



Developers Manual

Release 3.1

November 24, 2004

Track+ License

Track+ has been published as open source to facilitate modifications. The copyright © is with the Track+ Project Office.

Redistribution and use of the software described herein in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer listed below in the documentation and/or other materials provided with the distribution.
3. Redistribution of the any modified work is limited to the distributors organization.
4. Any derivative work may not use the name and logo of **Track+** or TrackPlus as a product name or to endorse or promote products derived from this work without specific prior written permission by the copyright holders.
5. There are not more than 10 users in the **Track+** database, or it is being used on an open source project, or in a charity, or in an educational institution. Other users should obtain a commercial license (see <http://www.trackplus.de>).

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Track+ is also available as a commercial product. If you plan to contribute to the development of **Track+** please consider the following:

- If you make significant contributions over a period of at least one year you are entitled to receive a share of all proceeds from selling **Track+** licenses. The share has to be negotiated with the Track+ Project Office (see <http://www.trackplus.org>). You will be informed about the number and kind of licenses sold on a regular basis. If you have made a singular, significant contribution you will receive some reward depending on the amount licenses sold.
- If you have not made any significant contributions for more than one year, your share will drop and has to be renegotiated with the Track+ Project Office on a yearly basis.

- Significant contributions are those that encompass more than 10% of the code base or equivalent amounts of work in testing, support or documentation.
- The copyright of this software stays with the Track+ Project Office.
- If you significantly contribute on your company time, or for the purpose of obtaining an academic degree, your company or institution gets free access to **Track+** for the year the contributions took place, and two years beyond that. However, in this case you will not be entitled to a share of any proceeds any more.

Contributors

Track+ was mostly conceived and written by Jörg Friedrich. The following people have significantly contributed to the development of **Track+** :

- Knut Erik Ballestad
- Maxym Dmytrychenko
- Clemens Fuchslocher
- Alexei Khatskevich
- Hartmut Lang

The following people have contributed to **Track+** in some way:

- Christoph Bräuchle
- Clemens Edel
- Philip Eisenhardt
- Andreas Kaffer
- Marcin Sobieszek

1

Development Process

1.1 Getting Started

The **Track+** project is hosted on its own server and on Sourceforge. In order to actively participate in the development process you will need to go through the following steps:

1. Download the current version of **Track+** (binary and database) and get it to run on your development machine. How to do this is described in the **Track+** user manual which you can also download.
2. Once you have a running **Track+** system you have to set up CVS on your development machine. Follow the instructions on SourceForge and access the **Track+** CVS repository first as user “anonymous”.
3. Download module “track” from the CVS repository to a convenient place on your machine.
4. Download the Torque generator package, ANT and httpunit, and install it in a convenient place on your machine.
5. Adapt the appropriate paths in file `build.bat` in the **Track+** repository root directory so that they show to the places where you have just installed the different packages. This holds only for a Windows development environment. If you are developing under Unix, you may have to write your own build shell script.
6. Run `build.bat all`. You should not get any errors.
7. Modify `motd.jsp`, run `build.bat compile` and start your application server. You should see your modified message of the day if you call context `track` on your application server.
8. Once you have everything running, apply for a user account on the Track+ server and send an email with a short skill resume and area of interest to `support@trackplus.org`. You will then be added to the **Track+** developers list. To be able to work on the Track+

server (`cvs.trackplus.org`) you need to generate an SSH2 DSA key pair for CVS. How this is done is described in the next section. The configuration of your CVS client is the same as that for the Sourceforge CVS, except for the client address of `cvs.trackplus.org` instead of `cvs.sourceforge.org`

9. Everybody can access the **Track+** source code via http through <http://www.emron.org/cgi-bin/viewcvs.cgi>. Your CVSROOT should look something like `yourLogin@cvs.emron.org:/cvsroot/trackplus`.

1.2 CVS Configuration

1.2.1 Prerequisites

These instructions assume that you already have installed the CVS software and the OpenSSH client (e.g. `putty` on Windows systems). They make no attempt to tell you how to go about this; contact your local technical support for details, or chase down instructions on the appropriate website.

We also assume a reasonable familiarity with the Unix or Windows user environment; you should be able to ensure that environment variables are set to appropriate values and be able to edit a file, change file permissions, and so on.

