequence.

If we're not happy with the reply the operation plans created are undone again and we can go back to the first step and ask for the remaining material or at a later date.

The answer in each of the above steps consists of 1) ask-quantity and 2) ask-date.

The reply used in each of the above steps consists of 1) reply-quantity and 2) reply-date. The reply-quantity represents how much of the requested quantity can be made available at the requested date. The reply-date is useful when the ask can not -or only partially- be met: it then indicates the earliest date when the missing quantity might be possible.

In the above sequence the steps are described at a very high level.

In the following sections each of the different ask-reply steps are now explained in further detail.

1. Top level loop
2. Demand solver
3. Buffer solver
4. Operation solver
5. Flow solver
6. Load solver
7. Resource Solver

4.2.1 Top level loop

Delete the existing operation-plans, as far as they aren't locked.
 Identify the clusters to be planned.
 Categorize the demand to be planned by cluster and sort them by priority.
 Create parallel threads for the planning.
 In each planning thread, loop through all demands.

Call demand→solve()

4.2.2 Demand solver

Ignore the demand if quantity is 0
 Erase previous delivery operation plans, except the ones that are locked
 Loop until the full demand quantity is planned.

Call operation→ask(missing quantity,due date), where operation is the demand's or the items delivery operation

If planned quantity = requested quantity, or the demand planning policy allows planning the demand in parts or shorts then

Commit the operation plan creation

Else

Clear the list of scheduled operation plans

If planned quantity > 0 then

// This last step is required to make sure all supplying paths are planned for the quantity of the most constraining path

Call operation→ask(planned quantity, due date)

Commit the operation plan creation

Update the planned quantity for the next iteration in the loop

Exit the loop if the demand can't be planned late

4.2.3 Buffer solver

Standard buffer

Buffer is asked for a quantity Q at the date D For each flow_plan on the buffer

If the on-hand value is positive

Set the variable ExtraInventoryDate if it is not set before. This variable stores the date when there is additional, unallocated inventory available.

Else if the on-hand value is negative

Compute the shortage as current onhand required minimum quantity + known shortage from previous dates

If a producing operation exists

Try to get extra supply for the shorted quantity. This replenishment will update the onhand value of the current flowplan

If the onhand is still less than the required minimum quantity - the known shortage

This situation happens when the producing operation can't replenish the buffer enough, or when all supply in a buffer without producing operation has been exhausted.

Increase the variable storing the known shortage at previous dates.

Reset the ExtraInventoryDate if it was set.

If there is a shortage, a producing operation exists and the above loop didn't already do the following

Try to get more supply at the requested date.

Not only can this reduce the shortage, but also important is the next-date returned by the producing operation.

Note that if this step creates more supply to meet the demand, that supply is not positioned such that inventory is minimized. The flowplan loop does minimize the inventory by replenishing only when the inventory drops below the minimum.

The final results are now:

Returned quantity: requested quantity shortages

Returned date:

= requested date if there is no shortage

Or = reply date of the producing operation

Or = ExtraInventoryDate if that is less than the operation reply date

todo Not up to date with the pre-op time loop...

Infinite buffer

Always reply for the full quantity.

4.2.4 Operation solver

Fixed time and time-per operation

Operation is asked for a quantity Q at the date D

Create required operation plan descriptor

Loop backward in time D until we have found the full resource capacity

Call Operation→ask(Qremaining, Dupdated)

For each consuming flow

Ask the buffer for the planned quantity on the requested date

Update Qremaining and Dupdated

Return the accumulated promise quantity

@todo incomplete documentation: need description of leadtime constraints + flowplan call + loadplan call

Alternate operation

Operation is asked for a quantity Q at the date D

Remaining quantity = Q

Next ask date = infinite future

Loop through all alternate sub operations

Create top operation plan descriptor

Call Operation→ask(Remaining quantity, D)

If some quantity could be planned along the alternate

Check for material and capacity constraints on the top operation plan

Reduce the remaining quantity

Break out of the loop if the requested quantity is completely planned

Else

If the next ask date of the alternate is less than the current minimum, update the next ask date

Return the planned quantity and the next ask date

Routing operation

Operation is asked for a quantity Q at the date D

Create the top operation plan

Check the flowplans and loadplans of the top operation plan


```

Initialize Q2 to Q and D2 to D
For all steps of the routing
    Call operation→ask(Q2,D2)
    Update Q2 if planned quantity < Q2
    Update D2 with the operation time
    
```

4.2.5 Flow solver

If the requested date is outside of the effective date range of the flow, reply for the full requested quantity.
 (@todo this date range isn't implemented yet in the flow model, and the check isn't implemented yet)
 Otherwise, ask the buffer to generate the reply for the quantity and date.

4.2.6 Load solver

If the requested date is outside of the effective date range of the load, reply for the full requested quantity.
 (@todo this date range isn't implemented yet in the load model, and the check isn't implemented yet)
 Otherwise, ask the resource to generate the reply for the quantity and date.

