TrackStudio Enterprise 3.1

Table of Contents

1 Installation Guide

1.1 What's New?

This topic describes some new features of TrackStudio.

Description

TrackStudio Enterprise is one of the most flexible Java-based issue tracking systems. It supports the widest range of commercial and open source application servers and DBMSs, and can be run on any computer platform.

The following new features are introduced in TrackStudio 3.1

- The user interface is now much simpler.
- A new demo database, tutorials, and docs are available.
- Major performance improvements were made, especially on large databases (with 30000-50000 tasks).
- TrackStudio Standalone is available for Linux.
- The ability to modify filter conditions dynamically (directly from the **Subtasks** page) was added.
- The ability to filter users by any field was added.
- New custom field types have been added:

Type	Description
Task	Use this field to link tasks (depends, duplicate, etc). You can choose several tasks here.
User	Use this field to link tasks and users (customer, manager, etc). You can choose several users here.
Multi Select	Choose multiple values in a select list.
URL	Allow the user to input a single URL with description.

- More flexible email notification templates were added.
- Templates may now be configured on a per-issue basis.
- Task and message descriptions now support HTML. A WYSIWYG editor was implemented.
- A manager can now logon as a subordinate user.

Several features have been removed in TrackStudio 3.1:

- URL now contains session ID and you can’t bookmark it or give a link to friend. Use the **Perm Link** hyperlink instead.
- The Task and User Headers have been removed. Use the **Task->View** or **User->View** tabs instead.
- Dropdown lists with all available projects have been removed, but you can still jump to project by name or alias.
- You should now use calculated custom fields to filter overtime or over-budget tasks.
- The icon near the task name now means task state instead of task time/budget.
- You cannot control custom field visibility in e-mail notifications using custom field properties. Modify your e-mail template instead.
- **Receive notification** workflow permission has been removed.
- **Last visited date** and **E-mail for SMS** user properties have been removed.
- There is no longer a **MessageScheduler**-related permissions. Use task permissions (like **editActualBudget**) instead.

1.2 TrackStudio Upgrading

1.2.1 Upgrading from Version 3.1.x

This topic describes how to update from TrackStudio 3.1.x to TrackStudio 3.1.y.

Description

To update TrackStudio WAR:

1. Download the latest TrackStudio WAR.
2. Re-deploy the TrackStudio WAR.

To update the TrackStudio SA:

1. Download the latest TrackStudio SA.
2. Stop the TrackStudio 3.1.x instance.
3. Un-install Windows services if used:

```
>hsqlService /uninstall
>jettyService /uninstall
```

4. Install the TrackStudio 3.1.y to a separate directory.
5. Configure TrackStudio database, e-mail notification, e-mail submission and LDAP properties.
6. Ensure that you set the correct **Upload directory** path.
7. Clean the **Index directory** content.
8. Start TrackStudio 3.1.y

Notes

- Minor version upgrades (3.1.x->3.1.y) do not require database upgrade.
- The SA and WAR versions use the same database scheme and differ only in the distributed components – no special actions are required to transfer the data between the WAR and SA versions.

1.2.2 Upgrading from Version 3.0.x

This topic describes how to update from TrackStudio 3.0.x. to 3.1.y

Description

To upgrade the system from TrackStudio 3.0.x to TrackStudio 3.1.y:

1. Backup your database. You can do this through the DBMS, for example, by using the **`imp/exp`** utility in ORACLE. The backup will allow you to use version 3.0.x until possible problems with TrackStudio 3.1.y are solved.

2. Stop the TrackStudio 3.0.x instance.

3. Un-install Windows services if used:

```
>Wrapper.exe -r etc/hwrapper.conf
>Wrapper.exe -r etc/jwrapper.conf
```

4. Install the TrackStudio 3.1.y to a separate directory.

5. Upgrade database:

- **TrackStudio SA**: Run Server Manager from TrackStudio 3.1.y, go to the **Database Management** tab, and press the **Upgrade Database** button.
- **TrackStudio WAR**: Execute the update script for your DBMS.

6. Configure TrackStudio e-mail notification, e-mail submission and LDAP properties.

7. Ensure that you set the correct **Upload directory** path.

8. Clean the **Index directory** content.

9. Start TrackStudio 3.1.y

10. Login as **root.**

11. Update e-mail templates:

Property	Value
Subject	<${task.getProjectAlias()}>: ${addval.taskViewFactory.inEmailText(task).getName()} ${addval.taskViewFactory.inEmailText(task).getNumber()}
Content Type	HTML
From	Submitter
Reply To	TrackStudio
E-Mail Body	Copy/paste from **templates/common_html.ftl**

Property	Value
Subject	<${task.getProjectAlias()}>: ${addval.taskViewFactory.inEmailText(task).getName()} ${addval.taskViewFactory.inEmailText(task).getNumber()}
Content Type	text
From	Submitter
Reply To	TrackStudio
E-Maill Body	Copy/paste from **templates/common_text.ftl**

Possible problems after upgrading the database.

- During the first launch TrackStudio re-indexes the database and converts task/messages descriptions to HTML. It can take several minutes for each 1000 tasks and messages.
- XML Import/Export allows you to transfer the data of TrackStudio 3.1 between various DB, but it cannot be used to transfer data between various versions of TrackStudio.
- If errors or problems occur while upgrading the DB, you should contact us and continue using TrackStudio 3.0 until the problem is resolved.

1.2.3 Upgrading from Version 2.8.x

This topic describes how to update from TrackStudio 2.8.x.

Description

To upgrade the system from TrackStudio 2.8.x to TrackStudio 3.0.x:

1. Download the TrackStudio 3.0 using the following link: http://www.trackstudio.com/products-edownload.html

2. Upgrade TrackStudio database from 2.8.x to 3.0.x using **sman** or database upgrade script.

3. Run TrackStudio to finish the database conversion.

4. Ensure that everything works.

5. Upgrade from TrackStudio 3.0.x to 3.1.x.

1.3 Windows Installation

1.3.1 Installing TrackStudio SA for Windows

This section describes TrackStudio Enterprise configuration (Standalone distribution).

Description

To install and configure TrackStudio:

1. Run HSQL Database (**hsql.exe**).
2. Run Server Manager (**sman.exe**).
3. Press the **Start** button to run the server.
4. When the server is running, click the **Login** button.
5. Use the following to log on: **login=root, password=root.**

1.3.2 Installing TrackStudio WAR for Windows

This section describes TrackStudio Enterprise configuration (WAR distribution).

Description

To install and configure TrackStudio Enterprise you should perform the following steps:

1. Run your DBMS.

2. Create a new database using the corresponding SQL script which you can find in the **sql/en** directory. If an error occurs during the creation of a new database, TrackStudio may work incorrectly or fail to work completely. Contact us, if such an error occurs.

3. Define the TrackStudio configuration in the files **trackstudio.properties**, **trackstudio.mail.properties** and **trackstudio.hibernate.properties**.

4. Use the **TS_CONFIG** environment variable to specify the directory name with configuration files (*.properties).

5. Perform the deployment of **TrackStudio.war**. You can also unpack WAR and perform the deployment of the directory structure.

6. Run the application server.

7. The application is available at **http://localhost:port/TrackStudio**

8. Use the following to log on: **login=root** and **password=root**

JBoss 4.0.1 installation notes:

- After you install TrackStudio, update **jboss\jboss-hibernate.deployer\cglib-full-2.0.1.jar** to **cglib-full-2.0.2.jar** and **jboss\jboss-hibernate.deployer\hibernate2.jar** to **hibernate-2.1.8.jar**, JBoss has its own outdated version.

ServletExec installation notes:

- After you install TrackStudio, delete **xercesImpl.jar** reference from the ServletExec startup script (**StartServletExec.bat**).

Remarks

You cannot install several TrackStudio instances (test and production, for example) on the same application server instance.

1.3.3 Running as Windows Service

This topic describes how to run TrackStudio as Windows service.

Description

HSQL Database

- To install HSQL Database as a Windows service:

```
>hsqlService /install
Installed service 'hsqlService'.
```

- To run HSQL Database as a Windows service:

```
>hsqlService /start
Starting service 'hsqlService'.
```

- To stop a Windows service:

```
>hsqlService /stop
Stopping service 'hsqlService'.
Service stopped
```

- To uninstall a Windows service:

```
>hsqlService /uninstall
Uninstalled service 'hsqlService'.
```

Jetty

- To install Jetty as a Windows service:

```
>jettyService /install
Installed service 'jettyService'.
```

- To run TrackStudio as a Windows service:

```
>jettyService /start
Starting service 'jettyService'.
```

- To stop a Windows service:

```
>jettyService /stop
Stopping service 'jettyService'.
Service stopped
```

- To uninstall a Windows service:

```
>jettyService /uninstall
Uninstalled service 'jettyService'.
```

1.3.4 Integrating IIS and TrackStudio

In this section we describe how to integrate TrackStudio with Internet Information Server.

Description

TrackStudio can be configured so that when you reference the IIS virtual folder (e.g. */TrackStudio*), you will be redirected to TrackStudio. Such configuration can be useful in case you want to permit access to TrackStudio via the existing Internet Information Server.

To integrate TrackStudio with Internet Information Server:

1. Start the ISS administration software (**Start -> Programs -> Administrative Files -> Internet Information Services**).
2. Create a virtual TrackStudio folder for one of the web sites, e.g. for the *Default Web Site* (**Action -> New -> Virtual Directory**).
   - Specify the **<TRACKSTUDIO_HOME>** path as a local path to the virtual folder.
   - Allow the **Execute** permission.
3. Add a filter to the selected web site (**Default Web Site -> Properties -> ISAPI Filter -> Add...**).
   - Select the **<TRACKSTUDIO_HOME>\lib\isapi_redirector2.dll** file as executable.
4. **IIS6 only:** Allow the Web Service Extension to operate:
   - Click on the **Web Services Extensions** item in the left hand pane.
   - In the right hand pane, add a new Web Service Extension.
   - Browse and set the required file for this extension to the **<TRACKSTUDIO_HOME>\lib\isapi_redirector2.dll**
   - Set the status to **allowed.**
5. Define the **trackstudio.siteURL** (in the **trackstudio.properties** file) as **http://<IIS server>/TrackStudio**
6. Execute **install4iis.js**
7. Restart Internet Information Server.
8. Restart TrackStudio Enterprise.
9. Now TrackStudio will be available as **http://<IIS server>/TrackStudio**

Remarks

It is very important to start the IIS and TrackStudio in the proper order -- first start IIS and then start TrackStudio. When starting TrackStudio you will see a warning message, informing you that the specified port is not available. Ignore it.

1.3.5 Configuring MS Active Directory Authentication

This topic describes how to configure the user authentication via the **MS Active Directory Service**.

Description

To configure the user authentication via the **MS Active Directory Service:**

1. Login into **MS Windows** as *Administrator*

2. Export LDAP context to the file.

```
ldifde –f ldap.txt
```

3. Open the result *ldap.txt* file. The first line of the file should be

```
dn: DC=ldap-server,DC=my-company,DC=com
```

4. Enable LDAP in **trackstudio.ldap.properties**

```
trackstudio.useLDAP yes
```

5. Set base DN to **cn=users** in specified DN

```
ldap.baseDN = cn=users,dc=ldap-server,dc=my-company,dc=com
```

6. Set the user which will be used to login to the server.

```
ldap.userDN = cn=Administrator,cn=users,dc=ldap-server,dc=my-company,dc=com
```

7. To login by **Name** set

```
ldap.loginAttrLDAP=displayName
ldap.loginAttrTS name
```

To login by **Login** set

```
ldap.loginAttrTS login
ldap.loginAttrLDAP=sAMAccountName
```

8. Set the password.

9. Click the **Test Connection** button to test the LDAP connection.

How it works:

If **trackstudio.useLDAP** is set to **yes**, TrackStudio will connect to the specified LDAP server during login and performs authentication using the login and password specified in **ldap.userDN** and **ldap.userDNpass**. TrackStudio then performs DB query and finds the user in the local DB by specified login and password. After that TrackStudio searches in the LDAP server for the user, the **ldap.loginAttrLDAP** parameter which is equal to the **name** or the **login** (depending on **ldap.loginAttrTS** value) of the found user. Then the authentication of the found user is performed using the password specified in the login window.

Notes

- You should always use your TrackStudio **login** in the **Login** window.
- Even if you use LDAP authorization, you will have to register a new user in TrackStudio first.
- When you change the password under the **Change Password** tab, the password will chang in the database, but not the LDAP.
- A user can log in if his/her password matches the one stored in the DB or the one specified in LDAP. To avoid authorization via the local database, you should remove **gran.app.adapter.auth.SimpleAuthAdapter** from the pipeline in the **trackstudio.adapter.properties** file.

See Also

- Using LDIFDE to Import and Export Directory Objects to Active Directory

1.4 UNIX Installation

1.4.1 Installing TrackStudio SA for UNIX

This section describes TrackStudio Enterprise configuration (Standalone distribution).

Description

To install and configure TrackStudio:

1. Run HSQLDB (**hsql**).
2. Run TrackStudio Enterprise Server Manager (**sman**).
3. Press the **Start** button to run the server.
4. When the server is running, click the **Login** button.
5. Use the following to log on: **login=root, password=root.**

1.4.2 Installing TrackStudio WAR for UNIX

This section describes TrackStudio Enterprise configuration (WAR distribution).

Description

To install and configure TrackStudio Enterprise:

1. Run your DBMS.
2. Create a new database using the corresponding SQL script which you can find in the **sql/en** directory. If an error occurs during the creation of a new database, TrackStudio may work incorrectly or fail to work completely. Contact us, if such an error occurs.
3. Define the TrackStudio configuration in the files **trackstudio.properties**, **trackstudio.mail.properties** and **trackstudio.hibernate.properties**.
4. Use the **TS_CONFIG** environment variable to specify the directory name with configuration files (*.properties).
5. Perform the deployment of **TrackStudio.war**. You can also unpack WAR and perform the deployment of the directory structure.
6. Run the application server.
7. The application is available at **http://localhost:port/TrackStudio**
8. Use the following to log on: **login=root** and **password=root**

JBoss 4.0.1 installation notes:

- After you install TrackStudio, update **jboss\jboss-hibernate.deployer\cglib-full-2.0.1.jar** to **cglib-full-2.0.2.jar** and **jboss\jboss-hibernate.deployer\hibernate2.jar** to **hibernate-2.1.8.jar**, JBoss has its own outdated version.

ServletExec installation notes:

- After you install TrackStudio, delete **xercesImpl.jar** reference from the ServletExec startup script (**StartServletExec.sh**).

Remarks

You cannot install several TrackStudio instances (test and production, for example) on the same application server instance.

Notes

TrackStudio does not contain graphical libraries for generating colors and fonts and other AWT information. For that, Java relies on the system's libraries for providing that information. Thus an environment capable of providing AWT information and a graphics card (for exporting to static formats) are required.

In a Windows environment, nothing extra needs to be done to set up such an environment as it already exists. A GUI interface is already running and a graphics card already exists.

For non-Windows environments, such is usually not the case. You need to have X or some form of X running on such systems and point the display to the machine running X (such as running the command export *DISPLAY=192.168.0.16:0.0* in a korn shell). For best performance, TrackStudio recommends running X on the machine (or setting the DISPLAY to point to another machine running X). However, if that is not an acceptable solution, there are alternative solutions available.

If your TrackStudio UNIX server does not have an X11 Server installed or the DISPLAY environment variable is not set, you may receive one of the following errors when executing your reports:

```
Can't connect to X11 window server using ':0.0'
  as the value of the DISPLAY variable., stack:
java.lang.InternalError: Can't connect to X11 window server using
  ':0.0' as the value of the DISPLAY variable.
at sun.awt.X11GraphicsEnvironment.initDisplay(Native Method)
```

or

```
Internal error: exception thrown from the servlet service
function (uri=/xxx/xxx2.jsp):
java.lang.NoClassDefFoundError: java/awt/SystemColor, stack:
java.lang.NoClassDefFoundError: java/awt/SystemColor
   at com.sas.visuals.BaseBorder.<init>(BaseBorder.java:209)
```

Possible circumventions follow:

1) Install the X11 Server and set the DISPLAY environment variable.

2) Pass the parameter **-Djava.awt.headless=true** to java when you run it. This no longer requires Xvfb to be running, but it

does require the X11 packages to be installed.

Where to specify the options will vary between servlet engines. For Tomcat 4.x, you would specify these options in the **catalina.bat** or **catalina.sh** file for **CATALINA_OPTS**.

1.4.3 Installing an SSL Certificate

The following part deals with the installation of an SSL certificate for jetty.

Description

1) Create a keystore. keytool supplied with Sun JDK.

```
> jdk/bin/keytool -genkey -alias my-cert -keyalg RSA
        -keystore .mykeystore
```

When creating a certificate, you must specify *keystorePassword* and *keyPassword*

2) Create a certificate request, CSR (into the file **cert.csr**)

```
> jdk/bin/keytool -certreq -alias my-cert -file cert.csr
        -keystore .mykeystore
```

3) Send CSR to Verisign (or some other company), in response you must get a **cert.crt**.

The following URL can be used for testing

https://www.thawte.com/cgi/server/test.exe

4) Convert cert.crt from PEM to DER (**cert.der**). You can use **openssl** to convert it:

```
openssl x509 -in cert.crt -out cert.der –outform DER
```

5) Import the certificate into the keystore:

```
> jdk/bin/keytool -import -alias my-cert -file cert.der -keystore .mykeystore
```

6) Edit **jetty.xml**:

```
<Call name="addListener">
    <Arg>
      <New class="org.mortbay.http.SunJsseListener">
        <Set name="Port">8443</Set>
        <Set name="MinThreads">5</Set>
        <Set name="MaxThreads">100</Set>
        <Set name="MaxIdleTimeMs">30000</Set>
        <Set name="LowResourcePersistTimeMs">2000</Set>
        <Set name="Keystore"><SystemProperty name="jetty.home"
             default="."/>/.mykeystore</Set>
        <Set name="Password">keystorePassword</Set>
        <Set name="KeyPassword">keyPassword</Set>
      </New>
    </Arg>
  </Call>
```

7) Change the protocol and port for **siteURL** in **trackstudio.properties**.

```
# URL of your site. Host name and port should be correct.
# We use this address in e-mail notification messages.

trackstudio.siteURL https://localhost:8443/TrackStudio
```

8) Launch jetty.

9) Open **https://localhost:8443/TrackStudio**

To create a self-signed certificate, do the following:

1) Create Certificate Authority. To do that, run

```
perl ./CA.pl -newca
```

or

```
./CA -newca
```

2) Create a certificate request:

```
jdk/bin/keytool -certreq -alias my-cert -file cert.csr
        -keystore .mykeystore
```

3) Create a certificate

```
openssl ca -config /usr/share/ssl/openssl.cnf
           -out cert.crt -infiles cert.csr
```

4) Verify the certificate:

```
openssl verify -CAfile ./demoCA/cacert.pem cert.crt
```

5) Convert the certificate from PEM to DER:

```
openssl x509 -in cert.crt -out cert.der -outform DER
```

6) Import **cert.der** into the keystore.

Notes

Please note that some functionality (Excel reports, Save target as... when file download, etc) will not work with demo cert under MS Internet Explorer. Use cert from certification authority (like Verisign) to solve this issue.

1.5 Creating and Configuring Database

1.5.1 Initializing DB2 Database

This section describes how to configure DB2 database.