In order to access the Track+ CVS server, you need to send a **DSA public key file** to `support@trackplus.org`. This is what you need to do.

1.2.2 Creating an SSH Identity

This is simplicity itself. For a Unix system, at the shell prompt, issue the following command:

```
$ ssh-keygen -d
Generating DSA parameter and key.
Enter file in which to save the key (/home/joeb/.ssh/id_dsa):
Created directory '/home/joeb/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/joeb/.ssh/id_dsa.
Your public key has been saved in /home/joeb/.ssh/id_dsa.pub.
The key fingerprint is:
ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab:ab joeb@your.host
$
```

On a Windows system, you may use the utility `puttygen.exe`. You need to generate an **SSH2 DSA** key pair. The public key looks something like this:

```
1024 37 929822406516452299586841092475691859400347936015013047
09829195523110430414835833146891760078793291184339732538846972
13536311994569725578746859159486455518645763280420208858022565
46698220240045982570535475899669589195136609307810465935061430
09605401645638279342635639381225799935059809889582359953437340
980461 My key for the Track CVS server
```

Obviously, some of those details (such as your username and the “fingerprint”) will be slightly different. When asked for a filename, just press return to accept the default. Similarly, just hit return when asked for a passphrase to encrypt your private key.

This is not particularly secure; ssh permits you to encrypt your private key with a passphrase. This will prevent anyone who has access to your computer from using your private key to masquerade as you. However, each time you use ssh, you’ll be prompted to enter your passphrase. You can avoid this for instance with the `pageant` program if you are using the putty SSH suite on Windows. For other SSH agents see the respective documentation for details.

The result of this operation will be two files in the `.ssh` directory under your home directory. One will be called `id_dsa`; this contains your private key. The other, `id_dsa.pub`, is your public key; you should email this file to the CVS administrator to enable him to complete the configuration of your account on their server.

Configuring CVS to use SSH for Connection

This is also pretty simple. You will need to set up two environment variables as follows:

```
CVSROOT=:ext:jbloggs@cvs.trackplus.org:/cvsroot/trackplus
CVS_RSH=ssh
```

You can ensure that these variables are always set when you log in by adding the following to your `.profile` script:

```
CVSROOT=:ext:jbloggs@cvs.trackplus.org:/cvsroot/trackplus
CVS_RSH=ssh
export CVSROOT CVS_RSH
```

If you use a csh-derived shell (ask someone if you’re not sure), you’ll need to append the following lines to your `.cshrc` file:

```
setenv CVSROOT :ext:jbloggs@cvs.trackplus.org:/cvsroot/trackplus
setenv CVS_RSH ssh
```

If you are using the Eclipse development environment, go to **Windows**→**Preferences**. Select the screen as shown in Fig. 1.1 and edit everything exactly as shown, except for the path to the `plink.exe` program from the Putty suite; this has to match your local installation.

You’re ready to go

Once you’ve sent your public key to the CVS administrator, you should wait until you get a response indicating that he has completed the remote configuration.

At that point, you’re ready to go. You can check out modules by issuing commands such as:

```
$ cvs co track
```

As a developer, you usually have access to module “track” which contains the sources for the **Track+** project.