4.2.7 Resource Solver

Standard resource

```

An operationplan is asked to be checked for capacity problems (NO date & quantity)
Set AllLoadsOkay to true
Loop through all loadplans of the operationplan
    If this is not an ending loadplan, move on to the next loadplan
    Call the resource solver
    Set HasOverload to false. (*)
    While HasOverload is still false and not yet at the very start
        Start recursing backwards in the timeline starting from the ending loadplan
        If the resource loading > maximum
            Continue going back till the resource loading < maximum
            Move the operation plan to end at that time in the time-
            line
            Set AllLoadsOkay to false.
            Go back to the step marked with (*)
    
```

Else if we have arrived at the loadplan at the start of the operationplan

Exit the resource solver function

If during the call of the constraint solver the operation plan is moved the variable AllLoadsOkay will have been set to true. In this case, the complete loop over all loadplans must be repeated.

Infinite resource

Always reply for the full quantity.

4.3 Cluster and level algorithm

Resources, operations and buffers are connected with each other with loads and flows. An operation has a collection of loads and flows. Each flow establishes a connection with a buffer, and each load a connection with a resources. The entities thus constitute a network graph. In this network context we define clusters and level as follows.

A **cluster** is a set of connected entities. When a network path across loads and flows exists between 2 entities they belong to the same cluster. When no such path exists they are effectively situated in independent sub-networks and clusters.

Internally, each cluster is represented by a number.

Clusters allow us to group entities and enable multithreading: since the clusters are completely independent we can use different threads to solve each cluster as a separate subproblem.

Material flows in the network have a direction. This creates a sense of direction in our network which is expressed by the **level** concept.

An operation consumes and produces material, as defined by the flow entities (aka bill of material or recipe). In this context the level is a number that is defined such that the level of a consumed material buffer is always higher than the level of the produced material buffer. The demand is normally (but not exclusively!) placed on the material buffers with level 0, and the level number increases as we recurse through the different levels in the bill of material.

Raw materials have the highest level number

The level and cluster number are helpful for the various solver algorithms. They provide valuable information about the structure of the network.

todo add picture to illustrate

The algorithm used to compute the level and cluster information is based on a walk through the network: We select an unmarked operation and recurse through the loads and flows to find all connected entities, updating the cluster and level information as we progress.

For efficiency, the algorithm is implemented as a lazy function, i.e. the information is only computed when the user is retrieving the value of a level or cluster field. The algorithm is not incremental (yet), but computes the information for the complete network in a single pass: a change to a single entity will trigger re-computation of all level and cluster information for all entities.

Note: An updated algorithm has been designed for the cluster computation. Its advantage compared to the current implementation is a much better efficiency in the case of frequent model updates. The computation will be completely incremental, compared to the single pass for all entities in the current implementation.

The detailed flow of the algorithm is as follows:

```
// Initialisation
Lock the function
Reset the level and cluster to  $-1$  on all resources, operations and buffers
Reset the total number of clusters

// Main loop
Loop through all operations
    If the operation has no producing flow
        Activate the level computation
        If the operation isn't part of a cluster yet
            Activate the cluster computation
            Increment the cluster counter
        If both cluster and level computation are inactive, move on to the next operation
        Push the current operation on the recursion stack, with level 0 or  $-1$ 
        Loop until the stack is empty
            Pop an operation from the recursion stack
            Pop the value of cur_level from the stack
            Loop through the sub operations and super operations
                If their level is less than the current level
                    Push sub operation on the stack, with the same level as the
                    current operation
                    Set the level and cluster fields
                Else if cluster is not set yet
                    Push sub operation on the stack, with  $-1$  as the level
                    Set the cluster field
            Loop through all loadplans of the operation
                If level search is active and the resource level is less than the level
                of the current operation
                    Update the level of the resource
                    If the cluster of the resource is not set yet
                        Set the cluster of the resource
```

```

        Loop through all operations that are loading the re-
        source
        If operation cluster isn't set yet
            Push the operation on the stack, level - 1
            Set the cluster of the operation
    Loop through all flows of the current operation
    If this is a consuming flow and level_search is active and the level
    of the buffer is less than the current level +1
        Level recursion is required
        If level recursion is required or the cluster of the buffer is not
        set yet
            Set the cluster of the buffer
            Loop through all flows connected to the buffer
            If it is a consuming flow and level search recursion
            was enabled
                todo incomplete documentation
// Catch buffers missed by the main loop
Loop through all buffers which don't have any flow at all.
    Increment the total number of clusters
    Set the cluster number to the new cluster
// Catch resources missed by the main loop
Loop through all resources which don't have any load at all.
    Increment the total number of clusters
    Set the cluster number to the new cluster
// Finalization
Unlock the function

```

CHAPTER

5

Modules

Frepple can easily be extended with modules that are loaded at runtime.
This chapter describes the modules that are provided with frepple.

The C++ code required to create a custom module is described in the developer section of this manual: Extension modules

An example is also available in the Test Sample Module

1. Python Module
2. Forecast Module
3. Linear Programming Solver Module

5.1 Python Module

This module implements an embedded interpreter for the Python language.
Using the module the full capabilities of this scripting language are accessible from Frepple.
The Python module has access to the Frepple object in memory.

The module enables the following extensions to the standard XML formats:

- `COMMAND_PYTHON` (p 71) is a command that executes your Python code in the embedded interpreter.
- The XML Processing instruction `PYTHON` (p 72) is an elegant way to define python functions in your XML-file.