Description

1. Create user tablespace and temp system tablespace.

2. Open a DB2 command window (DOS prompt) (Windows) or log into the server (UNIX)

3. Connect to the database

```
> db2 connect to <databasename> user <dbuser> using <password>
```

4.Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.DB2Dialect
hibernate.connection.url jdbc:db2://127.0.0.1/trackstudio
hibernate.connection.driver_class COM.ibm.db2.jdbc.net.DB2Driver
hibernate.connection.username db2admin
hibernate.connection.password db2admin
```

5. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.
- **TrackStudio WAR**: Execute **sql\en\trackstudio-db2.sql:**

```
> db2 -tvf trackstudio-db2.sql
```

1.5.2 Initializing HSQLDB Database

This section describes how to configure HSQLDB database.

Description

1. Start HSQL Database:

```
> java -cp hsqldb.jar org.hsqldb.Server -database TrackStudio
```

2. Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.HSQLDialect
hibernate.connection.url jdbc:hsqldb:hsql://localhost
hibernate.connection.driver_class org.hsqldb.jdbcDriver
hibernate.connection.username sa
hibernate.connection.password
```

3. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.
- **TrackStudio WAR**: Execute **sql\en\trackstudio-hsql.sql:**

```
> java -cp hsqldb.jar org.hsqldb.util.DatabaseManager
```

1.5.3 Initializing PostgreSQL Database

This section describes how to configure PostgreSQL database.

Description

1. Start postmaster.

```
> postmaster -D ../data/ -i -h host.mycompany.com
```

2. Create an empty database:

```
> createdb -E UNICODE -U postgres trackstudio
```

3. Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.PostgreSQLDialect
hibernate.connection.url jdbc:postgresql://127.0.0.1:5432/trackstudio
hibernate.connection.driver_class org.postgresql.Driver
hibernate.connection.username postgres
hibernate.connection.password postgres
```

4. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.
- **TrackStudio WAR**: Execute **sql\en\trackstudio-pgsql.sql:**

```
> psql --user=postgres -d trackstudio -f trackstudio-pgsql.sql
```

Notes

- To backup database execute:

```
> pg_dump -U postgres -Fc -Z9 trackstudio > trackstudio.dmp
```

- To restore database execute:

```
> createdb -E UNICODE -U postgres trackstudio
> pg_restore -U postgres --disable-triggers -S postgres -d trackstudio trackstudio.dmp
```

1.5.4 Initializing ORACLE Database

This section describes how to configure ORACLE database.

Description

1. Create Tablespace.

2. Create TrackStudio user

3. Grant **DBA** and **Resource** role to created user.

4. Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.OracleDialect
hibernate.connection.url jdbc:oracle:thin:@localhost:1521:ORCL
hibernate.connection.driver_class oracle.jdbc.driver.OracleDriver
hibernate.connection.username trackstudio
hibernate.connection.password trackstudio
```

5. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.
- **TrackStudio WAR**:

```
>sqlplus

SQL*Plus: Release 10.1.0.2.0 - Production

Copyright (c) 1982, 2004, Oracle.  All rights reserved.

Enter user-name: trackstudio
Enter password:

Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> set define off;
SQL> @@ trackstudio-oracle.sql
```

Notes

Oracle connection string includes **database URL**, **JDBC driver**, **Login** and **Password**. First part (before "@") of this **URL** is common, you have no need to modify it. After this character you need to enter your database location like **HostAddress:Port:ORACLE_SID**. If you are using locally installed version of Oracle, **HostAddress** is *localhost*. Default Oracle **port** is *1521*, default **ORACLE_SID** is *ORCL*. In the **JDBC driver** field there is a default JDBC Driver for Oracle, that you don't usually need to modify.

1.5.5 Initializing MS SQL Server Database

This section describes how to configure Microsoft SQL Server database.

Description

1. Start **Enterprise Manager**

2. Create database.

3. Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.GenericDialect
hibernate.connection.url jdbc:jtds:sqlserver://127.0.0.1:1433/trackstudio
hibernate.connection.driver_class net.sourceforge.jtds.jdbc.Driver
hibernate.connection.username sa
hibernate.connection.password
```

4. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.
- **TrackStudio WAR**: Start **Query Analyzer** and execute **sql\en\trackstudio-mssql.sql.**

Notes

To store unicode characters TrackStudio supports only UTF-8 encoding, but MS SQL supports only UCS-2 encoding. This means that you can't use UTF-8 character encoding with MS SQL. If you need to store national characters in MS SQL - please consider to use some national character encoding (TIS-620 for Thai characters, for example). Change

- varchar to nvarchar
- text to ntext
- char to nchar

data-types for storage in trackstudio-mssql.sql or in mssql.h that packed into sman.jar or just contact TrackStudio Support for more information.

1.5.6 Initializing Firebird Database

This section describes how to configure Firebird/Interbase database.

Description

1. Start Firebird's **isql** program.

2. Create database

```
SQL> create database 'c:\trackstudio.gdb' user 'sysdba' password 'masterkey';
```

3. Connect to the database

```
SQL> connect 'c:\trackstudio.gdb' user 'sysdba' password 'masterkey';
Commit current transaction (y/n)?y
Committing.
Database: 'c:\trackstudio.gdb', User: sysdba
```

4. **TrackStuido SA:** exit from **isql**:

```
SQL> exit;
```

5. Configure the database connection properties:

- **TrackStudio SA:** Go to the **Database Connectivity** tab, enter the JDBC connection properties.
- **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect net.sf.hibernate.dialect.InterbaseDialect
hibernate.connection.url jdbc:firebirdsql://localhost/c:/trackstudio.gdb
hibernate.connection.driver_class org.firebirdsql.jdbc.FBDriver
hibernate.connection.username sysdba
hibernate.connection.password masterkey
```

6. Initialize the database:

- **TrackStudio SA:** Go to the **Database Management** tab and click the **Create Database** button.

- **TrackStudio WAR**: Execute **sql\en\trackstudio-firebird.sql:**

```
SQL> in trackstudio-firebird.sql;
```

Notes

Before upgrade the database, execute the following:

```
gfix.exe -user sysdba -password masterkey trackstudio.gdb -sql_dialect 3
```

1.5.7 Creating Database for Performance Testing

Use **massive** for creating databases for TrackStudio benchmarks.

Description

Use **massive** to generate a large databases with the necessary configuration and structure. You can use this database to check the TrackStudio performance rate for that database and your hardware. Massive supplied with TrackStudio SA only.

To prepare a database

1. Edit **massive/default.properties** to configure test database
2. Execute

```
> massive/massive
```

To create test database:

1. Run TrackStudio Server Manager.
2. Select the **Database Connectivity** tab
3. Enter the JDBC connection properties.
4. Select the **Database Management** tab
5. Enter the generated XML file name
6. Click the **Create Database** button.

To login as an administrator, use **login=root, password=root**. Other users have logins of the following type: **user2, user3**, etc and password **root**.

Notes

When you run TrackStudio for the first time, it indexes all tasks for full text search. The process of indexing can take several hours. If you are not going to use full text search, you can skip the indexing process. To do this before running TackStudio, create the file **skipindex.flag** in the directory that is specified in the **trackstudio.indexDir** parameter in **trackstuido.properties**.

1.6 Configuring TrackStudio Cluster

This topic described how to configure TrackStudio to be launched on several application servers grouped into a cluster.

Description

For the sake of better scalability and stability TrackStudio can be launched on several application servers grouped into a cluster. Load balancing makes it possible to distribute the load between servers in the cluster. A failover cluster is a set of servers that are configured so that if one server becomes unavailable, another server automatically takes over for the failed server and continues processing.

TrackStudio uses cache for data processing. The cache stores the information about tasks and users that have been accessed and contains the results of database queries. When TrackStudio works within an application server cluster, it is important to synchronize the caches between cluster nodes. Once any object in the cache is changed, TrackStudio sends out notifications to TrackStudio instances running on other cluster nodes. Those notifications are used to clear the changed objects from their caches.

To configure a TrackStudio cluster:

1. Install TrackStudio on all cluster nodes. As TrackStudio uses broadcast messages to send notifications, the cluster nodes must be within one physical network. All instances must use the same version of TrackStudio Enterprise.

2. Edit the **trackstudio.properties** files on both nodes. Set **trackstudio.cluster** to **yes** and specify the same **trackstudio.cluster.name** for all cluster nodes.

3. Specify a directory for storing uploaded files: **trackstudio.uploadDir**. Both instances must use the same directory in which to store uploads. You can use shared disk in Windows or NFS in UNIX.

4. Specify a directory in which to store the full text search index: **trackstudio.indexDir**. Each instance must have a local copy of **indexDir**.

5. Configure database connections. Both instances must use the same database.

6. Configure the rest of the settings and launch TrackStudio on all cluster nodes. While loading, TrackStudio displays messages about active nodes in the cluster. For instance, if there are 2 instances on the cluster (both are running on one server), the display will be as follows:

```
-------------------------------------------------------
GMS: address is TMK-12X3:4390
-------------------------------------------------------
[18:38:19.93] ... finished starting new JavaGroups Communicator.
[18:38:19.98] A host has joined the cache notification bus: TMK-12X3:4383
[18:38:19.98] A host has joined the cache notification bus: TMK-12X3:4390
```

1.7 Importing and Exporting Database

This topic describes how to transfer the data stored in the TrackStudio database to another DBMS.

Description

To export the data into XML:

1. Stop TrackStudio, if it is running.
2. Start Server Manage (available in TrackStudio SA only).
3. Select the **Database Connectivity** tab.
4. Enter the JDBC connection properties.
5. Select the **General** tab.
6. Specify the character encoding of the database.
7. Select the **Database Management** tab.
8. Specify the name of the file to which the data will be exported.
9. Click the **Export Database** button.

You should now have an XML file containing all the information from the tables in your TrackStudio database. You can edit and view these using any text editor.

To import the data into a database:

1. Stop TrackStudio, if it is running.
2. Start Server Manager (available in TrackStudio SA only).

3. Select the **Database Connectivity** tab.
4. Enter the JDBC connection properties.
5. Select the **General** tab.
6. Specify the character encoding of the database.
7. Select the **Database Management** tab.
8. Specify the name of the imported XML file in the **Choose XML scheme** field.
9. Click the **Create Database** button.

Remarks

The import/export feature can only be used to transfer data between two similar versions of TrackStudio. In the event that you have a database from an old version of TrackStudio that you want to transfer to another DBMS you should first upgrade the database to the latest version.

1.8 Importing from Third-Party Systems

This topic describes the process of importing data to TrackStudio from other issue tracking systems.

Description

To import data from other systems:

Export to XML the original database to which the data will be imported afterwards:

1. Launch **sman** (available in TrackStudio SA only).
2. Switch to the **Database Connectivity** tab.
3. Fill-in the database connection properties.
4. Switch to the **Database Management** tab.
5. Click the **Create Database** button to initialize a database.
6. Enter a name for the export file.
7. Click the **Export Database** button.

Export the data from the old issue tracking system into TrackStudio XML file:

Open the created TrackStudio XML export file using any XML editor or text editor. The XML file looks as follows:

```
<tsExportData>
  <table name="TABLENAME">
    <row>
      <data name="COLUMNNAME1"><![CDATA[DATA1]]></data>
      <data name="COLUMNNAME2"><![CDATA[DATA2]]></data>
      ...
    </row>
    ...
  </table>
  ...
</tsExportData>
```

Where

- **TABLENAME** -- the name of a table in the database
- **COLUMNNAMEi** -- the name of a column in the current database
- **DATAi** -- the values of table fields for the current record

To import a task, create a record in the **GR_TASK** table. To import a user, create a record it the **GR_USER** table. To find out what tables should be modified to create an object:

1. Export the database.

2. Log into TrackStudio and create the needed object using the web interface (workflow, message, etc).

3. Export the database once again.

4. Compare the file created at step 3 with the file created at step 1.

Each record has a unique primary key, for example **task_id**, **user_id**. TrackStudio uses the GUID generator to set the value of the primary key. When importing data, you can use any method to define the primary key-- just take care that the record keys are unique for each table. The maximum key length is 32 bytes.

Initialize a database using the modified XML file:

1. Create a new empty database.
2. Launch **sman**
3. Switch to the **Database Connectivity** tab.
4. Fill-in the database connection properties.
5. Switch to the **Database Management** tab.
6. Specify the name of the XML file.
7. Click the **Create Database** button.

Notes

To simplify installation we don't send license with embedded evaluation period by e-mail. Instead, TrackStudio estimates the evaluation period based on submit date of your tasks. This feature has a side effect: if you import tasks submitted more than 90 days ago you will got the License Exception during Login process. To workaround this issue set the **task_submitdate** to the current date for all your imported tasks. After purchase you can re-import those tasks with actual submit dates.

1.9 Integrating IDE and TrackStudio

This topic describes how to install and use TrackStudio IDE plug-ins.

Description

Use plug-ins to manage issues within IDEA, Eclipse or JBuilder.

1.9.1 Installing IDEA Plug-in

This topic describes how to install the IntelliJ IDEA plug-in.

Description

To install a plug-in:

1. Install TrackStudio DevPack

2. Enable TrackStudio SOAP API in **trackstudio.properties**

3. Create the directory **[IDEA_INSTALLATION_PATH]/plugins** and unpack the archive file **trackstudio.com.idea.zip** to it.

4. Launch IDEA.

5. Open the **Settings** window (**File->Settings**).

6. In the TrackStudio settings specify the URL of the TrackStudio **server**, **login** and **password**.

To open the TrackStudio window:

1. Create a new project or open an existing one.
2. Open the **TrackStudio** tab at the bottom of the main window (near **TODO**)

1.9.2 Installing JBuilder Plug-in

This topic describes how to install the JBuilder plug-in.

Description

To install a plug-in:

1. Install TrackStudio DevPack
2. Enable TrackStudio SOAP API in **trackstudio.properties**
3. Place the file **ts-jbuilder.jar** into the directory **[JBUILDER_INSTALLATION_PATH]/lib/ext**.
4. Launch JBuilder
5. Select the **Settings** item in the **TrackStudio** menu.
6. In the opened window, specify the **URL** of the TrackStudio server, **login** and **password**.

To open TrackStudio window:

1. Create a new project or an open existing one.
2. Select **File->New File**
3. Choose **trackstudio** file type.
4. Click the **OK** button.
5. Choose the **TrackStudio** tab for the created file at the bottom of the window.

1.9.3 Installing Eclipse Plug-in

This topic describes how to install the Eclipse plug-in.

Description

To install a plug-in (Microsoft Windows only):

1. Install TrackStudio DevPack
2. Enable TrackStudio SOAP API in **trackstudio.properties**
3. Create the directory **[ECLIPSE_INSTALLATION_PATH]/plugins/com.trackstudio** and unpack the archive file **trackstudio.com.eclipse.zip** to it.
4. Launch Eclipse.
5. Open the **Window->Preferences** window.
6. In the **TrackStudio** settings, specify the **URL** of the TrackStudio server, **login** and **password**.

To open TrackStudio window:

1. Create a new project or open an existing one.
2. Select **File->New->Other.**
3. Check the **Show All Wizards** checkbox.
4. Choose **TrackStudio->New TrackStudio session** item.

1.10 Integrating SCM and TrackStudio

This topic describes how to integrate TrackStudio with SCM tools.

1.10.1 CVS Integration

This topic describes how to integrate TrackStudio with CVS.

Description

Description

TrackStudio can be integrated with CVS version control system through our SOAP API. CVS check-in messages which are automatically appended to tasks. They can also be bounced by e-mail to the development team. This ensures all changes are logged.

To implement CVS integration:

1. Install TrackStudio DevPack

2. Enable TrackStudio SOAP API in **trackstudio.properties**

3. Receive the administrative files.

```
$ cvs checkout CVSROOT
```

4. Add the following string to the **CVSROOT/loginfo**

```
DEFAULT c:/devpack/link --url http://localhost:8888/TrackStudio --login cvsLogin --password
cvsPassword
```

• *DEFAULT* -- is a regular expression which is tested against the directory relative to the CVSROOT in which the change is being made. If the match is found, the remainder of the line is a filter program that expects log information on its standard input. If the repository name does not match any of the regular expressions in this file, the specified **DEFAULT** line is used. All occurrences of the **ALL** name appearing as a regular expression are used in addition to the first matching regular expression or **DEFAULT**.

5. Commit the file.

```
$ cvs commit -m "" CVSROOT/loginfo
```

To import CVS message into TrackStudio commit your files. Message will be added to the tasks specified in the message body.

```
cvs -z9 commit -m "This message should be added to the task #1 and #2."
2.8_bugs.txt (in directory C:\42\)
Checking in 2.8_bugs.txt;
C:/43/2.8_bugs.txt,v <-- 2.8_bugs.txt
new revision: 1.16; previous revision: 1.15
done
Adding message to the task #1... done
Adding message to the task #2... done

*****CVS exited normally with code 0*****
```

1.10.2 Subversion Integration

This topic describes how to integrate TrackStudio with Subversion.

Description

TrackStudio can be integrated with Subversion version control system through our SOAP API. Subversion check-in messages which are automatically appended to tasks. They can also be bounced by e-mail to the development team. This ensures all changes are logged.

To implement SVN integration:

1. Install TrackStudio DevPack

2. Enable TrackStudio SOAP API in **trackstudio.properties**

3. Modify the **post-commit** hook. A hook is a program triggered by some repository event, such as the creation of a new revision. Note that **post-commit** must be executable by the user(s) who will invoke it (typically the user httpd runs as), and that user must have filesystem-level permission to access the repository.

- **UNIX:** edit the **post-commit**

```
#!/bin/sh
REPOS="$1"
REV="$2"
SVNLOOK=/usr/local/subversion/bin/svnlook
$SVNLOOK log "$REPOS" | sh /devpack/link \
--url http://localhost:8888/TrackStudio --login svnLogin --password svnPassword
```

- **Windows:** edit the **post-commit.bat**

```
@echo off
svnlook log %1 | c:/devpack/link \
--url http://localhost:8888/TrackStudio --login svnLogin --password svnPassword
```

To import SVN message into TrackStudio commit your files. Message will be added to the tasks specified in the message body.

```
$ svn commit -m "This message should be added to the task #1 and #2."
```

1.11 Integrating E-Mail Client and TrackStudio

1.11.1 Configuring E-Mail Notification

This topic describes how to enable e-mail notification.

Description

To enable e-mail notification:

1. Enable e-mail notification in **trackstudio.mail.properties**. You can also use the Server Manager.
2. Specify the e-mail addresses of the users you want to receive the notifications.
3. Create the email notification rule using the **Filter->Notifications** tab. The filter determines which tasks send e-mail notifications, while the task determines for which project the e-mail notification is enabled.

1.11.2 Configuring E-Mail Submission

This topic describes how to enable e-mail submission.

Description

To enable e-mail submission:

1. Enable e-mail submission in **trackstudio.mail.properties**. You can also use the Server Manager.
2. Create accounts for the users who you want to use this submission.
3. **Optional:** Create an e-mail submission rule.

Notes

To use the HTML form submission by e-mail enable JavaScript in your e-mail client. Please note that many of the popular Web-based e-mail systems such as Yahoo.com and Mail.com/Email.com intentionally disable JavaScript in messages, and there is no way to re-enable it, so these systems will not work with TrackStudio. Following are instructions for enabling JavaScript in some popular e-mail readers. (Different versions of these readers may be slightly different in the details, but are probably similar.) For more information about these readers, please check the documentation or visit the vendor sites.