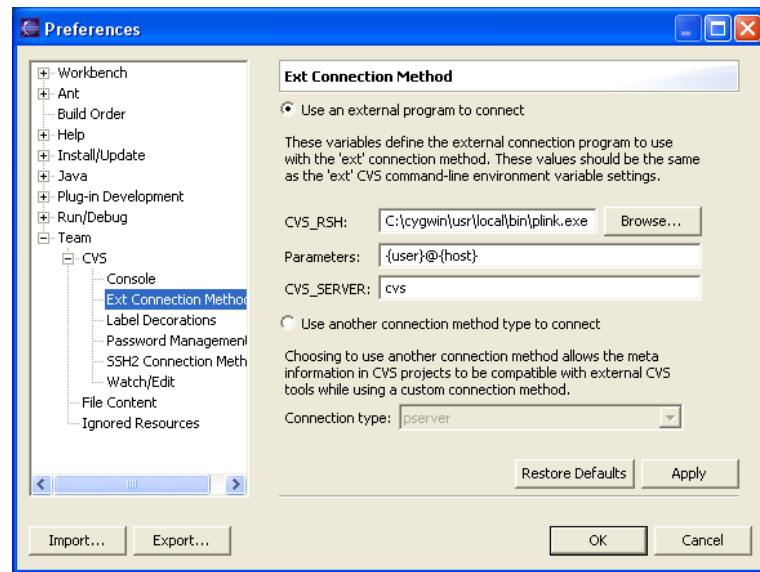


Figure 1.1: Eclipse CVS preferences

1.3 Configuration Management

Configuration management is supported by CVS on SourceForge. Releases are being tagged with tags named REL<major version><minor version><patch level>. Non-final version are called “release candidates” and shall be named by appending a string RC<X> to the original name. Thus, release candidate two of version 2.1.0 would be tagged as REL210RC2. Tags shall only be performed by the project administrator when a release is being published. Following a non-final release (i.e. publishing of a release candidate) there is some time when new features are being implemented while bug fixes have to be done on the release candidate. To manage this requirement a branch will be opened for the development of new features. It will be closed about two months after a final release has been published.

1.4 Development Coordination

For efficiency reasons, bugs should preferably be fixed by those developers that have originally written the code containing them. To indicate that you are working on removing a bug you set the person responsible from “none” to your name in the SourceForge bug list. Once you have fixed the bug and verified the fix you set the bug resolution field to “fixed”. Do not close the bug! Bugs will only be closed when a new release is being published.

The same procedure goes for minor feature requests. You can “grab” a feature request by assigning it to yourself on the feature request list. For major features we should write a little paper illustrating the solution. I would include the suggestion into this document so everybody can see it and offer their opinion.

1.5 The Elements of Style

There is a book “The Elements of Java Style”. Good if you follow its recommendations. My minimum wish list is here:

- Lines shall not be longer than 80 characters
- Try to find good variable and procedure names
- Variable names should start with a lower case letter
- Class names should start with an upper case letter
- Package names should start with a lower case letter
- Constants should be all CAPITAL
- Please use JavaDoc comments at the beginning of classes and public fields and methods
- Do not use wild cards in import statements

That’s all, folks!

1.6 Testing

Track+ is being tested using manual testing and automated testing. Automated testing is performed (on Windows) with script `buildtest.bat all`. There should be no failed tests on a published release. The test uses `httpunit` and directly accesses the database to set the stage for the tests. It is recommended to use a freshly initialized database to run the tests. If no manual manipulation takes place, the tests can be run many times as the test scripts clean the database before every run from any debris of previous test runs.

Automated testing focuses on testing at the user interface, i.e. the browser. To emulate the browser the `httpunit` package is used. If any new screens are being added to the system, new `httpunit` test cases should be written.

In order to be able to check email output, the email output is redirected as comment at the bottom of the next page following an action if the application is in test mode.

Currently, tests should be run on the following systems:

1. Operating systems: Linux, Windows NT, Windows XP, Solaris
2. Browsers: Netscape 4.7 (deprecated), Netscape 7.x, Opera 7.x, Internet Explorer 5.x, Internet Explorer 6.x, Mozilla
3. Database: MySQL 4.x, Firebird 1.0, Oracle 8i, Oracle 9.x, MSSQLServer 7 and 2000, PostgreSQL, HSQL, Sybase
4. Application Servers: Tomcat 4.x, JBoss, Jetty, Websphere, Sun ONE

Testing on any other systems is quite welcome.

1.7 Publishing a Release

Releases will be published by the project administrator using the SourceForge File Release System FRS. Releases shall be named `<major version>.<minor version>.<patch level>`. Non-final version are called “release candidates” and shall be named by appending a string `rcX` to the original name. Thus, release candidate two of version `2.1.0` would be named `2.1.0rc2`.

No release may be published if any test cases are broken. All test cases have to run before a release is being published. Bug reports and feature requests must not be closed before the appropriate release is being published. Release notes describe which bugs have been removed with this release and which features have been implemented. Optionally, there may be a list in the release notes stating which bugs have *not* been closed.

1.8 Style Guide

The style of the **Track+** application is mostly defined the the style sheet `web/common/styles.css`. The following table lists some of the most important style elements.