5.1.1 COMMAND_PYTHON

The command allows you to run Python code in the interpreter that is embedded in this frepple module.

The interpreter can execute generic scripts, and it also has access to the frepple objects.

The interpreter is multi-threaded. Multiple python scripts can run in parallel. Internally, Python allows only one thread at a time to execute and the interpreter switches between the active threads, ie a quite primitive threading model.

Frepple uses a single global interpreter. A global Python variable or function is thus visible across multiple invocations of the Python interpreter.

| Field | Type | Description |
|----------|-------------------|---|
| CMDLINE | String | Python command to be executed. |
| FILENAME | Normalized string | Filename with Python commands to be executed. When both the CMDLINE and FILENAME fields are filled in only the CMDLINE Python code will be executed. |
| VERBOSE | Boolean | Echo information about the command execution in the log. |

Example XML structure:

```
<PLAN>
  <COMMANDS>
    <COMMAND xsi-type="COMMAND_PYTHON"
      CMDLINE="print 'Hello World' " />
  </COMMANDS>
</PLAN>
```

5.1.2 Processing instruction PYTHON

Python code can also be included as a XML processing instruction.

Example XML structure:

```
<PLAN>
  <?PYTHON
    def MyFunction():
      print "Hello World"
  ?>
  <COMMANDS>
    <COMMAND xsi-type="COMMAND_PYTHON" CMDLINE="MyFunction()" />
  </COMMANDS>
</PLAN>
```

5.2 Forecast Module

This module for representing forecast as a special type of demand.

Given a calendar and item, the forecast will automatically create a separate demand for each time bucket.

5.3 Linear Programming Solver Module

This module is intended to implement a linear programming solver.

It is currently not functional, and its development is put on-hold till an appropriate time...

CHAPTER

6

Developer

1. Code structure
2. Class diagram
3. Extension modules
4. Portability
5. Version control
6. Style guide
7. Security

6.1 Code structure

This chapter provides a high level description of the code structure.

It provides brief notes that helps a developer find his/her way in the detailed C++ API reference and Class diagram .

Three layers can be distinguished:

- **Utility classes** which provide infrastructure-like services as a foundation for the next layers.
 - Object (p 75) as an abstract base class for all Frepple objects.
 - Metadata (p 75) about objects.
 - Date, DateRange and TimePeriod (p 75) for dealing with dates and times.
 - Timer (p 76) for measuring execution time.
 - XML serialization (p 76) for reading and writing XML data.
 - Command (p 77) for executing state changes.
 - Exception classes (p 76) for reporting error conditions.
 - Mutex and LockManager (p 77) provide support for concurrent access to memory objects in a multithreaded environment.
 - HasName and Tree (p 77) for representing entities with a name and storing them in a binary tree container.

- HasHierarchy (p 77) allows objects be structured in a hierarchical tree, ie to refer to a parent and have children.
 - Leveled (p 77) for representing entities that are connected in a network graph.
- **Model classes** which represent the core modeling objects.
See the chapter Modeling for the details.
They are structured as a base class (or Category) with one or more concrete implementations (or Classes).
- **Extension classes** which inherit from the core model classes and implement specific new models or solver techniques.
See the section Extension modules for more details.

6.1.1 Object

Object is an abstract base class.

It handles to following capabilities:

- **Metadata:** All subclasses publish metadata about their structure and the memory they consume.
- **Concurrency:** Locking of objects is required in multithreaded environments. The implementation of the locking mechanism is delegated to the LockManager class, and this class provides only a pointer to a lock object and convenience guard classes.
- **Callbacks:** When objects are created, changing or deleted, interested classes or objects can get a callback notification.
- **Serialization:** Objects need to be persisted and later restored.
Subclasses that don't need to be persisted can skip the implementation of the writeElement method.

6.1.2 MetaData

Frepple uses a two level structure to group metadata:

- A **MetaCategory** represents an entity type. The metacategory will implement a container for all instances of this type, and also a handler method to control persistence of the objects.
E.g. "Buffer"
- A **MetaClass** represents a concrete class. It belongs to a certain MetaCategory, and contains a factory method to generate objects.
E.g. "BufferDefault", "BufferMinMax", "BufferInfinite"...
- **MetaData** is the abstract base class for the concrete class MetaClass and MetaCategory.

After creating an MetaClass or MetaData object it needs to be registered, typically in the initialization of the library.

6.1.3 Date - DateRange - TimePeriod

These classes allow easy and intuitive manipulation of dates, durations and date ranges.

The classes are implemented as a thin wrapper around the standard ansi C time functions and provides time accuracy of 1 second.

An example:

```
Date start = Date::now();
TimePeriod duration("24:00:00");
Date end = d + t;
DateRange dr(start, end);
cout << d << " " << t << " " << dr << endl;
```

The C library is respecting daylight saving time (DST). Depending on the timezone configured on your computer, you will have two days a year which last 23 or 25 hours instead of the regular 24 hours.

This means that “midnight on day 1” + “24 hours” will not always give you “midnight on day 2”!

6.1.4 Timer

This is a class to measure the execution time of the application with (at least) millisecond precision. An example:

```
Timer t;
do_something();
cout << "something took " << t << " seconds" << endl;
t.restart();
do_something_else();
cout << "something else took " << t << " seconds" << endl;
```

6.1.5 Exception

Frepple uses 3 exception classes to report errors. Each of the classes inherits from `std::exception`.