Mozilla Messenger

1.From the menu bar, choose **Edit**, then **Preferences**

2.Select **Advanced** from the list of options, then **Scripts & Plugins**

3.Click the box to **Enable JavaScipt for Mail and News**

Outlook Express

1.From the menu bar, choose **Tools**, then **Options**, then **Security**

2.Under **Virus Protection**, then under **Select the Internet Explorer security zone to use**, select **Internet Zone (less secure but more functional)**

Outlook 2000

1.From the menu bar, choose **Tools**, then **Internet Options**

2.Select **Security** Tab

3.Under **Secure Content**, select **Internet Zone**

Problems with E-Mail Stationery in Outlook 2002

If you have the **Preview Pane** enabled, note that Outlook 2002 will never display stationery as intended in the preview pane. It always has "scripting" turned off for the preview pane and cannot display advanced effects there.

To view a received stationery in Outlook 2002 you must double-click the e-mail to open it in a separate window, and then in the message window click on **View** and then under that on **View in Internet Zone**. (If you do not have this option in your **View** menu, please read the following paragraphs.)

There is also a feature in Outlook 2002 which is supposed to permanently enable the viewing of messages in the **Internet Zone** as above, so that you don't need to manually select this every time. This option is in the main Outlook 2002 window, under **Tools**, **Options...**, the **Security** tab. Under **Secure Content** beside **Zone** you are supposed to be able to select **Internet** instead of the default **Restricted Sites**.

Unfortunately, the Zone setting which is supposed to permit viewing of stationery has a bug and can just cause a worse problem. In the initial release of Outlook 2002, using this setting does not actually put you into the **Internet Zone** but it does remove the **View in Internet Zone** option from the message window's menu. The result is that you then can't view stationery at all! We hope that this bug will be fixed in a service pack. In the meantime, if you try using **Tools**, **Options**, **Security**, **Zone** to select the **Internet Zone**, and the result is that stationery still does not work, we recommend changing that setting back to

Restricted sites and then using the method described earlier when you view an e-mail stationery message.

See Also

- Problems with E-Mail Stationery in Outlook 2002
- Adding a Task by E-Mail (see page 69)
- Adding a Message by E-Mail (see page 70)

1.11.3 Using JES for E-Mail Integration

This section describes how to configure e-mail notification and e-mail submission for working with Java E-Mail Server (JES).

Description

To use e-mail notification and e-mail submission, TrackStudio needs an SMTP/POP3 server. You can use any SMTP/POP3 server, but TrackStudio SA already includes preconfigured Java E-Mail Server (JES) to make it easier to configure the program.

To use JES for e-mail notification:

1. Open the file **jes/etc/mail.conf** in a text editor. Un comment the **defaultsmtpservers** parameter and specify the address of your organization's SMTP server as its value.
2. Run Java E-Mail Server (**jes/jes**).
3. Run Server Manager (**sman**).
4. Go to the **E-Mail Notification** tab in **Server Manager**.
5. Check the **Enable e-mail notification** check box. Do not change the rest of the parameters.
6. Click the **Start** button to run TrackStudio.
7. Log in.
8. Specify your e-mail address in the user settings (the **User -> Edit** tab).
9. Enable e-mail notification for a filter.

How it works

1. A user modifies a task or creates a message.
2. If the task meets the filtering conditions of the e-mail notification rule, TrackStudio generates an e-mail message and sends it to JES.
3. JES redirects it to the server specified in the **defaultsmtpservers** parameter defined in the **mail.conf** file.
4. The mail server redirects the message to the user’s e-mail address.
5. The user receives the message using his e-mail client.

To use JES for e-mail submission:

1. Configure JES for e-mail notification as above and make sure it works.
2. Run Server Manager (**sman**).
3. Go to the **E-Mail Submission** tab in Server Manager.
4. Check the **Enable e-mail submission** check box. Do not change the rest of the parameters.
5. Click the **Start** button to run TrackStudio.
6. Change the SMTP server host in the e-mail client: **host = <JES IP>, port 25**. Now JES will process all your outgoing e-mail first and, if it has nothing to do with e-mail submission, it will be sent further.
7. Configure e-mail submission rules for some task (the **Task->E-Mail** import tab).
8. Send a message to trackstudio@127.0.0.1

How it works

1. The user sends a message to trackstudio@127.0.0.1
2. JES determines that the recipient’s address belongs to the local domain and sends it to the POP3 account of TrackStudio.
3. Once TrackStudio detects a message in its mailbox, it processes it and creates a task or a message.

See Also

- Receiving E-Mail Notification when Tasks Change (see page 66)
- Adding a Task by E-Mail (see page 69)

1.12 TrackStudio Configuration Properties

TrackStudio look for configuration files in the following order:

- **WEB-INF** subdirectory in **TrackStudio.war.** Use **jar** utility (supplied with JRE) to pack and unpack **TrackStudio.war**.
- If the TrackStudio home directory is explicitly specified by setting the **trackstudio.Home** property (case sensitive) using the **-D** option in **java** command line, the files are read relative to this directory:

```
> java -Dtrackstudio.Home=c:/trackstudio <options and parameters>
```

- path specified in **TS_CONFIG** environment variable

```
> set TS_CONFIG=c:/trackstudio
> java <options and parameters>
```

When you start a TrackStudio, following property files are loaded at startup:

File	Server Manager tab	Description
trackstudio.properties	**General**	A general-purpose configuration file.
trackstudio.hibernate.properties	**Database Connectivity**	Database connection configuration file.
trackstudio.mail.properties	**E-Mail Notification** **E-Mail Submission**	E-mail notification and e-mail submission configuration file.
trackstudio.ldap.properties	**LDAP Authorization**	LDAP configuration file
trackstudio.license.properties	**N/A**	TrackStudio license file. You should not modify it.
trackstudio.adapters.properties	**N/A**	TrackStudio adapters configuration. You should not modify it.

trackstudio.properties

Property	Server Manager (General tab)	Description	Example
trackstudio.siteURL	**HTTP port** **HTTPS port** **Host**	URL of your site. TrackStudio uses this URL to generate links (in e-mail notification messages, for example).	*http://www.mycompany.com:8080/TrackStudio*

trackstudio.logoutURL	**N/A**	Logout URL. The specified URL to load upon logout. If empty, goes to login screen.	*http://localhost:8888/TrackStudio*
trackstudio.uploadDir	**Upload directory**	Upload directory. Should exist and be accessible. We suggest you use the absolute (not relative) path here.	*/mnt/upload* *c:/TrackStudio/upload*
trackstudio.indexDir	**Index directory**	Full text search index directory. Should exist and be accessible. We suggest you use the absolute (not relative) path here.	*/mnt/index* *c:/TrackStudio/index*
trackstudio.indexMessages	**Index messages**	Index messages.	yes
trackstudio.debug	**Debug level**	Debug log level. Sets the type of log/debug messages that are sent to the log file.	*SILENT* *INFO* *DEBUG* *CALL*
trackstudio.encoding	**Character encoding**	Character encoding. Should match the codepage of the database.	*UTF-8*
java.protocol.handler.pkgs	**N/A**	Handler for SSL protocol.	*com.sun.net.ssl.internal.www.protocol*
trackstudio.cluster	**Cluster node**	TrackStudio cluster support.	*yes* *no*
trackstudio.cluster.name	**Cluster name**	TrackStudio cluster name.	*MyCluster*
trackstudio.soap	**Enable SOAP**	Allowed to use SOAP interface. Enable to use SCM and IDE integration.	*yes*
trackstudio.maxUploadSize	**Max upload file size**	Max size for uploaded files.	*52428800*
trackstudio.loginAsAnotherUser	**Enable logon as another user**	Allowed login as another user. To login as subordinate user, use their name and your password.	*yes*
trackstudio.skinPath	**N/A**	Skin path.	*/skins/defaultSkin*

trackstudio.hibernate.properties

Property	Server Manager (Database Connectivity tab)	Description	Example
hibernate.dialect	**Select DBMS**	SQL dialect	*net.sf.hibernate.dialect.HSQLDialect*
hibernate.connection.url	**URL**	JDBC connection URL	*jdbc:hsqldb:hsql://localhost*
hibernate.connection.driver_class	**JDBC driver**	JDBC driver class	*org.hsqldb.jdbcDriver*
hibernate.connection.username	**Login**	Database user	*sa*
hibernate.connection.password	**Password**	Database user password	

trackstudio.mail.properties

Property	Server Manager (E-Mail Notification/Submission tabs)	Description	Example
trackstudio.sendMail	**Enable e-mail notification**	Enable email notification	*yes* *no*
mail.transport.protocol	**Protocol**	Mail transport protocol. Should be smtp	*smtp*
mail.smtp.host	**SMTP server**	The SMTP server to connect to.	*127.0.0.1*
mail.smtp.port	**SMTP port**	The SMTP port to connect to.	
mail.from	**TrackStudio e-mail**	This sets the envelope **From** address.	*trackstudio@127.0.0.1*
mail.smtp.user	**SMTP server login**	SMTP user. Required only if SMTP server requires authentication	
mail.smtp.password	**SMTP server password**	SMTP password. Required only if SMTP server requires authentication	
mail.smtp.timeout	**N/A**	Socket I/O timeout value in milliseconds. Default is infinite timeout.	*10000*
mail.smtp.connectiontimeout	**N/A**	Socket connection timeout value in milliseconds. Default is infinite timeout.	*10000*
trackstudio.FormMailNotification	**Enable e-mail submission**	Enable e-mail submission. Enable this option with **trackstudio.sendMail yes** only.	*no*
mail.store.protocol	**Protocol**	Protocol	*pop3* *imap*
mail.store.host	**Mail server**	POP3/IMAP host	*127.0.0.1*
mail.store.port	**Mail port**	POP3/IMAP port	
mail.store.user	**Login**	Check this mailbox for email submission messages	*trackstudio@127.0.0.1*
mail.store.password	**Password**	Mail server password	*ChangeMe*

mail.store.forward	**Delete/forward unprocessed e-mails**	Delete or forward an invalid e-mail submission messages	*yes* *no*
mail.store.fwdaddress	**Forward unprocessed e-mails to**	Forward e-mail address (when mail.store.forward yes)	*admin@mycompany.co m*
mail.debug	**N/A**	SMTP/POP3/IMAP debug logs	*true* *false*

trackstudio.ldap.properties

Property	**Server Manager** **(LDAP Authorization)**	**Description**	**Example**
trackstudio.useLDAP	**Use LDAP authorization**	Specifies whether the authorization on the LDAP server is used. If the parameter is set to *yes*, the user authorization on the LDAP server will be performed alongside the usual authorization in the TrackStudio system.	*yes* *no*
ldap.host	**Server host**	Specifies the LDAP server address.	*192.168.22.10*
ldap.port	**Server port**	Specifies the server port.	*389*
ldap.baseDN	**Base DN**	Specifies the base DN. TrackStudio uses the specified DN for the user authentication.	*cn=users,dc=ldap-server,dc=my-compa ny,dc=com*
ldap.userDN	**User DN**	Specifies the user DN, which is connected to the LDAP server. Objects (users, groups, computers) in the LDAP directory are referred to by the **cn** attribute -- the **Common Name**. Containers, which may contain many objects, are also referred to by the **cn** attribute. LDAP supports special containers -- **Organizational Units** and **Domain Components**. Part of the binding string composed of **Domain Component** elements is the DNS domain name. For example, the *cn=TrackStudio* user above is in the *cn=users* container, which is in the *dc=ldap-server,dc=my-company,dc=com* DNS domain (sometimes referred to as *ldap-server.my-company.com*).	*cn=TrackStudio,cn=users,dc=ldap-serv er,dc=my-company,dc=com*
ldap.userDNpass	**User DN password**	Specifies the password for the user detailed in **ldap.userDN.**	

ldap.loginAttrTS	**Authorize by TrackStudio properties**	Specifies which user parameters are used for authorization on the server. It shows which of the TrackStudio user parameters is authorizes on the LDAP server.	*name* *login*
ldap.loginAttrLDAP	**Authorize by LDAP properties**	Specifies the property which should be used to search the user on the LDAP server. For example, if the **ldap.loginAttrLDAP** is *cn*, the common name is used to search the user.	*displayName* *sAMAccountName*

2 User's Guide

2.1 Demo Database Overview

This topic describes TrackStudio demo database

Description

This comprehensive sample demonstration describes how to manage products, versions, departments and customers in the database.

Organizational Structure

Suppose, we have a company - *Sample, Inc*. This company has the following organizational structure:

- *R&D Department*
- *QA Department*
- *Customer Support Department*
- *Project Management Department*
- *Sales Department*

Employees in departments have at least 2 managers (i.e their line manager and department manager). Each department manager is only responsible for their own employee. Each line manager is only responsible for their own project.

Sample, Inc has the following user statuses (groups):

- *administrator*
- *001 department manager*
- *010 line manager*
- *020 customer support member*
- *030 software developer*
- *040 software tester*
- *100 external customer*

Name each employee of *Sample, Inc*, their login/password and user status (group):

- *John Smith* - jsmith - administrator
  - *Peter Dagley* - pdagley - Project Management Department Manager
    - *John Baetz* - jbartz - Line Manager for *YTracker*
    - *Jesse Levon* - jlevon - Line Manager for *XWare*
  - *Steve Trudelle* - strudelle - Customer Support Department Manager
    - *Mike Clinton* - mclinton - Customer support member
  - *Sean Law* - slaw - QA Department Manager
    - *Jacob Miller* - jmiller - Software Tester for *XWare*
    - *Jeff Franke* - jfranke - Software Tester for *YTracker 1.0*
  - *Bill Richardson* - brichardson - R&D Department Manager
    - *Stuart Manske* - smanske - Software Developer for *XWare* and *YTracker*

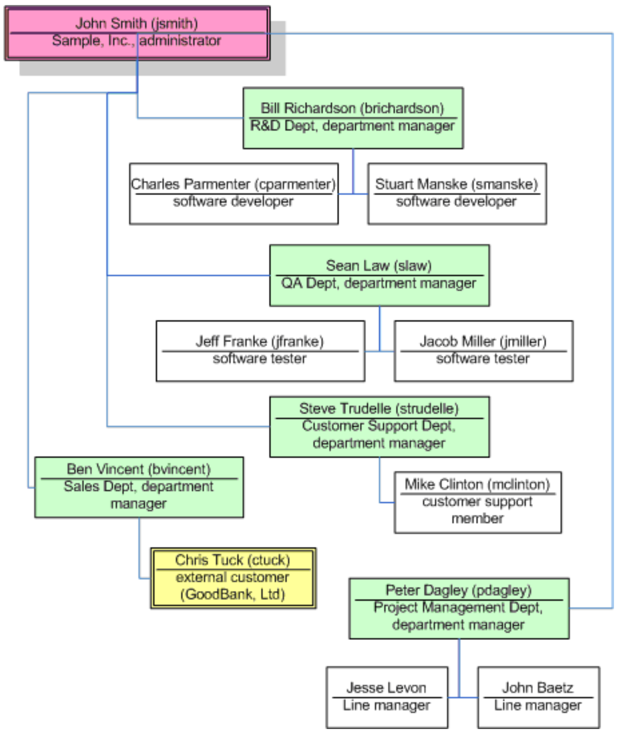
- *Charles Parmenter* - cparmenter - Software Developer for *XWare*
- *Ben Vincent* - bvincent - Sales Department Manager

Ben Vincent will manage customers. He will define

- additional custom field for the customer - *Address*, check out the **Customize** tab for *Ben Vincent.*
- the customer self-registration rule - check out the **Registration** tab for *Ben Vincent*. All new customers can create their bugs in the *Customer Support* project.

Samples, Inc also has a customer - *Chris Tuck*, from *GoodBank*. He can login as ctuck/ctuck and add a bug into the system. Also he can create subordinate users from his company to add bug reports. You can limit how many accounts each customer can create and set an expiry date for your customer as defined in your support contract (one year, for example).

A user's password is the same as his/her login. Login as root/root or jsmith/jsmith to examine the organizational structure.



Projects

Sample, Inc develops two software products: *XWare* and *YTracker*. Currently they have 2 versions of *XWare*: 1.0 (already mature) and 1.5 (active). *YTracker* is still in development and they have no customer support for it.

YTracker Team

User Name	Role
John Baetz	project manager
Jeff Franke	software tester
Stuart Manske	software developer

XWare Team

User Name	Role
Jesse Levon	project manager
Stuart Manske	software developer
Charles Parmenter	software developer
Jacob Miller	software tester

Ben Vincent is the sales representative and *Mike Clinton* is the customer support member for both products.

Permissions

Now let's check out how user groups are be defined. Login as jsmith and check out **User Management -> Statuses**. Note that each user group has a parent user group. A group can't have more permissions than the parent group has. If you disable some permission for parent group - it will be disabled for all child groups automatically.

With user groups you can control every tab and every button the user can view. You can also set which fields the user can see or edit. When you grant any permission, all dependent permissions will be granted automatically.

Workflows

This team also uses the following workflows:

Workflow Name	Description
Folder	A simple container for other objects, one state, no transitions.
Product Lifecycle	Implements the life cycle for products and software releases. Contains states: • 001 Development • 002 Active • 003 Mature • 004 Retired Contains two workflow-specific custom fields: • GA Date - General Availability • Availability This workflow is used primary by management.
Issue Lifecycle	Implements the life cycle for software bugs or issues. Contains steps: • 001 New • 002 Resolved • 003 Verified • 004 Closed • 005 Waiting Feedback - used for customer support task when our team waiting response from customer

Take close look at permissions defined for the *Issue Lifecycle* workflow

Message Type	Description	Permissions
001 Resolve	Resolve a bug	• Any administrator, department and line manager or customer support member can *resolve* tasks. • Tester or developer can *resolve* a task only when their task handler. • External customers cannot see the *resolve* comments from team members or add such messages.
002 Verify	Verify a bug	• Any administrator, or department and line manager can *verify* tasks. • Software testers can *verify* task only when their task handler. • Supporters, developers and customers cannot add such messages. • Customers cannot see such messages.

003 Close	Close a bug	• Any users except software developers, testers and customers can *close* tasks. • Customers can't see *close* messages from other users
004 Reopen	Reopen a bug	• All users, except customers, can *reopen* tasks. • Customers cannot see *reopen* messages from other users.
000 Note	Add a note	• Any users, except customers, can see *notes* • Customers cannot see notes or add them. This message type is only for internal communication.
005 Request to Customer	Question or note to customer	• Any team member can ask for additional info from a customer
006 Response from Customer	Feedback from customer	• Customer can respond to a bug only when asked (bug handler)

Categories

When user creates a bug - he/she doesn't specify workflow directly. Instead, he/she should choose task categories (such as *Software Bug*) that are connected to some workflow. Several categories can be connected to the same workflow.

For each category, you can specify who can create, edit, or delete tasks within this category and define the category's dependency (for example, *Folder* can contains *Software Bug*, but *Software Bug* can't contains *Folder*).

Assigning Employee to Product

Product managers cannot assign developers or testers directly to projects - they need approval from department managers. Check out task #11 where *YTracker's* project manager, *John Baetz,* asks R&D department manager, *Bill Richardson,* about 1 developer for his project. *Bill* approved it and assign *Stuart Manske* to *YTracker* as software developer (check out the **Access Control -> Assigned Statuses** tab for YTracker #7). Department managers can use the *My Assignment Requests* filter to find their assignment requests.

Implementing Calculated Custom Fields

We store the user departments in the *Company* field. We would like a report that shows the number of tasks created from each department.

As *Company* field is a user field, it cannot be referenced by a task filter directly. To solve this, we create a task custom field that uses a script which will return the *Company* field.