Element	Value
Blue Track+ color	<code>(#1d 50 90), rgb(29, 80, 144)</code>
Body background	<code>(#ff fc f2), rgb(255, 252, 242)</code>

Table 1.2: Some style elements

2

New Features

2.1 Overview

The features described in this manual will not all be implemented with the next release. They just serve as a collection of ideas of the development team and the many users that send in their feature requests. Table 2.2 gives an overview of the features planned and their priority as seen by the development team. Most likely, features will be implemented in this order. Feature requests by paying customers are given high priority. We like to consider the following criteria in deciding which features to give priority, in this order:

1. Does the organization using *Track+* benefit from it as a whole?
2. Does the regular user have a functional benefit from it?
3. Does the administrative user have a functional benefit from it?
4. Does it improve the markets perception of *Track+* ?
5. Does it improve aesthetic appearance and GUI efficiency?
6. Does it reduce the maintenance effort for the development team and lead to a cleaner design?

We use a quality function deployment method in deciding what to implement first. Simply put, we will implement those features first that have the greatest overall benefit with the smallest cost.

2.2 Enhanced Reporting

2.2.1 Introductory User Page

There shall be a special report page giving a user an overview. On this page she shall see

Section	Feature Synopsis	Cost	Priority	w.i.p.
2.5.1	extended regression test suite	medium	10	*
2.2	enhanced reporting (XML, PDF)	medium	8	
2.3	SOAP interface to create and edit artifacts	high	8	*
2.3.2	Eclipse plugin to create and edit artifacts	high	8	*
2.3.3	Integration with CVS	high	8	
2.4.6	workflow support	medium	8	*
2.5.2	HTML e-mails	high	8	
2.5.13	import from Bugzilla	medium	7	
2.5.8	prevent deletion of administrators	low	7	
2.4.3	easy installation	medium	7	*
2.3.1	e-mail interface	medium	6	
2.5.3	e-mail reminder	low/medium	6	
2.5.4	project schedule and cost management	medium/high	6	
2.5.5	individual notification	low	5	
2.4.5	registration verification	low	5	
2.4.4	hiding input fields	low	5	
2.5.6	copying of issues	low	5	
2.5.12	mass updates	medium	5	
2.5.11	bulk input of new project data	medium	4	
2.5.9	add resolution type field	low	3	
2.5.7	moving to Struts tiles	medium	3	
2.5.7	support DB2 and Sybase	medium	3	
2.5.10	icons for priority and severity	low	2	
2.4.1	copy project setup	low	1	
2.4.2	delete project	low	1	
	collected e-mails	medium	1	

Table 2.2: Overview of new Features (w.i.p.: work in progress)

- number of items responsible for and not closed, grouped by project
- number of items created by user and not closed grouped by project
- number of items closed during the last X days grouped by project
- number of items due the next X days grouped by project

The user shall be entitled to select on her personal profile page which report she wants to see directly after login, including none.

2.2.2 Report via XML

Reports appearance and structure should be user definable. For this purpose it would be nice to output reports as XML data and have it it formatted via XSL stylesheets. One suggestion was to use RSS as a default format. Another option might be to use JFreeReport.

2.2.3 Reports in PDF

There shall be an option to generate reports in PDF via \LaTeX and dvipdfm. One possible solution would be to provide an XSL to transform the general reports into \LaTeX . Using JFreeReport, this feature may coincide with the one described in section 2.2.2.

2.3 Interfacing

The following interfaces shall be added:

- E-mail interface
- WSDL/SOAP import interface to create and edit issues
- XML export interface (see reporting, 2.2.2)

2.3.1 E-Mail Interface

There shall be a separate program outside of the *Track+* application that periodically reads e-mails sent to the *Track+* e-mail account configured in the POP3 section in `web.xml` of the application.

This program accesses the *Track+* application via its HTML interface based on the `httpunit` API. The program shall support three functions:

- initiate the sending of an e-mail template and a “token” password that shall be valid for one day (requiring a new database field)
- add a new issue to an existing project
- add a comment to an existing issue

Which function to use is defined by the subject line. The subject line needs to contain one of three keywords:

- TOKEN would send a token

- NEW would check in a new issue
- COMMENT would add a comment to an existing issue

The sender is being authenticated via his user name and the token that he had requested previously. The **Track+** password facility checks during login not only for a valid password, it would also check if the password matches the token and then let the user in.