- A **DataException** is thrown when data errors are found.
The expected handling of this error is to catch the exception and allow the execution of the program to continue.
- A **RuntimeException** is thrown when the library runs into problems that are specific at run-time.
These could either be memory problems, threading problems, file system problems, etc... Errors of this type can be caught by the client applications and the application can continue in most cases.
- A **LogicException** is thrown when the code runs into an unhandled and unexpected situation.
The normal handling of this error is to exit the program, and report the problem. This exception always indicates a bug in the program code.

6.1.6 XML Serialization

The Object base class provides the following methods that need to be implemented by serializable classes:

- The **beginElement** is called by the parser when reading the start of a tag.
- The **endElement** event is called by the parser when reading the end of a tag or attribute.
- The **writeElement** is called when serializing the object.

Frepple uses the SAX parser from Xerces-C to parse and validate input XML data.

The class **XMLInput** is a wrapper around the parser. It receives the SAX events and makes the appropriate calls to the frepple objects.

Subclasses are available to parse a file or a string.

Writing XML output is done with the **XMLOutput** class which provides methods to write a header, elements and attributes. Subclasses are available to write to a file or a string.

6.1.7 Command

This class implements the design pattern with the same name. All state changes in the application are expected to be encapsulated in objects of this class.

The **CommandList** class works as a wrapper for a collection of other commands, following the classic composite design pattern.

This allows command hierarchies to be constructed, which can be executed in sequence or in parallel.

Quite a few subclasses are available: see the command modeling or the C++ API reference.

6.1.8 Mutex and LockManager

Working with Frepple in a multithreaded environment requires special control over concurrent access to the objects in memory.

- **Mutex** allows exclusive access to a object.
Depending on your platform it is implemented as a thin wrapper around a Windows `critical_section` or as pthread `pthread_mutex_t`.
- **LockManager** controls the locks on objects.
Only an empty implementation is currently provided, but an implementation using a ‘multiple read - single write’ lock will follow.
- The **CommandList** (described above) has the capability to execute commands in parallel by spawning separate threads.

6.1.9 HasName and Tree

The classes represent classes which use a `std::string / name` as a unique identifier.

The **Tree** class is implemented as a red-black binary tree, using **HasName** objects as the nodes (i.e. intrusive container).

6.1.10 HasHierarchy

The class allows objects be structured in a hierarchical tree. A **HasName** object can point to a single parent and it maintains a linked list of children.