Login as jsmith/jsmith and go to the **User Management->Scripts** tab. Check out the *getDepartment* script which will return department names when the task submitter is you or your subordinate user, and will return *Unknown* otherwise. For example, *John Smith* (administrator) can see the department names for all tasks, *Sean Law* (QA Department Manager) can see department names only for tasks submitted by testers and himself; *Jacob Miller* can see department names only for his tasks.

```
if (task.getSubmitter()==null || task.getSubmitter().getCompany()==null)
   return "Unknown";

return task.getSubmitter().getCompany();
```

We've connected the *getDepartment* script to the *Submitter Department* task custom field (check out the **Task -> Customize** tab for the task #2).

- To see a list of tasks with submitter's department name, use the *Tasks by department breakdown* filter.
- To see the number of tasks and each task's status submitted by users from each department, use the *Tasks by department breakdown* distribution report. Note that different users will be able to see different data in this report -- depending on permissions. Just compare it for administrator, department manager and software tester.

We also use the *getCustomerAddress* script that returns the addresses of customer that have submitted a bug. This script used by the *Customer Address* custom field of the task #5. Go to the task #24 (submitted by customer) to check how it works.

The calculated custom fields option is slow and should only be used when absolutely necessary.

Filters

Filters are used to search tasks by criteria. The following public filters are available for all projects in *Sample, Inc*.

Filter Name	Description
All	Returns all direct subtasks of the current task.
Change List	Returns list of closed bugs. You can use this filter for any project or project version.
My Assignment Requests	Functional department managers can use this filter to find all requests issued to assign an employee to a project.
Bugs (Personal and Subordinate)	Returns a list of bugs that should be fixed by logged user or subordinate users. This filter is very useful for developers, testers and their managers.
Roadmap	Returns a list of opened bugs.
Tasks by department breakdown	Returns a list of open bugs with info about the submitter’s department.

2.2 Implementation Guide

This topic describes how to configure TrackStudio.

Description

To configure the demo database for your company:

Step	How to
1. Login as **root**	
2. Use the **User Management -> Overview** tab to create managed administrator.	Creating a User Account (see page 50)
3. Use the **User Management -> User -> Change Password** tab to set password for the managed administrator.	Changing a Password (see page 53)
4. Use the **Task Management -> Overview** tab to create the root folder for your projects.	Creating a Project (see page 43)
5. Use the **Task Management -> Access Control -> Assigned Statuses** tab to give the managed administrator permission to manage your projects.	Granting User Access to a Project (see page 51)
6. Login as managed administrator.	
7. Use the **User Management -> Statuses** tab to configure user groups and set permissions for each user group.	Establishing a User Group Account (see page 49)
8. Use the **User Management -> Overview** tab to create user accounts for your developers, testers and customers.	Creating a User Account (see page 50)
9. Use the **Task Management -> Workflows** tab to configure workflows for your projects and bugs. Set workflow permissions for your user statuses.	Creating a Workflow (see page 65)
10. Use the **Task Management -> Categories** tabs to create categories for your projects and tasks. Set category permissions for your user statuses.	Creating a Category (see page 65)
11. Use the **Task Management -> Overview** tabs to create your project hierarchy.	Creating a Project (see page 43)

12. Use the **Task Management -> Access Control -> Assigned Statuses** tabs to allow your users to view projects and fill-in bugs.	Granting User Access to a Project (see page 51)
13. Use **Task->Customize** and **Workflows->Customize** tabs to create custom fields for your projects and bugs.	Adding a Custom Field (see page 55)
14. Use the **Filters** tab to customize task filters.	Filtering Subtasks by Properties (see page 61)
15. Login as **root**.	
16. Use the **User Management -> User -> Change Password** tab to change the **root's** password.	Changing a Password (see page 53)
17. Use the **User Management -> User -> Edit** tab to de-activate **jsmith's** account (all subordinated accounts will be deactivated automatically).	Locking a User Account (see page 53)
18. Use filters to search tasks by complex criteria.	Filtering Subtasks by Properties (see page 61)
19. Use reporting to manage project resources and analyze what work has been completed and what work remains to be done.	Generating a Report (see page 48)
20. Use SOAP API to integrate TrackStudio with a call-center, front-office and other third-party software.	Integrating with Third-Party Systems (see page 85)

2.3 Concepts

This topic describes TrackStudio Concepts.

Description

TrackStudio is a hierarchical database of objects. There are 12 main types of object that you need to be aware of:

- Tasks (items)
- Users
- Workflows
- Messages
- Categories
- User groups (or user status)
- Filters
- Reports
- Custom fields
- E-Mail templates
- Scripts
- External user self-registration rules

In brief, tasks (items) are objects whose state is tracked and updated by users in the system. Every task is of a particular category. Categories are objects like *Issue, Risk, Incident, Software Bug*, etc. When you define a category, you assign it a workflow. This is where TrackStudio becomes an extremely powerful management tool – The workflow defines the states that are allowed for a task (item), how it transitions between states, what type of user can transition the item to another state, and the custom fields that can be captured in the task.

When developing TrackStudio, we tried to use as few objects and concepts in the system as possible and attempted to realize the necessary functionality through enhancing the objects already existing in the system. This synergy allowed us to create a system that's powerful, yet easy-to-use-and-understand.

2.3.1 User Interface Concepts

This topic describes basic user interface concepts.

Description

Navigation Tree

The user interface of TrackStudio is optimized for working efficiently with a large number of tasks and users. To jump quickly to the task or the user you need, use the **Navigation Tree** in the left frame. For tasks, you can see the name, the number and the amount of subtasks. For users you can see their name and the number of subordinated users.

Main Menu

Use the main menu to switch between the working modes of the system: **Task Management** or **User Management**. The **User Management** mode is used to manage users and user groups. The **Task Management** mode is used to manage projects and tasks, build reports, etc.

Use the input field in the right part of the **Main Menu** to jump to the task by number or alias.

Full Path

Nearly all operations in TrackStudio are performed for the current task or with the current user. The **Full Path** field under the **Main Menu** contains the path to the current user or the current task. All the information below the **Full Path** field is related to the current task or to the current user depending on the working mode.

Tabs

Web pages with common functionality are grouped using the tabs. You can control what tabs are accessible for each of your users.

Object lists.

Most objects related to a task or user are displayed as lists in TrackStudio.

• To select an object from the list, click the radio button near it.

• To edit an object, go to the corresponding second level tab.

• To delete an object, select the **Delete** check box beside it and click the Delete button.

• To copy an object, click the **copy** icon beside it.

2.3.2 Task Concepts

Use tasks to represent the state of an object to be tracked (project, bug, defect, version, component, module, etc).

Description

A task (item) in TrackStudio is a very generic concept. It is an instance of a category, and as you will learn further on, you can define a category with a particular workflow and custom fields to represent anything you like.

TrackStudio stores tasks in a hierarchy. The best way to explain this is with an example: after you log on to the system, you can see a task called *Sample, Inc*. This example company have configured their task hierarchy as follows: There are two folders at the top level called *Products* and *Customer Support*. Both of these items of category *Folder* contains items of category *Product*. When you click on one of the products in the *Products* folder, you will notice that the purpose is to track

Product Version tasks. You will notice fairly quickly that all items, regardless of the category, have certain standard fields.

All the created objects (filters, workflows, custom fields) can be inherited, which means you do not have to declare the same custom fields and workflows for each new project. It is enough just to create a new project and it will automatically inherit all the properties common for this project group. You can also enhance the inherited objects to make them fit the specific project. To create a global object available for every task in the system, you must create an object attached to the root task.

Example

Suppose some bugs appear in versions of your software both for Windows and Linux. The versions for different platforms are being developed by different teams and you need to track the bug fixes individually for every version. There are two common ways to ensure this:

- You can add to the system one task, add messages about the systems in which it appears and track the state by adding more messages. This way you cannot track the exact time of fixing the bug, the time spent for each OS, etc.
- You can make individual copies of the task for each OS. In this case you will have problems tracking the total time spent on this task and discussing the problems common for both platforms. This method may also lead to numerous bugs and problems in managing them when the bug becomes apparent in later beta versions of the product.

In TrackStudio you can create several versions of each task (in fact, each of them is a lower level task of a special kind) and set up individual properties (handler, access rights etc.) for each of them. You will be able to view both the general information on the task (e.g. the total time spent on fixing the bug in all versions) and the version or configuration-specific information (e.g. the list of all bugs not yet fixed in the Windows version).

2.3.3 User Concepts

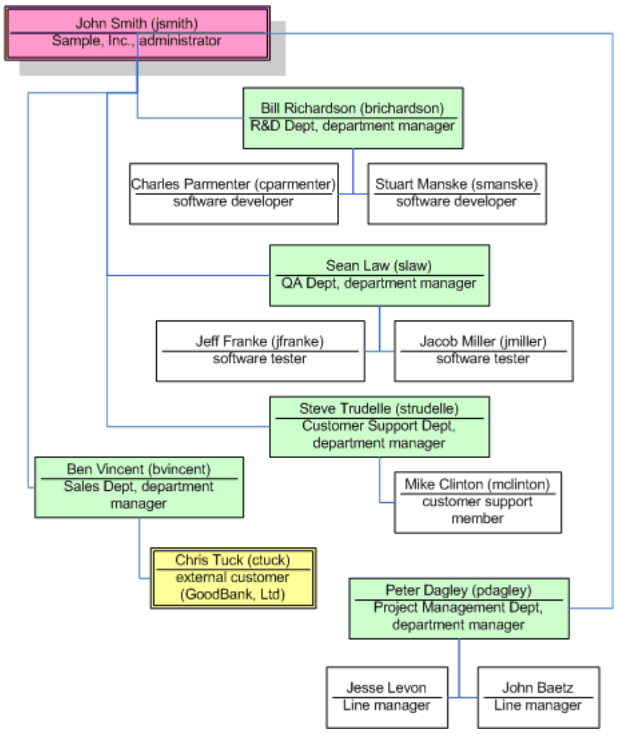
Use users to manage your employees, departments and customers.

Description

A “User” is a user of TrackStudio, with a logon id, password and certain access rights. As for tasks, TrackStudio supports a hierarchy of users, which makes it possible to manage the system effectively in medium sized and large organizations. TrackStudio allows transferring a part of the authority for managing the system to subordinate managers. Subordinate managers can have the same rights and privileges for their parts of a project as the project manager does over the whole project. These lower-level managers can use shared project rules, user roles, item categories, custom fields, and workflows, and can modify them according to their specific needs. The project manager has access to all the information and can create reports, and analysis relating to the project as a whole or to individual elements.

Example

Sample, Inc organizational structure



2.3.4 Workflow Concepts

Use workflows to define the life cycle of your issues.

Description

Workflows

The behavior of any task in the system is defined through its workflow. A workflow allow you to set rules to change the states of a task. The workflow contains a set of priorities, states, message types, transitions, and custom fields.

Priorities

A priority is used to specify the resolution order of the suggested task. You can specify priorities for each workflow.

States

A state is the position in the workflow where a task resides. While a task is in a certain state, it corresponds to a handler (the user responsible for a certain stage in the process of working with this task). This user must do his work before changing the state of this task and assigning another handler to it.

Specify the **Start** state among the created task states. When a task is created, the system will automatically set its state to the specified start state.

Then specify **Final** task states. Once a task reaches the specified final state, the system will automatically update the task **Close Date**. You can specify several final states or none.

Message Types

To change a task state, the user should add a message. One or several similar transitions can be grouped into each message type. A transition is a unidirectional link between two workflow states. Each message can be configured with permissions such that, for example, only a *manager* can perform the *close* message on an item.

Workflow-specific custom fields

You may want to create some properties such as *version*, *release*, *platform*, etc. These custom fields will only be available for the tasks with the edited workflow.

2.3.5 Message Concepts

Use messages to change task state, assign handlers and add comments.

Description

Use messages for:

- changing the task state: You can change the state of a task using messages. To do this, select the **Message Type** that will change the task from its current state into the one you need while creating a message. Task states and transitions between them depend on the task workflow determined by the category of the task.
- arranging interaction between users: The **Handler** task field defines the user whose response is necessary to continue processing the task. If you need another user work on task (for example: you have a code that needs to be tested or you need to ask another developer a question), specify the needed user in the **Handler** field and add the message.
- monitoring the progress in working on a task: In the process of working on a task, you can add messages describing the current state of affairs and specify the time spent on the task.

2.3.6 Category Concepts

Use categories to define the type of a task and to link a workflow to a tasks.

Description

Categories can be thought of as “task types”. Categories define the type of a task and are used to create links between tasks and workflows. When creating a category, you can specify its possible subcategories and the user groups who can create and delete tasks of that type. For example, you can specify that only a *manager* can create tasks of the *project* type, or that a *bug* cannot have a *project* as a subtask. The most basic pre-defined category is the *Folder*. The workflow for a *Folder* is simple: it has one state, no transitions, and no custom fields. The purpose of the *Folder* is simply to allow you to group tasks together, and to define security rules for accessing those subtasks.

2.3.7 User Status Concepts

Use user status to assign permissions to users.

Description

User status represents a user group. Each user status can have a specific set of privileges. User groups have a hierarchical structure (a subordinated user group cannot have more privileges than a parent user status). A user can be included in several different groups and the user status set can be different for different projects.

2.3.8 Security Concepts

This topic describes how to configure user permissions.

Description

Effective statuses determine:

- what tabs users can view (**Statuses -> Edit** tab in the **User Management** mode).
- what buttons users can press (**Statuses -> Edit** tab in the **User Management** mode).
- what fields users can view and edit (**Statuses -> Edit** tab in the **User Management** mode).
- what tasks users can create, modify and delete (**Categories -> Permissions** tab in the **Task Management** mode).
- what messages users can view and create (**Workflows -> Permissions** tab in the **Task Management** mode).

Effective statuses are based on

• user's **own status** -- specified either when a user is created or with the **User -> Edit** tab in the **User Management** mode.

• **assigned statuses** -- specified on the **Access Control -> Assigned Statuses** tab in the **Task Management** mode.

While you are working in the **User Management** mode, the list of effective statuses contains only one item – the user's own status.

While you are working in the **Task Management** mode, the list of effective statuses is estimated on the basis of the user's own status and assigned statuses. To perform a particular action, the action should be permitted for least one of the user's effective statuses.

- **Accessible tasks** are tasks (and subtasks) for which a user has an assigned status.
- **Restricted tasks** are tasks which lie between the root task and accessible tasks. You can use restricted tasks only for navigating. You can neither view the description of restricted tasks nor modify them
- Accessible tasks and restricted tasks are called **visible tasks**.
- You cannot access **invisible tasks**.

• To make a task accessible for the user and to apply the user's own status as the effective status for this task (a *developer* user must have the *developer* permissions for this task), create an assigned status with the default parameters.

• To make a task accessible for the user and to extend the list of effective statuses (a *developer* user must have both the *developer* and *manager* permissions for the task), create the necessary assigned statuses for the task. You do not have to define the user's own status and assigned statuses when they are already specified for the parent tasks. They will be inherited automatically.

• To make a task accessible for the user and to completely redefine the list of effective statuses (a *developer* user must have only the *manager* permissions for the task), create the necessary assigned status for the task and select the override check box. In this case the user's own status and assigned statuses specified for the parent tasks are not taken into account when effective status is estimated.

All statuses (except the *administrator* status) have one parent status and form a hierarchy of statuses. Which permissions you can set for a status depend on the position of the status in the hierarchy: a child status cannot have broader permissions than its parent status.

To configure:

- what tabs users can view
- what buttons users can press
- what fields users can view and edit

use the **Statuses -> Edit** tab in the **User Management** mode.

To configure:

- what tasks users can create
- what tasks users can modify
- what tasks users can delete

use the **Categories -> Permissions** tab in the **Task Management** mode.

To configure:

- what messages users can view
- what messages users can process (create)

use the **Workflows -> Permissions** tab in the **Task Management** mode.

2.3.9 Filter Concepts

Use filters for filtering users, tasks and messages, setting the e-mail notification rules and designing reports.

Description

Filters allow you to display and search for tasks and users meeting specified criteria. As with all the other objects, filters are inherited, which saves time configuring filters for each project. Filters can be either private or shared. Private filters are available only for the user who created them. You can only create private filters for restricted tasks.

2.3.10 Report Concepts

Use reports to print task information.

Description

Reports are defined by a filter against a task.

- Use the **List** type to display the list of tasks as a table. To configure the columns and the sorting order, specify filtering conditions.
- Use the **Detailed** type to display all task fields that have values.
- Use the **Distribution** type to group tasks and to calculate the aggregate functions (**Avg, Min, Max, Sum**) for each group.
- Use the **UserWorkload** type to display information about the time (**Actual Budget**) your subordinates worked on subtasks of this task.

The created report will be available both for the current task and for its subtasks. The reports can be generated in the form of HTML, PDF, CSV, XML or XLS.

2.3.11 Custom Field Concepts

Use custom fields to enlarge the list of fields available for tasks and users.

Description

User can fill-in field manually or it can be calculated dynamically. Calculated custom fields are especially useful in filters, email notification rules, and distribution reports. Calculated custom fields can be implemented via scripts.

See Also

- Script Concepts (see page 40)
- Adding a Custom Field (see page 55)
- Calculating Custom Field Value (see page 56)
- Custom Field Properties (see page 73)

2.3.12 Script Concepts

Script represents a description on the Java-like language of the calculated custom field algorithm.

Description

To create a calculated custom field, you should create a static (non-calculated) custom field and then specify a script for it. If a custom field has no script, it is considered static. So, if you delete the expression from a script, all custom fields that use it become static. No results is saved to the database for a calculated field.

TrackStudio uses Java-like language based on BeanShell to evaluate expressions. It means that you can create not only basic mathematical expressions, but also more complex expressions (**if, for**, or **while**). You can also use some Java classes. For security reasons you can only use following classes in your expressions:

```
java.lang.Boolean
java.lang.Byte
java.lang.Character
java.lang.Class
java.lang.Comparable
java.lang.Double
java.lang.Float
java.lang.Integer
java.lang.Long
java.lang.Math
java.lang.Number
java.lang.Object
java.lang.Short
java.lang.String
java.lang.StrictMath
java.lang.StringBuffer
java.util.*
java.text.*
java.sql.Date
java.sql.Time
java.sql.Timestamp
gran.secured.*
gran.tools.Logger
```

To use any other classes in your expressions change the class **gran.tools.ShellClassLoader** and add the classes and packages that you need. If the expression is incorrect or if the calculated value type does not match the field type, the result will be an empty field (**null**).

TrackStudio allows you to create calculated custom fields of the same types as static custom fields. (i.e. **Integer, Float, String, Date, List,** etc) You should ensure that the result matches the required type. For example, to get the result of date calculation in milliseconds (long), you should convert it to the **Date** type:

```
new Date(milliseconds)
```

You can use the following constants:

Constant	Type	Description
DAYS	long	msec/day
HOURS	long	msec/hour
MINUTES	long	msec/minute
SECONDS	long	msec/second

2.3.13 External User Self-Registration Concepts

Users can register in the system without the participation of the administrator and can gain access to certain tasks.

Description

If the system has at least one registration rule, a user will see the additional **Register** button on the **Login** page. Use this button to access the **Registration** page. If there are no registration rules, the user will not be able to register in the system. The project list on the registration page contains the names of all existing registration rules. To make the registration easier, give your users the direct **URL for registration**.

2.3.14 E-Mail Template Concepts

Use e-mail templates to customize e-mail notification messages.