The rest of the software would be very similar to the regression test cases that test the “new issue” and “edit issue” functionality.

2.3.2 Eclipse Plugin

There shall be an Eclipse plugin which permits to use **Track+** from within the development environment. The Eclipse plugin shall communicate with **Track+** via a SOAP interface. It shall provide the following functionality:

- It shall be possible to create and modify artifacts
- It shall be possible to retrieve artifacts via a report
- For defining the reports, it shall be possible to use either TQL or a graphical report configurator.
- The plugin shall be able to store logon parameters so that automatic logon is possible.

2.3.3 CVS Integration

Track+ shall provide an integration with CVS and Subversion. Integration shall be such that the following tasks can be performed:

- With a special tag, it shall be possible to directly access from within a **Track+** artifact an entity inside a CVS repository via viewCVS or similar (active hyperlink), including version.
- It shall be possible to configure the path to the CVS repository on a per project basis and store it as a project property in the database.
- It shall be possible to generate a list with all changed CVS entities for a specific release from within **Track+**.
- There shall be a Java version of viewCVS to remove the need for installing another program suite (Python, viewCVS).

In the other direction, CVS clients shall support the following features:

- When committing a file, it shall be possible to define **Track+** issue numbers with a special tag as part of the comment. The issue numbers could be part of a list, offered with synopsis, or just a raw field.
- With the commit, the **Track+** artifacts with the issue numbers referenced in the comment shall automatically be updated with the comment in the comment section of an artifact. State changes shall not be performed.

- The tags in the comments shall be of the same format as the tags understood by *Track+* as CVS repository references.
- Communication between the CVS client and *Track+* shall be via a SOAP interface.
- Authentication parameters can be stored with the CVS client.

2.4 Enhanced Administration

2.4.1 Copying Project Setup

It should be possible on the project administration page to copy all configuration settings like subsystem, release, class and later on additional project specific settings from existing projects. This would make setting up new projects much easier and faster, especially since some lists can be the same or similar.

2.4.2 Deleting Projects

It should be possible to permanently remove entire projects with all related data. It should be possible to mark projects as inactive and have them removed from the standard project lists.

2.4.3 Easy Installation and Configuration

There shall be a Java/Swing application that permits to configure a *Track+* server. With this utility, it shall be possible to check the database configuration and write the database configuration file.

When configuring the database there should be a test function available to check for proper configuration.

For the Windows platform, there shall be a complete installer packet (based on NIS) with an HSQL database, Tomcat server and the option to download the right JDK.

2.4.4 Hiding Input fields

It should be possible to hide input fields (like release, version, class) from the graphical user interface. Hiding fields should be configurable for each project, and for each list type like action items, bug reports, and so on.

2.4.5 Verification of Registration

The registration e-mail that is being sent to the newly registered user shall include a link to the *Track+* server. If that link is not visited in a period of one week after it has been sent the user shall be completely removed from the database. This would provide additional security against spam registrations.

2.4.6 Workflow Engine

It would be nice to have a configurable workflow engine to control possible state transitions. Each project could define specific workflows for its different item classes. I suggest to use the workflow package OSWorkflow of OpenSymphony. The allowed state transitions should be dependant on roles.

2.5 Miscellaneous

2.5.1 Extended Regression Test Suite

The existing regression test suite shall be extended to cover more test cases and to cover recently added functionality. Whenever possible, for a bug detected and fixed there shall be a test case added.

2.5.2 HMTL E-Mail

It shall be possible to send all e-mails from the *Track+* system in either simple format as implemented up to now, or in a nice HTML format. The user may set which format she prefers in her personal profile.

2.5.3 E-Mail Reminder

There shall be a thread running inside of *Track+* that sends reminder e-mails out to concerned users of their due or overdue tasks. The reminder frequency shall have a project specific default, but each user can set it individually in his personal profile.

The reminder shall be configurable as such that one can be reminded of issues where one is the owner, the manager, or the responsible, or combinations thereof.