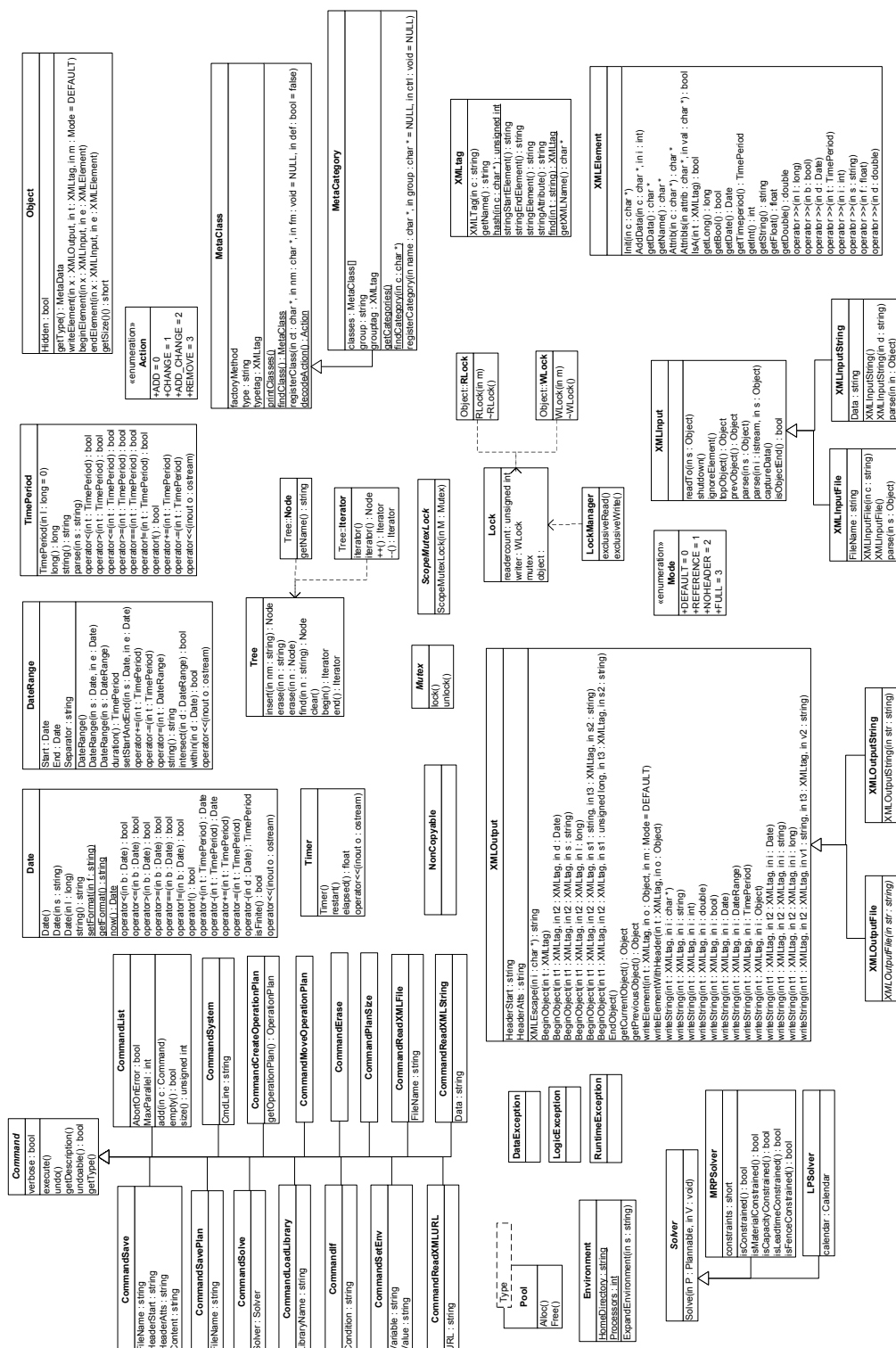
6.1.11 Leveled

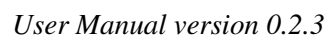
The model classes **Operation**, **Buffer**, **Resource**, **Load** and **Flow** are the key objects that are used to represent the network.

The first three represent the actual entities, while Load and Flow represent associations/links between the entities.

See the section Cluster and level algorithm for the details.

6.2 Class diagram





6.3 Extension modules

Frepple is designed as an extendable library.

Additional modeling and solver modules can be loaded at runtime without recompiling the library. Such extension modules can be shipped with Frepple, or can be developed by third parties. Modules can be open source or have a commercial license.

An simple example is available in the testcase sample_module.

Frepple currently includes three examples of such extension modules: a module implementing a python interpreter, a forecast class implementing a special type of demand, and a solver using a linear programming algorithm.

The steps below define how a custom extension can be build on the framework.

- The proper way to build extension is by creating modules.
Other ways of extending the package may technically be possible, but are not recommended. Copying the code and header structure from an existing module is the quickest and easiest start.
- Create your own header files, and include the frepple header file planner.h to have access to the frepple objects.
A simple header file can look like this:

```
#include " frepple.h"
using namespace frepple;

namespace your_module
{
    MODULE_EXPORT const char* initialize(const CommandLoadLibrary::ParameterList& z);
    ...
    your classes and function definitons
    ...
}
```

- Create your own c++ implementation files, which will include you customized header file.
It is important is to include an initialize() method, and use it to register your extension in the Frepple framework. The method is automatically called when the module is loaded.

```
#include " your_module.h"
namespace your_module
{

MODULE_EXPORT const char* initialize(const CommandLoadLibrary::ParameterList& z)
{
    ...
    your initialization code goes here
    ...
}

your method and class implementations go here
```

- Compile your code as a loadable module.
The command line options and arguments vary for each compiler and platform. For gcc I use

the options “-module -shrext .so -avoid-version -rpath /dev/null”, adding also “-no-undefined” when running under cygwin.

To keep things simple and transparant please use the .so extension for you modules and place them in the \$FREPPLE_HOME directory.

- Update the \$FREPPLE_HOME/init.xml file to load your module with a COMMAND_LOADLIBRARY tag.

Parameters specified with the PARAMETER tag are passed to the initialize() funtion when the module is loaded.

- Update the file \$FREPPLE_HOME/frepple.xsd by defining the xml constructs enabled by your module.

To keep things clean and modular, it is recommended to do this by including a seperate xsd file rather than directly entering the definition in the file.

6.4 Portability

The project is currently compiled and tested only for 32-bit linux and Windows environments, with Linux being the primary development platform. Porting to other platforms is encouraged - you'll have all my support in helping with this.

Here are some areas where porting may be a bit challenging:

- Availability of a modern C++ compiler and STL.
- File system functions such as fstat, paths, directory listings
- Availability of the Pthreads library for threading.

Frepple currently only supports the Windows threading functions and the Pthreads.

- Shared libraries

Currently the code only supports the dlopen (Solaris, Linux and various BSD flavors) and LoadLibrary (Windows) functions. HPUNIX uses different function name shl_load, while AIX doesn't allow shared libs depending on other shared libs.

- Availability of the Xerces-C XML parser.
- Availability of the Python language.

6.5 Version control

The software changes are tracked with subversion on the Sourceforge site.

The repository is accessible to everybody and can be browsed online. Complete instructions are available on *this page*¹.

A example subversion configuration is available in the file subversion.config for convenience. In particular the section on the automatic properties is of interest when adding files to the project.

6.6 Style guide

To enforce the same formatting of the source code the astyle tool is used.

See <http://astyle.sourceforge.net/> for more information.

¹ sourceforge.net/svn/?group_id=166214

The following formatting options are used:

```
-- style=ansi
-- indent=spaces=2
-- indent-classes
-- indent-switches
-- min-conditional-indent=2
-- one-line=keep-statements
-- one-line=keep-blocks
-- max-instatement-indent=2
-- convert-tabs
```

Astyle does a pretty decent job, but reviewing the astyle changes before committing them is still required: astyle sometimes misses the point. . .