Description

To generate the e-mail subject and body according to the template TrackStudio use the FreeMarker template engine -- a generic tool to generate text output based on templates. Please refer the FreeMarker manual for more information. Use standard TrackStudio’s templates as a basis to create your own templates. You can specify a separate e-mail template for each user and task.

2.3.15 E-Mail Notification Concepts

TrackStudio can send out e-mail notifications when a certain pre-specified event occurs or at regular intervals.

Description

Sending out e-mail notifications at regular intervals is performed according to the specified schedule irrespective of the events occurring in the system. Activate filter subscription on the **Filters->Subscribe** tab.

TrackStudio can send out e-mail notifications when the following events occur:

- a new task is added to the system.
- an existing task is modified.
- a message is added to the task or other associated events (changing the handler, entering the elapsed or estimated time by the developer, etc).

Activate event-based e-mail notification on the **Filters->Notifications** tab. E-Mail notification rules do not depend on the filter currently selected on the **Subtasks** tab.

When defining an e-mail notification rule it is important to differentiate between the following tasks:

- The parent task for the filter -- this task is displayed in the **Connected to** column on the **Filters->List** tab. This task determines for which tasks the filter will be visible; it does not directly influence the e-mail notification.
- The task for which the e-mail notification is activated. The e-mail notification rules for a task is displayed on the **Filters->Notifications** tab. The e-mail notification will be sent when this task or one of its subtasks are modified.
- The modified task -- modifying this task results in email notifications.

The email notification system checks the filtering conditions a bit different from the usual routine. That is why a filter may be useful for the e-mail notification system, but often cannot be used to filter tasks.

When you create a new task or update an existing one

The notification will be sent if the task meets the filtering conditions for tasks. If there are no filtering conditions specified, the e-mail notification will be sent when any task is modified. When a task is created or modified, filtering conditions for messages are not checked even if they are specified. The following parameters are also ignored:

- **Tasks per Page** (check all tasks)
- **Deep Search** (always check subtasks recursively)
- **Sort**
- **Hide**

When a new task is created, the e-mail notification is sent once the **Save** button is pressed -- not the **Create a Task** button. If the **Save** button is not pressed, the e-mail notification will not be sent.

When you create a new message

TrackStudio checks is whether the task meets task filtering conditions and the added message meets the filtering conditions for messages. If there are no filtering conditions for messages specified, the e-mail notification is sent when any message is added. The following parameters are ignored:

- **Tasks per Page** (check all tasks)
- **Deep Search** (always check subtasks recursively)
- **Sort**
- **Hide**
- **View Messages** (always check new message only)
- **Filter Messages** (always check new message only)
- **Bulk Processing Tool**

The **Current user** filter property means different things in the different contexts:

• When filtering tasks or generating reports, it means the logged user.

• When processing filter subscription rules, it means the subscribed user.

• When processing event-based e-mail notification rules, it means the user who has modified the task or added a message.

2.3.16 E-Mail Submission Concepts

Users with TrackStudio accounts can submit tasks or messages by e-mail.

Description

Use e-mail submission to:

• Create new tasks by e-mail.

• Submit messages to existing tasks.

• Upload files.

If the e-mail submission option is enabled, TrackStudio checks the mailbox at regular intervals. TrackStudio gets an e-mail from the mailbox and tries to import it.

- If the e-mail meets the requirements of one of the mail submission schemes, the system imports the e-mail as a new task or message.
- If the e-mail is a reply from a mailer-daemon about a failure while sending the e-mail, it is deleted from the queue.
- If the message cannot be imported and at the same time it is not a reply from a mailer-daemon, it is either deleted from the queue or forwarded to the specified address.

2.4 Step-by-Step Guides

2.4.1 Managing Projects and Bugs

2.4.1.1 Creating a Project

This topic describes how to create a project.

Description

To create a project:

1. Choose a parent task for your project.
2. Click the **Overview** tab.
3. Select a category for the project that will allow you to create the subtasks you need in this category.
4. Enter the project name.
5. Click the **Create a Task** button.
6. Fill-in project properties
7. Click the **Save** button.

Conditions

- You cannot create a project if you do not have permissions to create subtasks for the current task. To check your category-related permissions use the **Categories->Permissions** tab.

See Also

- Task Concepts (see page 34)
- Finding a Task (see page 61)
- Defining Category Dependency (see page 66)
- Defining Users Who Can Create, Edit or Delete Task (see page 60)
- Task Properties (see page 72)

2.4.1.2 Adding a Bug

This topic describes how to add a bug or issue.

Description

To add a bug:

1. Choose a parent project for your bug.
2. Click the **Overview** tab.
3. Select the bug category.
4. Enter the bug summary.
5. Click the **Create a Task** button.

6. Fill-in task properties
7. Click the **Save** button.

Conditions

- You cannot add a bug if you do not have permissions to create subtasks for your project task. To check your category-related permissions use the **Categories->Permissions** tab.

See Also

- Task Concepts (see page 34)
- Finding a Task (see page 61)
- Defining Category Dependency (see page 66)
- Defining Users Who Can Create, Edit or Delete Task (see page 60)
- Task Properties (see page 72)

2.4.1.3 Editing Task Properties

This topic describes how to edit task properties.

Description

To edit task properties:

1. Click the **Task -> Edit** tab.
2. Enter task properties
3. Click the **Save** button

See Also

- Task Properties (see page 72)
- Bulk Updating Tasks (see page 48)
- Receiving E-Mail Notification when Tasks Change (see page 66)

2.4.1.4 Assigning a Task

This topic describes how to assign a task.

Description

When you create a task use the **Task->Edit** tab to specify a task handler.

To assign an existing task :

- Click the **Task -> Messages** tab.
- Choose a new task handler.
- Click the **Save** button.

Conditions

You can't assign a task if:

- You do not have permissions to add messages for the task (**Statuses -> Edit** tab in the **User Management** mode).
- You do not have permissions to change the task handler (**Workflows -> Permissions** tab in the **Task Management** mode, **Statuses -> Edit** tab in the **User Management** mode).
- There are no message types available for the task in the current state (**Workflow -> Transitions** tab in the **Task Management** mode).
- There are no available handlers for this task (**Access Control** tab in the **Task Management** mode).

See Also

- Task Concepts (see page 34)
- Adding Comments to a Task (see page 45)
- Setting Default Handler for a Project (see page 55)
- Restricting Handler List (see page 59)
- Receiving E-Mail Notification when Tasks Change (see page 66)
- Task Properties (see page 72)
- Message Properties (see page 73)

2.4.1.5 Adding Comments to a Task

This topic describes how to add comments to a task.

Description

To add comments to a task:

1. Click the **Task -> Messages** tab.
2. Fill-in the message description with the comment information.
3. **Optional:** Fill-in other task properties.
4. Click the **Save** button.

Conditions

You can't add a comment to a task if:

- You do not have permissions to add messages for the task (**Statuses -> Edit** tab in the **User Management** mode).
- There are no message types available for the task in the current state (**Workflows -> Transitions** tab in the **Task Management** mode).

See Also

- Task Concepts (see page 34)
- Setting Default Handler for a Project (see page 55)
- Restricting Handler List (see page 59)
- Receiving E-Mail Notification when Tasks Change (see page 66)
- Task Properties (see page 72)
- Message Properties (see page 73)

2.4.1.6 Changing Task State

This topic describes how to change task state.

Description

Task state is tied to task workflow, and progressing a task through its workflow changes its state accordingly. To change task state:

1. Click the **Task -> Messages** tab.
2. Choose the message type that will move your task to the required target state.
3. **Optional:** Fill-in other message properties.
4. Click the **Save** button.

Conditions

You can't change task state if:

- You do not have permissions to add messages to the task (**Statuses -> Edit** tab in the **User Management** mode).
- There are no message types that move the task to its target state available (**Workflows -> Transitions** in the **Task Management** mode).
- You do not have permissions to add messages of the required type (**Workflows -> Permissions** in the **Task Management** mode).

See Also

- Task Concepts (see page 34)
- Adding Comments to a Task (see page 45)
- Restricting Handler List (see page 59)
- Restricting Who Can Close Task (see page 59)
- Receiving E-Mail Notification when Tasks Change (see page 66)
- Closing a Bug (see page 48)
- Task Properties (see page 72)
- Message Properties (see page 73)

2.4.1.7 Attaching a File to a Task

This topic describes how to attach a file to a task.

Description

To attach a file to a task:

1. Click the **Task -> Uploads** tab.
2. Fill-in the file name and description.
3. Click the **Upload** button.

Conditions

You can't attach a file to the task if:

- You do not have permissions to upload files for this task (**Statuses -> Edit** tab in the **User Management** mode).
- The file is larger then the maximum upload file size (**trackstudio.maxUploadSize** property in **trackstudio.properties**).

See Also

- TrackStudio Configuration Properties (see page 23)

2.4.1.8 Moving Tasks

This topic describes how to move a task

Description

To move a task:

1. Click the **Task -> Edit** tab.
2. Enter a new parent task number.
3. Click the **Save** button.

See Also

- Task Concepts (see page 34)

- Task Properties (see page 72)

2.4.1.9 Linking Tasks

This topic describes how to link tasks.

Before You Begin

- Create a custom field with **Task** type accessible for the task.

Description

To link tasks:

1. Click the **Task -> Edit** tab.
2. Use the custom field with **Task** type to mark tasks linked to this task.
3. Click the **Save** button.

See Also

- Task Concepts (see page 34)
- Custom Field Concepts (see page 39)
- Adding a Custom Field (see page 55)
- Custom Field Properties (see page 73)

2.4.1.10 Linking User to a Task

This topic describes how to link users to a task.

Before You Begin

- Create a custom field with **User** type which is accessible for the task.

Description

To link users to a task:

1. Click the **Task -> Edit** tab.
2. Use the custom field with **User** type to mark users related to this task.
3. Click the **Save** button.

See Also

- Task Concepts (see page 34)
- Custom Field Concepts (see page 39)
- Adding a Custom Field (see page 55)
- Custom Field Properties (see page 73)

2.4.1.11 Exporting Task

This topic describes how to export tasks.

Description

To export tasks:

1. Click the **Task -> XML Export** tab.
2. Enter XML export options.
3. Click the **Submit** button.

See Also

- Importing and Exporting Database (see page 15)
- Importing from Third-Party Systems (see page 16)

2.4.1.12 Generating a Report

This topic describes how to generate a report.

Description

To generate a report:

1. Create a filter to select tasks for the report.
2. Click the **Reports -> List** tab.
3. Choose existing report or create a new one.
4. Click the **Reports -> View** tab.
5. Fill-in report properties.
6. Click the **Submit** button.

See Also

- Filtering Tasks Using AND/OR/NOT (see page 62)
- Analyzing Tasks Distribution (see page 63)

2.4.1.13 Bulk Updating Tasks

This topic describes how to update multiple tasks.

Description

To update multiple tasks:

1. Click the **Filters -> List** tab.
2. Choose an existing filter or create a new one.
3. Click the **Filters -> Edit** tab.
4. Check the **Bulk Processing Tool** setting.
5. Click the **Save** button.
6. Click the **Subtasks** tab.
7. Use your filter to show task list.
8. Fill-in message properties for the tasks. To add a message using the **Bulk Processing Tool** you should specify non-empty **Description** for it.
9. Click the **Save** button.

See Also

- Editing Task Properties (see page 44)
- Message Properties (see page 73)
- Task Filter Properties (see page 78)

2.4.1.14 Closing a Bug

This topic describes how to close bugs.

Description

To close a bug:

1. Click the **Task -> Messages** tab.
2. Choose the appropriate message type.
3. **Optional:** Fill-in other message properties.
4. Click the **Save** button.

Conditions

You can't close a task if:

- You do not have permissions to add messages for the task (**Statuses -> Edit** tab in the **User Management** mode).
- The option is unavailable in the current task state (**Workflows -> Transitions** tab and **Workflows -> Permissions** tab in the **Task Management** mode).

See Also

- Task Concepts (see page 34)
- Adding Comments to a Task (see page 45)
- Restricting Handler List (see page 59)
- Restricting Who Can Close Task (see page 59)
- Changing Task State (see page 45)
- Task Properties (see page 72)
- Message Properties (see page 73)

2.4.1.15 Deleting a Task

This topic describes how to delete a task.

Description

To delete a task:

1. Choose the task you want to delete.
2. Click the **Overview** tab.
3. Click the **Delete** link.

Conditions

You can't delete task if:

- You do not have permissions to delete task (**Statuses -> Edit** tab in the **User Management** mode, **Categories->Permissions** tab in the **Task Management** mode).

2.4.2 Managing User Account

2.4.2.1 Establishing a User Group Account

This topic describes how to create a user group.

Description

To create a user group:

1. Click the **User Management -> Statuses** tab.
2. Enter the status **Name** and choose a **Parent Status**.
3. Click the **Save** button.
4. Click the **Statuses -> Edit** tab.
5. Set permissions for users who hold this status.
6. Click the **Save** button.
7. Use the **Categories->Permissions** tab in the **Task Management** mode to specify which categories in which the user can create, delete or edit tasks.
8. Use the **Workflows->Permissions** tab in the **Task Management** mode to specify which types of messages the user can create and view.

See Also

- User Status Concepts (see page 37)
- Security Concepts (see page 37)
- Defining Users Who Can Create, Edit or Delete Task (see page 60)
- Restricting Who Can Close Task (see page 59)
- Restricting Handler List (see page 59)

2.4.2.2 Creating a Department Account

This topic describes how to create a department.

Description

To create a department you need to create an account for the department manager.

See Also

- Creating a User Account (see page 50)

2.4.2.3 Creating a User Account

This topic describes how to create an account.

Description

To create a user account:

1. Click the **User Management -> Overview** tab.
2. Fill in the user's **Login**, **Name** and choose **Status**.
3. Click the **Create a User** button.
4. Fill out the rest of the fields.
5. Click the **Save** button.
6. Click the **User -> Change Password** tab in the **User Management** mode.
7. Type the new password twice.
8. Click the **Save** button.
9. Specify the tasks and projects to which the user will have access by using the **Access Control -> Assigned Statuses** tab in the **Task Management** mode.

See Also

- User Concepts (see page 35)
- Security Concepts (see page 37)

- Defining Users Who Can Create, Edit or Delete Task (see page 60)
- Restricting Who Can Close Task (see page 59)
- Restricting Handler List (see page 59)
- Choosing Localized Interface (see page 60)
- User Properties (see page 72)

2.4.2.4 Creating a Customer Account

This topic describes how to create a customer account.

Description

In order to create a customer account, you should first create a user account.

See Also

- Creating a User Account (see page 50)

2.4.2.5 Editing Account Properties

This topic describes how to edit user account.

Description

To edit user account properties:

1. Click the **User -> Edit** tab in the **User Management** mode.
2. Enter user account properties.
3. Click the **Save** button.

See Also

- User Properties (see page 72)

2.4.2.6 Granting User Access to a Project

This topic describes how to grant user access to a project.

Description

To grant access to a project:

1. Select a project for which you want to give your subordinate user access.
2. Click the **Access Control -> Assigned Statuses** tab.
3. Select the user to whom you are going to assign status for this task.
4. Click the **Save** button.
5. **Optional:** change assigned user status.
6. Click the **Save** button.

See Also

- Security Concepts (see page 37)

2.4.2.7 Customers Creating Accounts for Themselves

This topic describes how to give customers access to create their own accounts.

Description

To allow customers to create accounts for themselves.

1. Click the **User Management -> Registration** tab.
2. Fill in registration rule properties.
3. Click the **Save** button.
4. Click the **Registration -> Edit** tab.
5. **Optional:** Specify additional options for the created registration rule.
6. Click the **Save** button.

Example 1

Suppose we need to organize a hosted service. To do that, new users with *administrator* status should register. They should be able to create 2 sub-users and to use their account for 30 days. They should also have their own project for creating subtasks. In this case the *root* user should create a registration with the following parameters:

Property	Value
Status	*administrator*
Task	*Projects*
Children Allowed	*2*
Expire in Days	*30*
Create a New Project for Each User	*On*
Category	*Folder*

Example 2

Suppose we need to grant customers access to information about bugs in the system which are currently being developed. To do that, the new users should register as a *100 external customer*. They should not be allowed to create sub-users. They should have no time limitations to use their account, and they should have access to the project *Customer Support*. In this case you should create a registration rule like:

Property	Value
Status	*100 external customer*
Task	*Customer Support*
Children Allowed	0
Expire in Days	<empty>
Create a New Project for Each User	Off

See Also

- External User Self-Registration Concepts (see page 41)
- Self-Registration Rule Properties (see page 75)

2.4.2.8 Finding a User Account by Properties

This topic describes how to find a user account by properties.

Description

To find a user account by properties:

1. Click the **Filters -> List** tab in **User Management** mode.
2. Create a new filter or choose an existing one.

3. Click the **Filter -> Edit** tab.
4. Fill in user search conditions.
5. Click the **Save** button.
6. Click the **List of Users** tab.
7. Select the filter.

See Also

- Filter Concepts (see page 39)
- User Properties (see page 72)
- User Filter Properties (see page 80)

2.4.2.9 Changing a Password

This topic describes how to change user password.

Description

To change a password:

1. Click the **User -> Change Password** tab in the **User Management** mode.
2. Type new password twice.
3. Click the **Save** button.

Conditions

- You can't change LDAP user passwords from TrackStudio.

2.4.2.10 Moving a User Account

This topic describes how to move a user account.

Description

To move a user account:

1. Choose your subordinate user account.
2. Click the **User -> Edit** tab in the **User Management** mode.
3. Choose a new manager for the user.
4. Click the **Save** button.

Conditions

- You can't move your own account.

2.4.2.11 Locking a User Account

This topic describes how to lock a user account.

Description

To lock a user account:

1. Choose your subordinate user account.
2. Click the **User -> Edit** tab.
3. Mark the **Active** flag off.
4. Click the **Save** button.

Now the locked user and their subordinated users can't login into the system.

Conditions

- You can't lock your own account.

See Also

- User Properties (see page 72)

2.4.2.12 Deleting a User Account

This topic describes how to delete a user account.

Description

To delete user account:

1. Choose a subordinate user account.
2. Click the **Overview** tab in the **User Management** mode.
3. Click the **Delete** link.

Conditions

- You must have permission to delete a user (**Statuses -> Edit** tab in the **User Management** mode).
- You cannot delete a user who has subordinate users.
- You cannot delete your own account.

2.4.3 Customizing Tasks and Users

2.4.3.1 Hiding Unused Fields

This topic describes how to hide unused fields.

Description

To hide unused fields:

1. Click the **User Management - > Statuses** tab.
2. Select the user status for which you want to hide fields.
3. Click the **Statuses -> Edit** tab.
4. Mark checkboxes off for **./tasks/taskFields/*** and **./users/userFields/***.
5. Click the **Save** button.

Conditions

- You cannot hide fields for the *administrator* status.

Example

To hide **Budget**, **Actual Budget** and **Deadline** fields do the following:

1. Choose the **User Management - >Statuses** tab.
2. Select the user status for which you want to hide fields.
3. Click the **Statuses -> Edit** tab.
4. Mark checkboxes off for the following fields:

```
./tasks/taskFields/viewTaskBudget
```

```
./tasks/taskFields/viewTaskActualBudget
./tasks/taskFields/viewTaskDeadline
```

See Also

- Adding a Custom Field (see page 55)

2.4.3.2 Hiding Tab or Button

This topic describes how to hide tabs or buttons from a user.