2.5.4 Project Schedule and Cost Management

There shall be support for project schedule management along the suggestions of IEEE Standard 1490-1998, which is an adoption of the Project Managers Institute Standard "A Guide to the Project Management Body of Knowledge". According to the standards suggestion, we would like to trace activity duration (estimates and actual) plus activity costs (as hours or days). We create a new table TCOST which for each issue includes

- a key linking to a company cost account (TACCOUNT)
- a key linking to a user
- a key linking to a TWORKITEM
- a field for number of hours
- a field for cost
- a key linking to a currency table (TCURRENCY)

There may be many such entries for one TWORKITEM. We will add five fields to TWORKITEM: totalActualHours and totalActualCost which has the sum of all related TCOST entries, and estimatedHours and estimatedCost which has the latest estimates in it, plus a key linking to TCURRENCY.

For the estimated effort we use table TBASELINE. We add three fields: estimatedHours, estimatedCost, and a key linking to TCURRENCY. The latest values of these will be automatically copied into TWORKITEM.

We add two fields to TWORKITEM: actualStartDate and actualEndDate. The actual duration is given by the difference of these two. They will be automatically set on state change from opened to the next one, and when changing to state closed.

We would thus trace the costs based on a simplified Earned Value method with "budgeted cost of work scheduled" (tracing, also called original estimate), "actual cost of work performed" (not tracing, related to "remaining work") and "actual cost of work performed at closure of an activity" (also called final actual effort).

For improving an estimation process the "budgeted cost or work scheduled" and the final "actual cost of work performed at closure of an activity" are most useful.

Related to this feature there need to be appropriate reports such as

- total cost for a certain company account
- total hours for a certain user grouped by company account

In TCURRENCY we keep for each project the permitted currencies and their conversion factors. The base currency gets the conversion factor 1.0.

2.5.5 Notification

It should be possible to register for being notified for each individual issue, not just a group of issues. For this purpose there should be an appropriate checkbox on the issue overview page (`printItem.jsp`). For this we create a new table TISSUENOTIFY, linking users to specific issues for notification.

2.5.6 Copying an Issue

It should be possible to copy an issue. All relevant fields should be duplicated, except for the originator, the create date and the state. Attachments shall not be copied.

2.5.7 Moving to Struts Tiles and Validator

Moving the design of the JSP pages to Struts Tiles and validating inputs with the Struts Validator classes should enhance mostly maintainability and ease customization of the user interface. It would have no real visible benefit to customers.

2.5.8 Disable Deletion of Administrator

It should be prevented that a project administrator can remove his own administrator privileges by accident.

2.5.9 Resolution Type Field

There shall be a field “resolution” added stating how an issue was resolved (e.g. “fixed”, “rejected”, etc.).

2.5.10 Icons for Priority and Severity

There shall be icons for priority and severity in the edit and printItem forms as well as the reports.

2.5.11 Bulk Input

When starting a project it should be possible to input a whole bulk of data without editing each item. This is particularly useful when setting up standard lists like quality gate checklists which are basically the same for many different projects. A possible solution could be to initiate projects from XML templates.

2.5.12 Mass Updates

From the report list, it should be possible to select any reported issue and have a common operation performed on all issues. The first operation that should be implemented is a state change.

2.5.13 Bugzilla Import

To ease the migration for the many Bugzilla users that want to migrate to **Track+** there shall be a standalone tool that will move an existing BugZilla database into an existing **Track+** database.

2.5.14 Add new Database Support

To cover all “big” database systems, Sybase and DB2 shall be officially supported.

2.6 Down the Road

2.6.1 Support for Crystal Reports Report Application Server

It would be nice to integrate the Crystal Reports Report Application Server (optionally) into **Track+**.

2.6.2 More Interfaces

These interfaces shall be added in future releases:

- Microsoft Project import and export interface

2.6.3 Custom Fields

An administration interface could be provided allowing to add custom fields. The major problem with this feature is the design of the different screens.

2.6.4 Multi-Company Support

We will add a new table named TCOMPANY. Departments will be assigned to companies. Projects will be assigned to companies. One project may be related to different companies, and one company may be related to different projects. The project administrators will then only see users from the companies that are related to their projects. Only the system administrator will be able to see all users in the system. The users e-mail address shall be used to determine to which company she belongs. She will then automatically only see those departments that belong to her company. The registration process shall be thus a two stage process:

1. First the user registers without being offered a department
2. When he has to verify his e-mail address he is being offered a department list where he has to select something.

2.6.5 Help-Desk Support

We will add customers and assets into your system to support common help desk requirements.