6.7 Security

When Frepple is used in a networked multi-user environment, security is very important. The Frepple C++ code is developed with security in mind.

Here are some notes and considerations on this topic:

- Frepple can validate incoming XML data with an XML-schema. Invalid data will be rejected and an error message is generated.
The default xsd files `frepple.xsd` and `frepple_core.xsd` cover all valid structures.
When integrating Frepple with other systems it is strongly recommended to validate the incoming XML data against a small and well-controlled subset of the default XML-schema.
- The `COMMAND_SYSTEM` and `COMMAND_IF` commands allow execution of arbitrary shell commands with the privilege of the user running the Frepple executable.
While allowing a maximum of flexibility for configuring and customizing Frepple, it also creates an open door to access your system. Access to this command should be restricted, and/or frepple should be run by a user account with limited privileges.
- The `COMMAND_PYTHON` command allows execution of arbitrary python commands with the privilege of the user running the Frepple executable.
While allowing a maximum of flexibility for configuring and customizing Frepple, it also creates an open door to access your system. Access to this command should be restricted, and/or frepple should be run by a user account with limited privileges.
- The `COMMAND_SETENV` command allows environment variables to be updated.
Access to this command should be restricted, as it can alter the behavior of the system.
- When using Django, its standard web authentication mechanism is relatively weak.
In secure environments consider plugging in a different login mechanism.

CHAPTER

7

Samples and tests

This pages documents the examples available in the ‘test’ subdirectoy. The examples can be categorized in the following functional categories:

- Unit tests, which verify the behavior specific parts of the code.
- Performance tests, which focus on the performance (memory and/or cpu-time).
- Samples, which provide more real-life usage of the tool.

1. Test Callback
2. Test Cluster
3. Test Command 1
4. Test Command 2
5. Test Command 3
6. Test Constraints Leadtime 1
7. Test Constraints Material 1
8. Test Constraints Material 2
9. Test Constraints Material 3
10. Test Constraints Resource 1
11. Test Constraints Resource 2
12. Test Constraints Resource 3
13. Test CSV
14. Test Datetime
15. Test Deletion
16. Test Demand Policy
17. Test Demo 1
18. Test Forecast
19. Test LP Solver 1
20. Test Name
21. Test Operation Effective
22. Test Operation Pre Op
23. Test Pegging

24. Test Python 1
25. Test Python 2
26. Test Problems
27. Test Procure 1
28. Test Safety Stock
29. Test Sample Module
30. Test Scalability 1
31. Test Scalability 2
32. Test Scalability 3
33. Test Sizeof
34. Test XML
35. Test XML Remote

7.1 Test Callback

This test verifies the event publishing and subscription mechanism.

7.2 Test Cluster

This test verifies the correctness of the clustering algorithm. A network is built with a whole range of possible interconnections between operations, buffers and resources.

7.3 Test Command 1

In this test commands are being run in parallel and in sequence. The proper branching and merging of the tasks is verified, and the behavior in case of errors.

7.4 Test Command 2

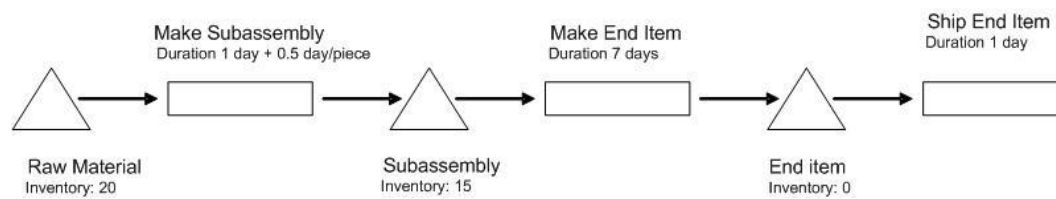
In this test conditional execution of commands is tested.

7.5 Test Command 3

Verifies how environment variables are set and their values expanded.

7.6 Test Constraints Leadtime 1

This test verifies the solver behavior for leadtime constraints. Demands are placed on the network such that operations are planned in the past in the unconstrained plan. Demands are appropriately shorted or planned late in the constrained plan to solve the problems.



A first order for 7 units is due on day 3 after the current date.

It is planned to be delivered late on day 8: the production of the end item starts on the current date, and takes 7 days. The delivery takes an additional day.

A second order for 14 units is due on day 11.

The inventory of the subassembly is now depleted and 6 new subassemblies need to be produced. These subassemblies are due on day 3.

In the 2 days between the current date and the due date of the subassemblies 2 units can be produced. There are 3 subassembly operations are planned in parallel, each for 2 units, starting on the current day and finishing on day 3.

Sufficient raw material is available in inventory for the subassemblies.

The order is delivered on time.

7.7 Test Constraints Material 1

This test verifies the behavior of the buffer solver for the case where no producing operation is defined.

Four variations of a base scenario are tested:

- 3 consumers, ordered in chronological order
- 3 consumers, not ordered in chronological order
- extra supply arriving at a different date, causing a late order
- extra supply arriving at a different date, but already partially used up

7.8 Test Constraints Material 2

@todo

7.9 Test Constraints Material 3

@todo

7.10 Test Constraints Resource 1

A simple capacity problem that can be resolved by moving operation plans early.