Description

To hide a tab or button:

1. Click the **User Management - > Statuses** tab.
2. Select the user status for which you want to hide the tab or button.
3. Click the **Statuses -> Edit** tab.
4. Check checkboxes off for extra tabs or buttons.
5. Click the **Save** button.

Conditions

- You cannot hide tabs or buttons for the *administrator* status.

Example

To hide the **Workflows** and **Categories** tabs:

1. Choose **User Management - >Statuses** tab.
2. Select the user status for which you want to hide fields.
3. Click the **Statuses -> Edit** tab.
4. Mark checkboxes off for the following tabs:

```
./tasks/viewCategoryList
./tasks/viewWorkflowList
```

2.4.3.3 Setting Default Handler for a Project

This topic describes how to set a default handler for a project.

Description

To set the default handler for a project:

1. Choose a project
2. Change handler for the project to the required user.

Now for each new task in the project, TrackStudio will set the handler to the selected user.

See Also

- Assigning a Task (see page 44)

2.4.3.4 Adding a Custom Field

This topic describes how to add a custom field.

Description

- To add a custom field for a task and its subtasks:
  1. Click the **Task -> Customize** tab.

2. Enter the custom field information.
3. Click the **Save** button.

- To add a custom field for all tasks that use the specified workflow:
  1. Click the **Workflows -> Customize** tab.
  2. Enter the custom field information.
  3. Click the **Save** button.
- To add a custom field for the user and subordinated users:
  1. Click the **User -> Customize** tab in the **User Management** mode.
  2. Enter the custom field information.
  3. Click the **Save** button.

See Also

- Custom Field Concepts (see page 39)
- Script Concepts (see page 40)
- Calculating Custom Field Value (see page 56)
- Hiding Unused Fields (see page 54)
- Custom Field Properties (see page 73)

2.4.3.5 Calculating Custom Field Value

This topic describes how to use scripts to calculate a custom field value.

Description

To calculate a custom field value:

1. Click the **User Management -> Scripts**.
2. Create a new script or choose an existing one from the list.
3. Click the **Scripts -> Edit** tab.
4. Choose a task to test the script.
5. Click the **Edit** button.
6. Type the text of the script in the popup window. Left-click on a variable or a constant in the left section of the window to add it to the expression.
7. **Optional:** Click the **Test** button to check the correctness of the expression by calculating it.
8. Click the **Save** button.
9. Add a custom field and fill in the **Script** property for it.

Example 1

To return the day of the week on which an issue was created use the following **task processing** script:

```
gran.tools.Logger log = new gran.tools.Logger("script");
log.debug("start script");
Calendar ca = Calendar.getInstance();
ca.setTime(new Date(task.getSubmitdate().getTime()));
int day = ca.get(Calendar.DAY_OF_WEEK);
switch (day){
     case Calendar.SUNDAY: return "Sunday";
     case Calendar.MONDAY: return "Monday";
     case Calendar.TUESDAY: return "Tuesday";
     case Calendar.WEDNESDAY: return "Wednesday";
     case Calendar.THURSDAY: return "Thursday";
     case Calendar.FRIDAY: return "Friday";
     case Calendar.SATURDAY: return "Saturday";
}
```

```
return null;
```

Associated custom field properties:

Property	Value
Type	List
List of Values	*Sunday* *Monday* *Tuesday* *Wednesday* *Thursday* *Friday* *Saturday*

Example 2

To collect text from all messages for the current task use the following **task processing** script:

```
String s = "";
for(Iterator it = task.getMessages().iterator(); it.hasNext();) {
    String desc = it.next().getDescription();
    if (desc != null) {
      s += desc + " | ";
    }
}
return s;
```

Associated custom field properties:

Property	Value
Type	String

Example 3

To calculate a summary of the actual budget for all not-closed tasks, use the following **task processing** script:

```
public double getAbudget(Object t) {
    double d = 0;
    if(t.getActualBudget() != null && t.getClosedate() == null)
        d = t.getActualBudget().doubleValue();
    for(Iterator it = t.getChildren().iterator(); it.hasNext();)
        d += getAbudget(it.next());
    return d;
}
return getAbudget(task);
```

Associated custom field properties:

Property	Value
Type	Float

Example 4

To list all custom fields with values for the current task, use the following **task processing** script:

```
String s = "";
Map udfs = task.getUDFValues();
for(Iterator it = udfs.keySet().iterator(); it.hasNext();) {
    Object udf = udfs.get(it.next());
    s += udf.getCaption() + ":";
    s += udf.getValue(task) + "|";

}
return s;
```

Associated custom field properties:

Property	Value
Type	String

Example 5

To return the number of days since the last update (this field can be used to find out what tasks have been neglected for a long time), use the following **task processing** script:

```
((new Date()).getTime() - task.getUpdatedate().getTime())/DAYS
```

Associated custom field properties:

Property	Value
Type	Integer

Example 6

To show the link to **Task->Uploads** for tasks with custom fields use the following **task processing** script:

```
if (task.hasAttachments())
   return gran.app.Config.getInstance().getSiteURL()
          +"/jsp/task/viewtask/uploads/Uploads.jsp?TSSESSION="+task.getSecure().getId()
          +"&ID="+task.getId()
          +"\nDownload";
else
   return "";
```

Associated custom field properties:

Property	Value
Type	URL

Example 7

To get a logged user login use the following **task processing** script:

```
ArrayList arr = new ArrayList();
arr.add(task.getSecure().getUser().getLogin());
return arr;
```

or the following **user processing** script:

```
ArrayList arr = new ArrayList();
arr.add(user.getSecure().getUser().getLogin());
return arr;
```

Associated custom field properties:

Property	Value
Type	User

Example 8

To get a task list, use the following script:

```
ArrayList arr = new ArrayList();
arr.add("1");
arr.add("5");
arr.add("10");
return arr;
```

Associated custom field properties:

Property	Value
Type	Task

Example 9

To get multiple values, use the following script:

```
ArrayList arr = new ArrayList();
arr.add("Sunday");
arr.add("Monday");
return arr;
```

Associated custom field properties:

Property	Value
Type	Multiple List
List of Values	*Sunday* *Monday* *Tuesday* *Wednesday* *Thursday* *Friday* *Saturday*

See Also

- Custom Field Concepts (see page 39)
- Script Concepts (see page 40)
- Adding a Custom Field (see page 55)
- Hiding Unused Fields (see page 54)
- Custom Field Properties (see page 73)

2.4.3.6 Restricting Who Can Close Task

This topic describes how to restrict access for closing tasks.

Description

To define which users can close tasks:

1. Click the **Task Management -> Workflows** tab.
2. Choose the workflow for which you want to restrict access for task closure.
3. Click the **Workflows -> Permissions** tab.
4. For each message type that would move tasks to the **final** state, set the **Can Process** permission.
5. Click the **Save** button.

See Also

- Closing a Bug (see page 48)
- Creating a Workflow (see page 65)

2.4.3.7 Restricting Handler List

This topic describes how to restrict the handler list for a message type.

Description

To restrict handler list:

1. Click the **Task Management -> Workflows** tab.

2. Choose the workflow for which you want to restrict the handler list.
3. Click the **Workflows -> Permissions** tab.
4. For each message type, specify the **Can be Handler** permission.
5. Click the **Save** button.

Conditions

- You cannot restrict the handler list from users who submit tasks.

See Also

- Creating a Workflow (see page 65)

2.4.3.8 Defining Users Who Can Create, Edit or Delete Task

This topic describes how to define which users can create, edit or delete tasks.

Description

To define users who can create, edit or delete tasks:

1. Click the **Task Management -> Categories** tab.
2. Choose a category.
3. Click the **Categories -> Permissions** tab.
4. For each user status, specify the statuses of those users you want to provide access to:
   - create tasks
   - edit tasks
   - delete tasks
5. Click the **Save** button.

See Also

- Security Concepts (see page 37)

2.4.3.9 Choosing Localized Interface

This topic describes how to choose the localized version of user interface.

Description

To choose the localized version of user interface:

1. Click the **User -> Edit** tab in the **User Management** mode.
2. Choose **Locale**.
3. Click the **Save** button.

Conditions

- All the necessary translations for the English and Russian locales are already included in TrackStudio.

See Also

- Localizing User Interface (see page 81)

2.4.4 Searching and Analyzing Tasks

2.4.4.1 Finding a Task

This topic describes how to find a task.

Description

To find a task:

- by task number:
  1. Enter the task number into the input field in the right section of the **Main Menu**.
  2. Click the **Go** button.
- by task alias:
  1. Enter the task alias into the input field in the right section of the **Main Menu**.
  2. Click the **Go** button.
- using **Navigation Tree:**
  1. From the **Navigation Tree**, choose the parent project for your task.
  2. Click the **Subtasks** tab.
  3. Choose the task filter.
  4. Choose the required task from the list.

2.4.4.2 Finding Task by Content

This topic describes how to find task by content.

Description

To find a task by content:

1. Click the **Task Management -> Subtasks** tab.
2. Choose a task filter. The full text search is performed only those tasks which pass the filter.
3. Choose the **Full Text Search** filtering parameter.
4. Enter the query string.
5. Click the **Go** button.

See Also

- Full Text Search Reference (see page 76)

2.4.4.3 Filtering Subtasks by Properties

This topic describes how to filter subtasks by properties.

Description

To filter subtasks by properties:

1. Click the **Task Management -> Filters** tab.
2. Choose an existing filter or create a new one.
3. Click the **Filters -> Edit** tab.
4. Fill in the filter conditions.
5. Click the **Save** button.
6. Click the **Subtasks** tab.

7. Switch the current filter.

Example 1

To filter all tasks for which you are handler, use the following filter:

Column	Condition
Handler	*is Current user*

Example 2

To filter all your tasks due today, use the following filter:

Column	Condition
Handler	*is Current user*
Deadline	*1 days after or early*

Example 3

To filter all tasks which were closed within the past week use the following filter:

Column	Condition
Status	*is closed*
Close Date	*7 days before or later*

See Also

- Task Properties (see page 72)

2.4.4.4 Filtering Tasks Using AND/OR/NOT

This topic describes how to filter tasks using AND/OR/NOT.

Description

To filter tasks using AND/OR/NOT:

1. Create a script that will return 1 whenever the task passes the filter conditions.
2. Create a calculated custom field that will return the value of that script.
3. Choose an existing task filter or create a new one.
4. Set filtering conditions for the calculated custom field.
5. Click the **Subtasks** tab.
6. Choose your filter.

See Also

- Adding a Custom Field (see page 55)
- Calculating Custom Field Value (see page 56)

2.4.4.5 Sorting Tasks

This topic describes how to sort task by properties.

Description

- To sort tasks by properties:
  1. Click the **Task Management -> Filters** tab.
  2. Choose an existing filter or create a new one.

3. Click the **Filters -> Edit** tab.
4. Set the order direction and the order level.
5. Click the **Save** button.
6. Click the **Subtasks** tab.
7. Choose your filter.

- To temporarily change the sort order
  1. Click the **Subtasks** tab
  2. Click the column header.

2.4.4.6 Sorting Tasks by Product

This topic describes how to sort tasks by product name.

Description

To sort tasks by product (parent project) name:

1. Click the **Task Management -> Filters** tab.
2. Choose an existing filter or create a new one.
3. Click the **Filters -> Edit** tab.
4. Click the **Deep Search** checkbox.
5. Click the **Full Path** checkbox and set the sorting order for this field.
6. Click the **Save** button.
7. Click the **Subtasks** tab.
8. Choose your filter.

See Also

- Task Filter Properties (see page 78)

2.4.4.7 Analyzing Tasks Distribution

This topic describes how to use distribution report to analyze task distribution.

Description

To analyze task distribution:

1. Click the **Reports -> List** tab.
2. Create a new **Distribution** report.
3. Click the **Reports -> View** tab.
4. Enter the report parameters.
5. Click the **Submit** button.

Example 1

To calculate the total work time required to solve tasks for each submitter and the category of the task use the following report:

Property	**Value**
Type	*Distribution*
Group by	*Category*
Subgroup by	*Submitter*

Value	*Actual Budget*
Function	*Sum*

The system will also calculate the total time expenditure from each category and each submitter as well as the total time spent on all the tasks.

To generate the report, the system finds subtasks of the current task which satisfy the filter condition. After that, the data is grouped based on the **Group by** and the **Subgroup by** properties, and for each group the value of the target function is calculated.

Example 2

To calculate the distribution of the number of tasks based on category and status use the following report:

Property	**Value**
Type	*Distribution*
Group by	*Category*
Subgroup by	*Status*
Value	*Task Amount*
Function	*Sum*

Example 3

To find out how many tasks are created and who created them on specific days of the week:

1. Create the following **task processing** script

```
Calendar ca = Calendar.getInstance();
ca.setTime(new Date(task.getSubmitdate().getTime()));
int day = ca.get(Calendar.DAY_OF_WEEK);
switch (day){
     case Calendar.SUNDAY: return "Sunday";
     case Calendar.MONDAY: return "Monday";
     case Calendar.TUESDAY: return "Tuesday";
     case Calendar.WEDNESDAY: return "Wednesday";
     case Calendar.THURSDAY: return "Thursday";
     case Calendar.FRIDAY: return "Friday";
     case Calendar.SATURDAY: return "Saturday";
}
return null;
```

2. Create the calculated custom field *submitdate'*with the **List** type based on this script. The list of possible element values must correspond to the list of possible results of the formula calculation (*Sunday, Monday*, etc). This calculated field returns the name of the day of the week when the task was created.

3. Create the following report:

Property	**Value**
Type	*Distribution*
Group by	*submitdate*
Subgroup by	*Submitter*
Value	*Task Amount*
Function	*Sum*

See Also

- Script Concepts (see page 40)
- Generating a Report (see page 48)
- Calculating Custom Field Value (see page 56)

2.4.5 Managing Categories and Workflows

2.4.5.1 Creating a Workflow

This topic describes how to create a workflow.

Description

To create a workflow:

1. Click the **Task Management - > Workflows** tab.
2. Enter the name for a new workflow into the input field.
3. Click the **Save** button.
4. Create states in which a task can be:
    1. Click the **Workflows -> States** tab.
    2. Create required task states and mark the start state.
5. Create message types for task transition from one state into another.
    1. Click the **Workflows -> Message Types** tab.
    2. Create required message types and set resolution list for each message type (if required).
6. Specify transitions between task states for each message type:
    1. Click the **Workflows -> Transitions** tab.
    2. For each message type, specify transitions between task states.
7. Specify permissions for each message type:
    1. Click the **Workflows -> Permissions** tab.
    2. For each message type, specify users with which statuses:
        - can create messages.
        - can view added messages (including messages added by other users).
        - can be made handler when a message is added.

See Also

- Workflow Concepts (see page 36)
- Security Concepts (see page 37)

2.4.5.2 Creating a Category

This topic describes how to create a task category.

Description

To create a task category:

1. Click the **Task Management -> Categories** tab.
2. Specify the name of the category.
3. Select the workflow to be used by tasks of this category.
4. Click the **Save** button.
5. Click the **Categories -> Edit** tab. Set possible subcategories for the tasks within the created category.

6. Click the **Categories -> Permissions** tab. For each user status set permissions for the tasks within the created category.
7. Use the **Edit** tab of the parent category (not the category you just created!) to specify your category as a possible subcategory.

Conditions

- You cannot delete a category or specify another workflow for it if there is at least one task in this category.

See Also

- Category Concepts (see page 37)
- Security Concepts (see page 37)
- Defining Category Dependency (see page 66)

2.4.5.3 Defining Category Dependency

This topic describes how to define category dependency.

Description

To define category dependency:

1. Click the **Task Management -> Categories** tab.
2. Select a category from the list.
3. Click the **Categories -> Edit** tab.
4. Edit the possible subcategories list.

See Also

- Category Concepts (see page 37)

2.4.5.4 Setting Workflow for Task Category

This topic describes how to set workflow for a task category.

Description

To set workflow for task category:

1. Click the **Task Management -> Categories** tab.
2. Choose the workflow for your category
3. Click the **Save** button.

Conditions

- You cannot specify another workflow for a category if at least one task exists in this category.

See Also

- Category Properties (see page 74)
- Workflow Concepts (see page 36)

2.4.6 Managing E-Mail Notification and Submission

2.4.6.1 Receiving E-Mail Notification when Tasks Change

This topic describes how to create an e-mail notification rule.

Description

To create an e-mail notification rule:

1. Choose a project or bug for which you want to enable e-mail notifications.
2. Click the **Task Management -> Filters** tab.
3. Choose a filter that will return tasks or messages you want the user to receive via email notification.
4. Click the **Filters -> Notifications** tab.
5. Choose the e-mail notification recipient.
6. Click the **Save** button.
7. Choose the **E-Mail Template.**
8. Click the **Save** button.

Notes

Enabling notification to the *All* filter for a certain bug or task is similar to using the "watch" mode in some systems -- i.e. a user will get email notifications whenever there is a change in the task status or any messages are added.

Example 1

To get e-mail notifications both when a new task is added and when some other user (not him/herself) adds a message (bug-note), use the following filter:

Section	Column	Condition
Message Settings	**Submitter**	*is not Current user*

Example 2

To get e-mail notifications only if the handler of the task, use the filter:

Section	Column	Condition
Task Settings	**Handler**	*is Current user*

Example 3

To get e-mail notifications both when a high-priority task is added and when the John Smith adds some messages to the task:

Section	Column	Condition
Task Settings	**Priority**	*High*
Message Settings	**Submitter**	*Customer*

Example 4

To get e-mail notifications when other users add a message to the task for which the handler is the subscribed user:

Section	Column	Condition
Task Settings	**Handler**	*is Current user*
Message Settings	**Submitter**	*is not Current user*

Example 5

Use the e-mail notification rule based on the following script to send e-mail notification only when a handler has been changed. This script will return 1 when you change a handler adding a task or message -- 0 otherwise.

```
int ret = 0;

Collection messages = task.getMessages();
Collections.reverse(messages);
```

```
String oldHandler = task.getParent() != null ? task.getParent().getHandlerId() : "null";

if (messages.isEmpty() && !oldHandler.equals(task.getHandlerId()))
     ret = 1;

for(Iterator it = messages.iterator(); it.hasNext();) {
    String newHandler = it.next().getHandlerId();
    if (newHandler == null) newHandler = "null";
    if (!newHandler.equals(oldHandler)) {
       ret = 1;
    } else {
       ret = 0;
    }
    oldHandler = newHandler;
}
return ret;
```

Example 6

Use the e-mail notification rule based on the following script to send an e-mail notification only when the user adds a message and not when the user modifies task properties. The following script will return 1 when the task has been modified by adding a message, or 0 when the task has been modified using **Task->Edit**.

```
int ret = 0;

Collection messages = task.getMessages();
java.sql.Timestamp taskDate = task.getUpdatedate();

for(Iterator it = messages.iterator(); it.hasNext();) {
    java.sql.Timestamp date = it.next().getTime();
    if (date.equals(taskDate)) {
       ret=1;
       break;
    }
}
return ret;
```

Example 7

To get the time difference (in seconds) between two consecutive messages:

```
int ret = 0;

Collection messages = task.getMessages();
Collections.reverse(messages);

java.sql.Timestamp oldDate =
     task.getSubmitdate();
long diff = 0;

for(Iterator it = messages.iterator();
    it.hasNext();) {

   java.sql.Timestamp newDate =
      it.next().getTime();
   diff = newDate.getTime() - oldDate.getTime();
   oldDate = newDate;

}
return diff/1000;
```

See Also

- Configuring E-Mail Notification (see page 20)
- E-Mail Notification Concepts (see page 41)
- E-Mail Notification Rule Properties (see page 80)

2.4.6.2 Periodically Receiving Lists of Tasks by E-Mail

This topic describes how to create a filter subscription rule.

Description

To periodically receive a list of tasks by e-mail, create a filter subscription rule:

1. Choose the project or bug for which you want to enable filter subscription.
2. Click the **Task Management -> Filters** tab.
3. Choose the filter that will return tasks or messages for the filter subscription.
4. Click the **Filters -> Subscribe** tab.
5. Choose the e-mail notification recipient
6. Click the **Save** button.
7. Fill-in subscription properties.
8. Click the **Save** button.

See Also

- Configuring E-Mail Notification (see page 20)
- E-Mail Notification Concepts (see page 41)
- Filter Subscription Properties (see page 75)

2.4.6.3 Sending Alert for Overdue Task

This topic describes how to periodically receive a list of overdue tasks.

Description

To periodically receive a list of overdue tasks, you should:

1. Create a filter that will return a list of overdue tasks.
2. Create subscription rule for the filter.

See Also

- E-Mail Notification Concepts (see page 41)
- Filtering Subtasks by Properties (see page 61)
- Periodically Receiving Lists of Tasks by E-Mail (see page 69)

2.4.6.4 Adding a Task by E-Mail

This topic describes how to add a task by e-mail.

Before You Begin

To create an e-mail import rule:

1. Choose the parent project for your tasks.
2. Click the **Task -> E-Mail Import** tab in the **Task Management** mode.
3. Mark the **Enable E-Mail Import** checkbox.
4. Fill-in the e-mail import rule properties.
5. Click the **Save** button.

Description

1. Use your e-mail client to create a new e-mail message.
2. Fill-in task name in the e-mail subject and task description in the e-mail body.
3. Fill-in keyword somewhere in the e-mail subject or body.
4. Set e-mail recipient to TrackStudio e-mail address (**mail.store.user** property in **trackstudio.mail.properties**).

Conditions

- The user should be registered both in TrackStudio and their e-mail client (Outlook, Outlook Express, etc) with the same name or e-mail.
- The user who sends e-mail submission messages should have permission to add tasks to the specified project.
- E-mail submission should be enabled (**trackstudio.FormMailNotification** in **trackstuido.mail.properties**)

See Also

- Configuring E-Mail Submission (see page 21)
- E-Mail Submission Concepts (see page 42)
- Task Submission Rule Properties (see page 75)

2.4.6.5 Adding a Message by E-Mail

This topic describes how to add a message by e-mail.

Description

To add a message by e-mail

- using html form in the e-mail notification messages
  1. Create a notification rule for the project.
  2. Choose the **HTML** e-mail template for this notification rule.
  3. Use web form in the bottom of e-mail notification e-mail to enter the message information.
  4. Click the **Create message** button.
  5. Enter message description and attach files.
  6. Send e-mail to TrackStudio.
- using plain text e-mail
  1. Create a new message in your e-mail client.
  2. Enter a task number (like *#23*) into the e-mail subject
  3. Enter a message description into the e-mail body.
  4. Send an e-mail notification to the e-mail defined in TrackStudio e-mail settings (**mail.store.user property**).

Conditions

- The user with same name or e-mail should be registered both in TrackStudio and e-mail client properties
- User who sends e-mail submission messages should have permission to add messages to the specified task.
- E-mail submissions should be enabled (**trackstudio.FormMailNotification** in **trackstuido.mail.properties**)

See Also

- Configuring E-Mail Submission (see page 21)
- E-Mail Submission Concepts (see page 42)

2.4.6.6 Customizing E-Mail Notification Templates

This topic describes how to customize e-mail notification template.

Description

To customize e-mail notification template:

1. Click the **User Management -> E-Mail Templates** tab.
2. Choose an existing e-mail template or create a new one.
3. Click **E-Mail Templates -> Edit** tab.
4. Fill in the template properties.
5. Click the **Save** button.

Example

To receive text e-mail notifications like

```
Dear [customer name],
Your task has been updated.
It now has the current [Task Status].
[Message text for only the message that fired this notification]

You may find more additional information on your project by accessing the following link
[TrackStudio URL link to task/message tab]

Thanks,
[Handler Name]
```

use the following template

```
<#if (task.getSubmitter()?exists)>
Dear ${task.getSubmitter().getName()},
</#if>
Your task has been updated.
It now has the current ${task.getStatus().getName()}.

<#if (msglist?exists && msglist?size>0)>
<#assign msg = msglist?last>
<#if msg.getDescription()?exists>${msg.getTextDescription()}</#if>
</#if>

You may find more additional information on your project by accessing the following link
${addval.tasklink}

Thanks,
<#if (task.getHandler()?exists)>
${task.getHandler().getName()}
</#if>
```

See Also

- E-Mail Template Concepts (see page 41)
- E-Mail Notification Rule Properties (see page 80)

2.5 Reference

2.5.1 Task Properties

Following table provides list of task properties

Task Property	Description	Examples
Number	Unique numeric task number.	*#23*
Full Path	Shows the full task path in the tasks hierarchy.	*Projects / Sample, Inc*
Name	Use this field to store a bug summary or project name. The size is limited to 200 characters.	*TrackStudio Enterprise*
Alias	Use Alias to assign a short name to a project. Use the task alias instead of its number to jump to a task quickly.	*TS*
Category	The type of a task is defined by its category. Possible task states and transitions depend on its category.	*Folder, Product Version, Software Bug*
State	Task state.	*New, Resolved, Closed*
Resolution	Task resolution.	*fixed, duplicate*
Priority	Task priority.	*low, normal, high*
Submitter	The user who has submitted this task.	*John Smith*
Handler	The user who has assigned to this task.	*John Smith*
Submit Date	Date/time when the task was submitted.	*01/01/2005*
Update Date	Date/time of the last task update. The last update date for a project is the maximum of the last update of any of its subtasks.	*2005-01-01*
Close Date	Date/time when the task was closed (if the task is closed).	*2005/01/01*
Deadline	This is the date when the current task must be finished.	*2005/01/01*
Budget	Estimated time (in hours) allocated for the task. Each task has one Task Budget common for all users. This is working time, not calendar time.	*5 hh 30 mm 10 ss*
Actual Budget	Elapsed task processing time (working time, not calendar time). The actual budget for the project is the sum of actual budgets of current task and all its subtasks.	*3 hh 10 mm 05 ss*
Description	Enter the task description, if its name is not informative enough.	

2.5.2 User Properties

Following table provides list of user properties:

Property	Description	Examples
Login	The user's login.	*jsmith*
Name	The user's name.	*John Smith*
Full Path	The position of the user in the hierarchy.	*Admin / John Smith / Sean Law / Jeff Franke*
Company	The user's company or department name.	*Sample, Inc.*

Status	The user own status (a user group).	*administrator*
E-Mail	The user's e-mail.	*jsmith@sample.com*
Phone No	The user's phone number.	*(012) 456-789*
Locale	The locale of the user. All dates, as well as figures with floating points must fit the specified locale.	*English (United States)*
Time Zone	The time zone of the user. All the date/time information is displayed according to the specified Time Zone.	*America/New_York*
Expire Date	The date the user's login will expire. The user and his/her subordinate users will not be able to log into the system after the expiration date.	*01/01/2006*
Licensed Users	Contains the number of direct and indirect child users available for the user.	*20*
Default Project	The project that will be selected right after the user logs in. Ensure that specified task is accessible for user.	*#2*
Default E-Mail Template	The default e-mail template for the user.	*HTML*
Active	Clear this flag to mark a discharged user. Inactive users cannot login and you cannot assign them to the tasks.	

2.5.3 Message Properties

Following table provides list of message properties:

Property	Description	Examples
Submitter	The user who has submitted this message.	*John Smith*
Submit Date	Date/time when the message was submitted.	*01/01/2005*
Message Type	The action performed within a task.	*resolve, close*
Handler	User who should now process this task.	*John Smith*
Resolution	New task resolution.	*fixed, not a bug*
Priority	New task priority.	*low, normal, high*
Deadline	This is the date when the current task must be finished.	*01/01/2005*
Budget	Estimated time (in hours) allocated for the task. This is working time, not calendar time.	*20 hh 30 mm*
Actual Budget	Time spent on processing the task.	*10 hh 20 mm*
Description	Message comment.	

2.5.4 Custom Field Properties

Following table provides list of custom field properties:

Property	Description	Examples
Caption	Custom field name. It can be *software platform, release, version, customer name* and so on.	*platform* *release* *version* *customer name*
Order	An integer value which indicates the position of the field on the user page, the system sorts custom fields by this field.	*10* *20* *30*
Type	Custom field type. Please check out next table for more information about available custom field types.	*String* *Memo*
Default	Default field value.	*100*
Script	The script for the calculated custom field. A field is considered calculated if there is a script specified for it.	*getCustomerAddress*
List of Values	Contains possible values for **List** and **Multiple List**. To fill this property enter first list item, then press the **Save** button, then enter the second and save that, etc.	*Red* *Green* *Blue*
Required	Indicates whether a field value is required or not. The users can't save a task if they do not fill in all required properties.	*Yes* *No*

Custom field types:

Type	Description	Examples
String	String of symbols	*XWare 1.0*
Memo	Text area	*Long text*
Float	Floating point value	*2.3*
Integer	Integer value	*5*
Date	Date/Time value	*01/01/2005 15:00*
List	Drop-down list with values specified in the **List** field	*["London",* *"Paris",* *"Moscow"]*
Multiple List	Multi-select list with values specified in the **List** field	*["London",* *"Paris",* *"Moscow"]*
Task	Task link	*["#1","#5"]*
User	User link	*["jsmith","clist"]*
URL	Uniform Resource Locator	TrackStudio

2.5.5 Category Properties

Following table provides list of category properties:

Column	Description
Category	The name of the category.
Workflow	Workflow associated with this category.

Connected to	The task that has this category assigned to it. You need to have access rights for this task to edit or delete this category.

2.5.6 Self-Registration Rule Properties

Following table provides list of self-registration rule properties:

Property	Description
Child Allowed	The maximum number of sub-users that the new user can create.
Expire in days	Number of days after which the created account expires. If the parameter is not specified there will be no time limitations for the new user to use his/her account. In that case the account will expire when/if one of the parent users' accounts expires.
Create a New Project for Each User	Check this option if your tasks for different self-registered users should be independent. In this case a new parent project is created and the new user is granted access rights to it.
Category	If **Create a New Project for Each User** is selected, you can define **Category** that specifies the category of the new task.

2.5.7 Filter Subscription Properties

Following table provides list of filter subscription rule properties:

Property	Description
Connected to	You will receive email notification on this task and the subtasks of this task.
User	Subscriber. You can subscribe yourself or your subordinate users.
E-Mail Template	The template that will be used for e-mail notifications. A user can receive several e-mail notification messages that differ only in their templates.
Valid Time	You can specify start (**From**) and stop (**To**) date of subscription. You will receive e-mails only when the current date is between the two.
Next Run	Then next time when the filter should be executed. Generally, you should not modify this field.
Interval	You should select **mailing interval** - from 30 minutes to 1 month.

2.5.8 Task Submission Rule Properties

Following table provides list of task submission rule properties:

Property	Description
Enable E-Mail Import	Enables task submission for the current task.

Contains Keyword	Allows you to specify the **keyword** either in the subject or in the body of the message. Keywords make it possible to use one mailbox for importing messages into several projects, therefore you should use different keywords for different projects. If no keyword is specified, the e-mail is imported regardless of its body or subject. The e-mail import rules with the empty keyword list are checked after the rules with a keyword specified. The keyword check is case-insensitive.
In	Allows you to specify the field in which to search for the keyword. The **Subject** and **Body** fields are supported.
Category	The category of the created tasks. The category must be associated with a workflow which has the start state.

2.5.9 Full Text Search Reference

TrackStudio uses Lucene for text indexing, which provides a rich query language that can make constructing full text queries daunting. This document is derived from the Lucene document on Query Parser Syntax.

Description

A query is broken up into terms and operators. There are two types of terms: Single Terms and Phrases. A Single Term is a single word such as "test" or "hello". A Phrase is a group of words surrounded by double quotes such as "hello dolly". Multiple terms can be combined together with Boolean operators to form a more complex query (see below). All query terms are case insensitive.

TrackStudio supports modifying query terms to provide a wide range of searching options.

Wildcard Searches

TrackStudio supports single and multiple character wildcard searches. To perform a single character wildcard search use the "?" symbol. To perform a multiple character wildcard search use the "*" symbol. You cannot use a * or ? symbol as the first character of a search. The single character wildcard search looks for terms that match that with the single character replaced. For example, to search for "text" or "test" you can use the search:

```
te?t
```

Multiple character wildcard searches looks for 0 or more characters. For example, to search for Windows, Win95 or WindowsNT you can use the search:

```
win*
```

You can also use the wildcard searches in the middle of a term. For example, to search for Win95 or Windows95 you can use the search

```
wi*95
```

Fuzzy Searches

TrackStudio supports fuzzy searches. To do a fuzzy search use the tilde, "~", symbol at the end of a Single word Term. For example to search for a term similar in spelling to "roam" use the fuzzy search:

```
roam~
```

This search will find terms like foam and roams

Boolean Operators

Boolean operators allow terms to be combined through logic operators. TrackStudio supports AND, "+", OR, NOT and "-" as Boolean operators . Boolean operators must be ALL CAPS.

OR

The OR operator is the default conjunction operator. This means that if there is no Boolean operator between two terms, the OR operator is used. The OR operator links two terms and finds a matching document if either of the terms exist in a document. This is equivalent to a union using sets. The symbol || can be used in place of the word OR.

To search for documents that contain either "software TrackStudio" or just "TrackStudio" use the query:

```
software || TrackStudio
```

or

```
software OR TrackStudio
```

AND

The AND operator matches documents where both terms exist anywhere in the text of a single document. This is equivalent to an intersection using sets. The symbol && can be used in place of the word AND.

To search for documents that contain "software TrackStudio" and "issue tracking" use the query:

```
TrackStudio AND tracking
```

Required term: +

The "+" or required operator requires that the term after the "+" symbol exist somewhere in a the field of a single document.

To search for documents that must contain "TrackStudio" and may contain "software" use the query:

```
+TrackStudio software
```

NOT

The NOT operator excludes documents that contain the term after NOT. This is equivalent to a difference using sets. The symbol ! can be used in place of the word NOT.

To search for documents that contain "software TrackStudio" but not "japan" use the query:

```
TrackStudio NOT japan
```

Note: The NOT operator cannot be used with just one term. For example, the following search will return no results:

```
NOT TrackStudio
```

Excluded term: -

The "-" or prohibit operator excludes documents that contain the term after the "-" symbol.

To search for documents that contain "software TrackStudio" but not "japan" use the query:

```
TrackStudio -japan
```

Grouping

TrackStudio supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

To search for either "software" or "TrackStudio" and "bugs" use the query:

```
(software OR TrackStudio) AND bugs
```

This eliminates any confusion and makes sure you that bugs must exist and either term software or TrackStudio may exist.

Escaping Special Characters

TrackStudio supports escaping special characters that are part of the query syntax. The current list special characters are

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

To escape these character use the \ before the character. For example to search for (1+1):2 use the query:

```
\(1\+1\)\:2
```

2.5.10 Assigned Status Property

Following table provides assigned status properties:

Property	Description
User	The user who has this status assigned to him.
Assigned Status	User status for the task.
Override	Mark to override inherited statuses (both user own status and assigned statuses for the parent tasks).
Connected to	The task that has this status assigned to it.
Owner	The user who assigned the status.

2.5.11 Task Filter Properties

Filter properties:

Property	Description	Example
Name	Filter name.	*Roadmap*
Description	Filter description.	*Tasks roadmap*
Type	Filter type: • **normal** filters can be used anywhere • **notification** filters can be used only for e-mail notification and filter subscription • **report** filters can be used only for report generation	*normal*
Private	Private filters visible only for owner, non-private filters visible to all users that have access rights to this task or their subtasks. If you have no access rights for some task, you can create only private filters for it.	*On*
Connected to	Parent task for filter. Filter will be available for this task and its subtasks.	*Sample, Inc*
Owner	The user who created the filter. You can't modify or delete foreign filters, but you can use them for task filtering and email notification.	*John Smith*

General filter settings:

Setting	Description	Example
Tasks per Page	Amount of tasks per page.	*20*
Deep Search	Use the **Deep search** checkbox to recursive filter tasks through the entire hierarchy beginning with the current task.	*Off*
Full Text Search	The **Full Text Search** field allows you to search for specific words and phrases within a particular task. The full text search is performed only through the tasks that pass the filter.	*`(software OR TrackStudio) AND bugs`*

Task filter settings:

Setting	Description	Example
Filter	Mark this checkbox to filter by task field	*On*
Hide	Mark this checkbox to hide value from subtasks list	*Off*
Column	Task field name	*Update Date*
Condition	Filtering condition	*is 7 days before or later*
Sort	Sort order and level. The list of tasks can be sorted either by one field or by several fields. For example, you can sort the list of tasks by categories and then sort those within the same category by the date they were created. The order of sorting is specified in the **level** field.	*Category, Asc, 1,* *Submit Date, Desc, 2*

Message filter settings:

Setting	Description	Example
Filter	Mark this checkbox to filter by task field	*On*
Message Parameter	Message field name	*Submitter*
Condition	Filtering condition	*is John Smith*

Additional message filter settings:

Setting	Description	Example
View Messages	Set this checkbox if you want to see the list of messages for the task. You can specify the number of messages to be displayed in the list. You can choose the first few messages or the last few messages. Messages in all lists are sorted by the date of their creation in ascending order.	First 5
Filter Messages	Use to filter messages in tasks. If it is off, only tasks will be filtered.	Last 1
Bulk Processing Tool	Use the Bulk Processing Tool to add messages to a number of tasks at once. Behavior of the message editor in the **Messages** window differs from that in the **Subtasks** window. To add a message using the **Bulk Processing Tool** you should specify non-empty **Message Description**.	On

Remarks

The **Filter Messages** must be on if you wish to filter messages in tasks. If it is off, only tasks will be filtered. For example, if the following condition is specified – *Message Handler=John and Message Filter=on*, all the tasks will be found, but only the messages that meet *Message Handler=John* will be displayed.

If the following condition is specified – *Message Handler=John and Message Filter=off*, all the tasks that have at least one message where *handler=John* will be displayed (if **view messages=on**, all messages in these tasks will be displayed no matter who their handler is).

You can specify from the beginning what number of messages should be filtered. Suppose you specified that the last 20 messages should be filtered. TrackStudio would filter the last 20 messages from the list according to the filtering conditions. The rest of the messages will not be processed or displayed.

View Messages and **Filter Messages** are checked in the following order:

1) The condition **Filter Messages** is applied first; it leaves the specified number of the first and the last messages in each task.

2) Then filtering by **submitter**, **handler** and other fields is performed.

3) And then **view messages** displays the specified number of messages at the beginning or at the end of the list.

For example, if you specified:

```
Filter Messages: 5 last
Submitter: John
```

```
View Messages: 3 first
```

TrackStudio will find the last 5 messages for each task first, then it will keep the tasks where *submitter=John*, and if there are more than 3 such messages, it will display only the first 3 of them.