7.11 Test Constraints Resource 2

A capacity shortage where operation plans are moved earlier till they are in the past. The associated demands are then shorted.

7.12 Test Constraints Resource 3

A capacity problem where a single operation loads multiple resources. This test case also has capacity limits varying over time.

7.13 Test CSV

In this test the capability of reading data from CSV-formatted data files is verified. This functionality is implemented as a Python function that is reading the data file and then creating a XML-document from it.

7.14 Test Datetime

Frepple uses some wrapper classes around the C date and time functions. These are tested here: conversions to and from strings, additions, ...

7.15 Test Deletion

This test verifies the capability to delete parts of the model. After loading the model different entities are one-by-one being deleted. After each delete we replan and save the model to make sure the deletion is working correctly: an incorrect delete would crash the application!

7.16 Test Demand Policy

The test verifies the demand policies.

The supply situation is such that half of the demand can be met in time, and half of it late:

- Demand: 20 on due date 5 Jan
- Supply: 10 available as inventory, and 10 arriving on 10 Jan

The demand policy controls how the demand is allowed to be planned in such a constrained situation:

- Case A: PLANLATE MULTIDELIVERY
This is the default policy. It allows demands to be planned late and to be satisfied in multiple parts.
Result: Delivery of 10 units on 5 Jan and a second delivery on 10 Jan.

- Case B: PLANSHORT MULTIDELIVERY
No lateness is allowed, and demands can be partially met.
Result: A delivery of 10 units on 5 Jan.
- Case C: PLANLATE SINGLEDELIVERY
Lateness is allowed, but the order is met in a single delivery.
Result: A delivery of 20 units on 10 Jan.
- Case D: PLANSHORT SINGLEDELIVERY
Not lateness is allowed, and a partial delivery is not accepted.
Result: No delivery planned.

7.17 Test Demo 1

This test was intended to be a simple real-life model, but is currently not really useful. . .
todo Either clean up this test or remove it. . .

7.18 Test Forecast

This test verifies the behavior of the FORECAST extension module.

7.19 Test LP Solver 1

This test verifies the behavior of the linear program solver module. If you compiled the application without support for this solver this test will fail, which is nothing to worry about.

The lp_solver module isn't ready at all. . .

7.20 Test Name

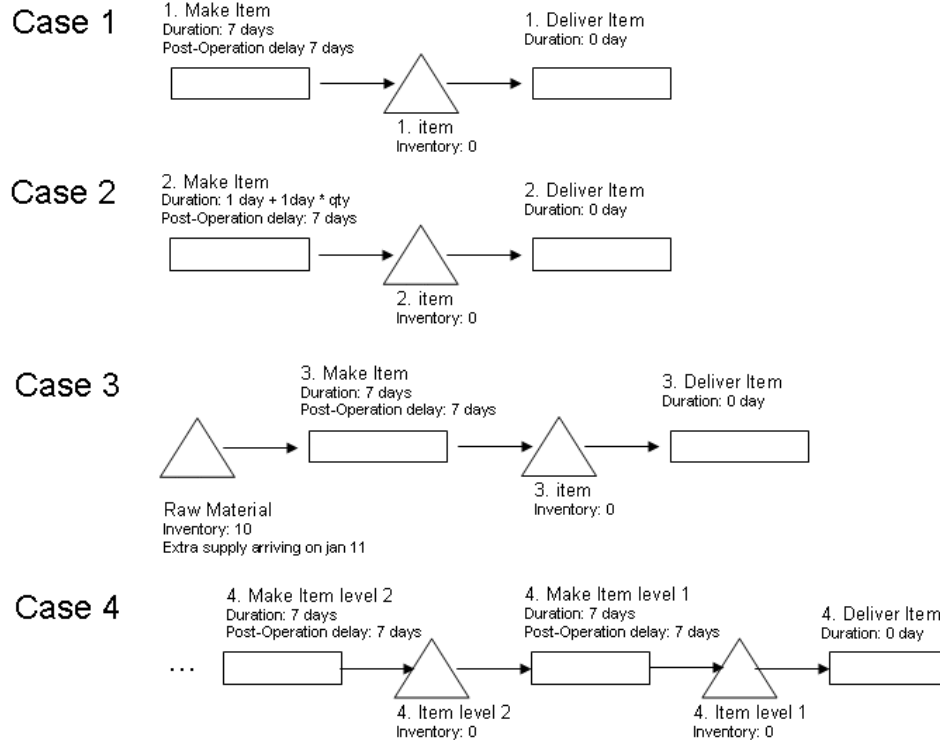
This test reviews the data structure that is used for storing all named entities: functionality of the insertion, deletion and search operations, as well as their scalability. The time for these operations properly fits a logarithmic profile, as expected with a binary tree data structure. A testing routine for this profile is also included in the test, but it isn't part of the regression tests since it isn't easy to produce a good pass-fail criterion.

7.21 Test Operation Effective

This test checks the code for creating and manipulating of operationplans of type "effective". The test resolves about a product that can be sourced from two locations. In a first part of the horizon location A is the only allowed source, while in the last part of the horizon only sourcing from location B is possible. A transition period exists where sourcing from both location is possible.

7.22 Test Operation Pre Op

This test verifies the behavior of pre-operation and post-operation delays. These are as delay times before and after an operation, which the solver tries to respect but can violate if required.



Several cases are included in this test:

1. Post-operation time on a fixed-time operation.
The post-operation time is respected when possible, but when running against a leadtime constraint the post-operation time is reduced to meet the demand on-time / asap.
2. Post-operation time on a time-per operation.
The constraint is again a leadtime constraint.
3. Post-operation time on fixed-time operation.
This time the constraint is the late supply of raw material supply. It causes the post-operation time to be reduced.
4. Post-operation time on multiple levels in the supply path.
The supply path is four levels deep, and a post-operation time is set at each level.
In case of material i.e. leadtime constraints the post-operation time on the most upstream operation/operations (i.e. operations deeper in the bill of material) is/are shrunk first.

7.23 Test Pegging

Verifies the correctness of the material pegging. Material streams are traced upstream and downstream and printed to the output.

7.24 Test Python 1

This test verifies and demonstrates the embedded Python interpreter.

No pass/fail criterion is present in this test.

7.25 Test Python 2

Verifies and tests the access to the frepple objects from Python.

7.26 Test Problems

Verifies that problems objects are created and deleted properly when the model is being updated in various ways.

7.27 Test Procure 1

This unit test verifies the behavior of procurement buffers in a number of scenario's.

The different cases are:

1. Base scenario.
2. Procure in multiples.
3. Procurement with minimum size, maximum size and in multiples.
4. Invalid parameters for size constraints.
5. Procurement with minimum and maximum interval.
6. The full monty. Procurement with minimum interval, maximum interval, minimum size, maximum size and in multiples.
7. Procurement with fixed interval.
8. Procurement in fixed quantity.
9. Procurement in fixed quantity with fixed interval.

In all these cases the demand is directly placed on the procured item (i.e. no bill of material is involved at all) and the demand pattern is also identical.

The test runs first an unconstrained plan, followed by a constrained plan.

7.28 Test Safety Stock

This test demonstrates the capabilities to model and plan safety stocks in frepple.

There are 2 ways:

1. Quantity-based safety stock.

A minimum calendar on a buffer defines the desired minimum stock level, which can vary over time.

The solver tries to replenish to this level when replenishing the buffer, but handles it as a soft

constraint only.

The buffer flags a problem when the inventory drops below the minimum target.

2. Time-based safety stock.

A post-operation time on an operation defines a time delay after the end of the operation.

The solver tries to respect this delay, but handles it as a soft constraint only.

No problem is shown when the post-operation time is shrunk or reduced.

7.29 Test Sample Module

A simple example on how to define an extension module for Frepple.

The example defines a new operation type that can be used to represent transportation operations easier.

7.30 Test Scalability 1

Tests the scalability of the data loading, running an MRP plan (including the clustering algorithm) and saving the plan. The network in this case consists of a lot of parallel clusters, which can be solved in parallel. See also the test scalability_2

The algorithms scale linearly with the model size, while the mayor underlying data structures are binary trees which scale logarithmically with the model size. . . The result is a runtime that combines both. In summary, one could say that the system scales a bit worse than linear, but definately not quadratic or worse

todo picture

7.31 Test Scalability 2

In this test a model is created based on parametrizable values of:

- Number of clusters.
- Number of demands per cluster.
- Depth of the supply chain, i.e. number of levels.

Comparing the runtime with different values of these parameters allows to gain a better understanding of the factors that are impacting memory and runtime most significantly

The algorithms scale linearly with the model size, while the mayor underlying data structures are binary trees which scale logarithmically with the model size. . . The result is a runtime that combines both. It depends on the data set, the platform and the compiler how your model will scale.

7.32 Test Scalability 3

This test is designed to verify the scalability of the timeline data structure. The network consists of a single buffer with a very simple operation producing into it. Since the timeline data structure is currently based on a linear list the scalability of the timeline is expected to be bad. . . A quadratic

increase in the runtimes can be observed. A more scalable data structure has been designed to provide a more scalable implementation.

7.33 Test Sizeof

Echoes the size of the main model classes.

The results are based on compiling using gcc under linux on a 32-bit processor. Different compilers and platforms can easily use a different organization and size of the data structures, which will give different test results. A failure of this test is thus nothing to worry about.

7.34 Test XML

This is a test for the XML parser routines. The test consists of a complex xml document to be parsed and processed:

- XML tags 8 nested levels deep
- ignore-element sections

7.35 Test XML Remote

This test uses HTTP to pick up XML-data from the url http://frepple.sourceforge.net/test/xml_remote.xml.

CHAPTER

8

Appendices

1. GNU Lesser General Public License
2. GNU Free Documentation License

8.1 GNU Lesser General Public License

Version 2.1, February 1999

Copyright © 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that

you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the “Lesser” General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

- c) Accompany the work with a written offer, valid for at least three years, to give

the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the

conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH

YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

8.2 GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, \LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in

this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be

listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the

copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright © YEAR YOUR NAME.  
Permission is granted to copy, distribute and/or modify this document  
under the terms of the GNU Free Documentation License, Version 1.2
```

or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled " GNU
Free Documentation License" .

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.