2.5.12 User Filter Properties

Filter properties:

Property	Description	Example
Name	Filter name.	*All*
Description	Filter description.	*List of all users*
Private	Private filters visible only for owner, non-private filters visible to all users that have access rights.	*On*
Owner	The user who created the filter. You can't modify or delete foreign filters, but you can use them for user filtering.	*John Smith*

General filter settings:

Setting	Description	Example
Users per Page	Amount of users per page.	*20*
Deep Search	Use the **Deep search** checkbox to filter tasks through the entire hierarchy beginning with the current user.	*Off*

User filter settings:

Setting	Description	Example
Filter	Mark this checkbox to filter by user field.	*On*
Hide	Mark this checkbox to hide value from the list.	*Off*
Column	User field name.	*Company*
Condition	Filtering condition.	*equals TrackStudio, Ltd*
Sort	Sort order and level.	*Asc, 1*

2.5.13 E-Mail Notification Rule Properties

Property	Description
<current filter>	Determines for which events e-mail notifications should be sent. For example, a rule can be defined so that a notification will be sent only if the subscriber is the handler of the task (*task handler=”current user”*).
Connected to	Email notification is activated for this task. If a user subscribed to email notifications for a project, it would automatically be activated for all subtask of this project. The rule would be activated both for the existing tasks and for new tasks .
User	The recipient of email notifications.
E-Mail Template	E-mail notification template used to notify specified **User** when event defined in the current filter occurs in the **Task** or their subtask.

3 Developer's Guide

3.1 Localizing User Interface

This section describes how to localize your TrackStudio interface.

Description

Internationalization is the process of changing the format of dates and numbers to the one used in your region and translating user-interface text to your language. For example, the 12th of January 1990 will look like 01/12/90 in the American date format and 12.01.1990 in the German one. The date format in TrackStudio is used when displaying and inputting information. Similarly, some user-interface text like *Your Name* in English can be translated to *Ihr Name* for German locales.

User locale is specified separately for each user in the user settings (**User Management->User->Edit**). The list of available locales likely contains the locale that you need, but there may be no language files in TrackStudio for some locales. If you select a locale for which there are no language files, the date format will change to the selected language, but the interface will remain English.

1. Set character encoding

Character Encoding is specified for the entire TrackStudio instance, use Server Manager to specify the character encoding.

```
trackstudio.encoding UTF-8
```

2. Translate the strings

The entire TrackStudio text is stored in resource bundles. A resource bundle is a file containing key/value pairs. TrackStudio loads its text using keys while the correct values are retrieved based on the user’s locale settings.

For example, the key/value pairs for the English locale may look like this:

```
MSG_CHOOSE_PROJECT=Please, choose a project.
MSG_CHOOSE_STATUS=Please choose a status.
MSG_COLUMN=Column
MSG_CONDITION=Condition
```

Making your own translation involves copying the English resource bundle, renaming the file, and translating its contents. To do that, find the file **language_en.properties** in the directory **TrackStudio/webapps/TrackStudio/WEB-INF/classes** and copy it to a new file. The last two letter in the name of the new file must be a valid ISO Language Code.

These codes are the lower-case two-letter codes as defined by ISO-639. You can find a full list of these codes at a number of sites, such as: http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

When translating from English, you may need to use special characters for you language. Unfortunately, all resource bundle files must be saved in ASCII format which doesn't allow for many international characters. It's recommended you work on your translation in a text editor that supports all characters in your language. After finishing your translation, use the **native2ascii** tool to convert international characters to the ASCII format. Here's how you use the **native2ascii** tool:

```
native2ascii -encoding XXX
             my_translation.properties language_YY.properties
```

where *my_translation.properties* is the input file that use national character encoding and *language_YY.properties* is the output file.The **-encoding XXX** parameter is optional. If you don't specify it, Java will use the default encoding value, taken from the system property **file.encoding**. After you translate the file, save it to the directory **WEB-INF/classes**.

Instead of modifying the file **language_YY.properties**, you can create **language_YY1.properties** (for example, *language_en1.properties*) and redefine only the strings you need in it. In its turn, *language_YY1.properties* can be extended and redefined with the help of *language_YY2.properties* and so on. If some

string you need is not defined in the national locale, TrackStudio will take it from the English locale (en, en1,...). If it is not defined there either, the identifier of the string will be used instead of it.

3.2 Extending TrackStudio Functionality

This topic describes TrackStudio Open API.

Description

Open API is used for modifying the present functionality and for enhancing TrackStudio as well as for integrating it with other applications. Access to system kernel is based on adapters with well-defined interfaces. Adapters can be arranged in a pipeline when the result of some method in one adapter is passed over to the next adapter in the pipeline. The application functionality can be modified or enhanced without any changes in the initial code of TrackStudio. Pipeline architecture gives a simple solution to such tasks as audit, additional security checks, method parameter logging and return value modifying.

Implementing adapters, you can realize the following functionality:

- Additional methods for user authentication (external database)
- Data export to various formats.
- Various recurrent operations, for example, the recurring export of data from TrackStudio into other systems.

3.2.1 Implementing Adapters

This topic contains adapter development overview.

Description

An adapter is a class realizing the `gran.app.adapter.Adapter` interface. Each subsystem in TrackStudio has its own interface, inherited from `gran.app.adapter.Adapter`. For instance, to implement an export adapter, you must implement the class, realizing the `gran.app.adapter.ExportAdapter` interface. Adapters are stateless components, i.e. they do not have an internal state and do not remember the history of the previous calls.

The list of the loadable adapters can be found in **trackstudio.adapter.properties**, for example:

```
# External adapters

adapter.export gran.app.adapter.export.XMLExportAdapter
adapter.email gran.app.adapter.email.BaseFilterNotifyAdapter
```

If you need to execute a pipeline of some adapters (implementing the same interface) to perform some operation, you must put them in one line using ';' as a separator, for example:

```
adapter.pop3 gran.app.adapter.pop3.BasePOP3Adapter;
             gran.app.adapter.pop3.MailImportMessagePOP3Adapter;
             gran.app.adapter.pop3.MailImportTaskPOP3Adapter;
             gran.app.adapter.pop3.CleanPOP3Adapter;
             gran.app.adapter.pop3.PostProcessingPOP3Adapter
```

If there are two identical adapters in the list, only the first of them will be executed. If there is an adapter not realizing the required interface in the list, it is not loaded.

TrackStudio takes each type of adapter as consisting of three components: the adapter interface, the proper adapter and AdapterManager realizing the pipeline.

Let's take the realization and interaction of the system components in the work of adapters:

1) Interface. The adapter interface must extend `Adapter`. The following requirements are demanded of the method signature:

- The name of the interface must be `SomethingAdapter`
- Every method must either throw `GranException` or not throw an exception.
- The names of methods must end with `Impl`
- If a method returns a value, this method must have a parameter of a returning type named result, which must come last in the list of parameters. For example,

```
  public boolean authorizeImpl(User user,
                               String password,
                               boolean result)
                throws GranException;
```

- Persistent objects (with a rare exception) are passed either by their string identifier or in the collections `java.util.Collection`, `java.util.LinkedList`, etc. To continue working in the case of the ID passing of the object, you must open a Hibernate session and load the object. When using the object list you do not have to open a session (remember that in this case each object in the collection must be Hibernate-initialized). If executing an adapter results in changes in the persistent object, you must always open and close the session. It is also recommended to use transactions in this case.

Interface example:

```
// $Id: OpenAPI.dtx,v 1.14 2004/05/31 13:49:10 maximkr Exp $
package gran.app.adapter.template;

import gran.app.adapter.Adapter;
import gran.exception.GranException;

public interface TemplateAdapter extends Adapter
{
    public String methodThatReturnSomethingImpl(String param,
                                                String result)
                  throws GranException;

    public void methodThatReturnNothingImpl(String param)
                throws GranException;
}
```

2) An adapter has the following structure:

```
// $Id: OpenAPI.dtx,v 1.14 2004/05/31 13:49:10 maximkr Exp $
package gran.app.adapter.template;

import gran.exception.GranException;
import gran.tools.Logger;

public class BaseTemplateAdapter implements TemplateAdapter
{
    private static Logger log = new
            Logger("gran.app.adapter.template.TemplateAdapter");

    public boolean init()
    {
        return true;
    }

    public String getDescription()
    {
        return "Base Template Adapter";
    }

    public String methodThatReturnSomethingImpl(String param,
                                                String result)
                  throws GranException
    {
        return param + " OK";
    }

    public void methodThatReturnNothingImpl(String param)
```

```
                throws GranException
    {
        return;
    }
}
```

Within an adapter, methods can be called only through `AdapterManager.` Direct calling *Impl-methods is not recommended as it may cause problems when enhancing the system.

3) `SomeAdapterManager` controls the lists of adapters supporting the defined interface and is responsible for the correct passing of the parameters. The realization of `SomeAdapterManager` may have the following structure:

```
// $Id: OpenAPI.dtx,v 1.14 2004/05/31 13:49:10 maximkr Exp $
package gran.app.adapter.template;

import java.util.Collection;
import java.util.Iterator;
import gran.exception.GranException;

public class TemplateAdapterManager
{
    private Collection am = null;

    public TemplateAdapterManager(Collection adapters)
    {
        am = adapters;
    }


    public String methodThatReturnSomething(String param)
                  throws GranException
    {
        String result = null;
        for (Iterator iter = am.iterator(); iter.hasNext();) {
            TemplateAdapter adp = (TemplateAdapter) iter.next();
            result = adp.methodThatReturnSomethingImpl(param, result);
        }
        return result;
    }

    public void methodThatReturnNothing(String param)
                throws GranException
    {
        for (Iterator iter = am.iterator(); iter.hasNext();) {
            TemplateAdapter adp = (TemplateAdapter) iter.next();
            adp.methodThatReturnNothingImpl(param);
        }
    }
}
```

The system enhancement is carried out through classes realizing the existing interfaces (for example, `gran.app.adapter.ExportAdapter`). At the same time you do not have to modify the initial system code, the adapter interface or AdapterManager.

To call the adapters, a singleton class `AdapterManager` is used. This class stores the list of all adapters available on the system and allows registering new adapters in the system. To call a method (e.g. for exporting), you must execute the following:

```
AdapterManager.getInstance().getExportAdapterManager()
                            .export(taskid, userid);
```

3.3 Building TrackStudio from Source

This topic describes building TrackStudio from the source code.

Description

If you have purchased TrackStudio Enterprise with the source code, you can build the application from it. To do this you will need:

- ant 1.6.2
- JDK 1.4.2 or higher.
- **Optional:** Install4j 3.1.x

To build the Standalone distribution for Windows, execute the following command:

```
> ant sa
> ant "-Dinstall4j.lib=C:\Program Files\install4j\bin\install4j.jar" sa-win
```

To build the Standalone distribution for Linux, execute the following command:

```
> ant sa
> ant "-Dinstall4j.lib=C:\Program Files\install4j\bin\install4j.jar" sa-linux
```

To build the WAR distribution, execute the following command:

```
> ant wardist
```

To speed up the process of building, you can use jikes, for example:

```
ant -Dbuild.compiler=jikes sa
```

If you experience any problems in process in building the application, please, contact us.

See Also

- Install4j

3.4 Integrating with Third-Party Systems

This section describes the interaction with TrackStudio Enterprise via SOAP API.

Description

To interact with external applications, import or export data, you can use TrackStudio SOAP API.

TrackStudio SOAP API features:

- Based on the Apache AXIS technology.
- Can be used by Java clients as well as .NET clients.
- Provides direct access to external adapters used for web interface realization. Unlike most other systems, TrackStudio SOAP API does not limit the possibility of interaction with TrackStudio by simple operations.
- It is safe to use TrackStudio SOAP API calls. If the user cannot perform the operation via web interface, he/she will not be able to perform it via SOAP API. SOAP API can be disabled to provide more safety.

The following services are available to work via SOAP API:

- JavaAttachment (Java-only)
- Acl
- Attachment
- Category
- EmailType
- Filter
- Find
- MailImport

- Message
- Prstatus
- Registration
- Script
- Step
- Task
- Udf
- User
- Workflow

This service functionality corresponds to the gran.app.adapter.external adapters. For example, the Acl service allows you to call the gran.app.adapter.external.SecuredAclAdapter adapter methods via SOAP. In order to provide compatibility with various SOAP implementations, TrackStudio uses proxy to convert parameter types. For example, Java collections are not supported in .NET so Java collection is converted to the arrays. The conversion is done in the com.trackstudio.soap.service classes. For example, the proxy for the getEmailTypeList method is the following:

```
    public EmailTypeBean[] getEmailTypeList(String sessionId)
    throws GranException {
        ArrayList list = new ArrayList();
        for (Iterator it =
             manager.getEmailTypeList(sessionId).iterator();
             it.hasNext();)
            list.add(((SecuredEmailTypeBean) it.next()).getSOAP());
        return (EmailTypeBean[])
            list.toArray(new EmailTypeBean[]{new EmailTypeBean()});
    }
```

Sometimes more complex conversions can be done. For example, the proxy for the getUserList method from the SecuredUserAdapter is as following:

```
    public UserSliderBean getUserList(String sessionId,
    String managerId, int page)
        throws GranException {
        Slider slider = manager.getUserList(sessionId,
                                            managerId, page);
        UserSliderBean bean = new UserSliderBean();
        bean.setId(slider.getId());
        bean.setKeyword(slider.getKeyword());
        bean.setPage(slider.getPage());
        bean.setPageSize(slider.getPageSize());
        bean.setSortOrder(slider.getSortorder());
        ArrayList list = new ArrayList();
        for (Iterator it = slider.getCol().iterator(); it.hasNext();)
            list.add(((SecuredUserBean) it.next()).getSOAP());
        bean.setUsers((UserBean[])
             list.toArray(new UserBean[]{new UserBean()}));
        return bean;
    }
```

3.4.1 Using Java SOAP

This section describes how to interact with TrackStudio from a Java application using TrackStudio SOAP API.

Description

To develop a client application which uses TrackStudio SOAP API:

1. Enable TrackStudio SOAP API

2. Start TrackStudio Enterprise

3. Implement client application. In order to work with SOAP API you should first perform authentication. For this, you need to:

- Create a DevPack class instance

```
DevPack dp = new DevPack("http://localhost:8888/TrackStudio");
```

- Call the **authenticate** method

```
String sessionId = dp.getUserService().authenticate("root","root");
```

The session ID received as the result of authentication can be used to call other methods.

The code presented below shows the example of the simplest Java client:

```
import gran.trackstudio.DevPack;

public class ATest {
    public static void main(String[] args) throws Exception {
        DevPack dp = new DevPack("http://localhost:8888/TrackStudio");
        String sessionId = dp.getUserService()
           .authenticate("root","root");
        System.out.println("Session ID is:"+sessionId);
    }
}
```

4. Compile the client application

```
javac -classpath tssoapclient.jar;axis.jar;jaxrpc.jar Test.java
```

5. Run the client application

```
C:\>java -classpath tssoapclient.jar;axis.jar;jaxrpc.jar;
commons-logging.jar;commons-discovery.jar;saaj.jar;. Test
Session ID is:297e234cfbf9889500fbf989ee890012
```

3.4.2 Using SOAP from Web Browser

This section describes how to use TrackStudio SOAP API from an Internet Browser.

Description

To use a browser to call TrackStudio SOAP API:

1. Enable TrackStudio SOAP API

2. Start TrackStudio Enterprise

3. To get the WSDL description of the service open the following URL in the browser:

```
http://<host>:<port>/TrackStudio/services/<service>?wsdl
```

For example, to get the **UserService** description open the following URL:

```
http://localhost:8888/TrackStudio/services/User?wsdl
```

4. To perform the authentication open the URL containing the service name, method name and parameters:

```
http://localhost:8888/TrackStudio/services/User?
     method=authenticate&p1=root&p2=root
```

This done you will get the following response:

```
<soapenv:Envelope>
  <soapenv:Body>
    <authenticateResponse
         soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <authenticateReturn xsi:type="soapenc:string">
         4458443e969146a510b799a6164c68bd
      </authenticateReturn>
    </authenticateResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5. Now we can use the received session id and load the root task

- Request:

```
http://localhost:8888/TrackStudio/services/Find?
                          method=findTaskById&
                          p1=4458443e969146a510b799a6164c68bd&
                          p2=1
```

- Response:

```
<soapenv:Envelope>
  <soapenv:Body>
    <findTaskByIdResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <findTaskByIdReturn href="#id0"/>
    </findTaskByIdResponse>
    <multiRef id="id0" soapenc:root="0"
              soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              xsi:type="ns1:TaskBean">
    <priorityId xsi:type="soapenc:string">2</priorityId>
    <workflowId xsi:type="soapenc:string">1</workflowId>
    <budget xsi:type="soapenc:double" xsi:nil="true"/>
    <submitdate href="#id1"/>
    <handlerId xsi:type="soapenc:string">1</handlerId>
    <abudget href="#id2"/>
    <childrenCount href="#id3"/>
    <id xsi:type="soapenc:string">1</id>
    <messageCount href="#id4"/>
    <closedate href="#id5"/>
    <name xsi:type="soapenc:string">Projects</name>
    <number xsi:type="soapenc:string">1</number>
    <submitterId xsi:type="soapenc:string">1</submitterId>
    <parentId xsi:type="soapenc:string" xsi:nil="true"/>
    <resolutionId xsi:type="soapenc:string" xsi:nil="true"/>
    <statusId xsi:type="soapenc:string">2</statusId>
    <deadline href="#id6"/>
    <updatedate href="#id7"/>
    <hasAttachments href="#id8"/>
    <description xsi:type="soapenc:string"/>
    <shortname xsi:type="soapenc:string" xsi:nil="true"/>
    <nameCutted xsi:type="soapenc:string">Projects</nameCutted>
    <categoryId xsi:type="soapenc:string">1</categoryId>
    <onSight href="#id9"/>
   </multiRef>

   <multiRef id="id2" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="soapenc:double">0.0
   </multiRef>
   <multiRef id="id9" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:boolean">true
   </multiRef>
   <multiRef id="id8" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:boolean">false
   </multiRef>
   <multiRef id="id3" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:int">1
   </multiRef>
   <multiRef id="id5" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:long">-1
   </multiRef>
   <multiRef id="id7" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:long">1099161350000
   </multiRef>
   <multiRef id="id6" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:long">-1
   </multiRef>
   <multiRef id="id4" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:int">0
   </multiRef>
   <multiRef id="id1" soapenc:root="0"
             soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             xsi:type="xsd:long">1084368653000
   </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

6. In case of invalid parameters the Exception is generated:

Request:

```
http://localhost:8888/TrackStudio/services/Find?
                          method=findTaskById&
                          p1=4458443e969146a510b799a6164c68bd&
                          p2=10
```

- Response:

```
<soapenv:Envelope>
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server.userException</faultcode>
      <faultstring>
         gran.exception.TaskNotFoundException: Specified task not found.
      </faultstring>
      <detail>
        <ns1:stackTrace>
          gran.exception.TaskNotFoundException: Specified task not found.
             at com.trackstudio.tools.HibernateUtil.getObject(HibernateUtil.java:404)
        </ns1:stackTrace>
        <ns2:hostname>trackstu-server</ns2:hostname>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

3.4.3 Using .NET SOAP

This section describes how to realize the interaction between TrackStudio and a .NET application using TrackStudio SOAP API.

Description

To develop the client application that uses TrackStudio SOAP API you are to:

1. Enable TrackStudio SOAP API

2. Start TrackStudio Enterprise

3. Create XML Web service proxy classes.

Create a proxy class for each service. Every proxy class must be a separate namespace. To generate a proxy to access the User service execute the following command:

```
wsdl.exe http://localhost:8888/TrackStudio/services/User?wsdl
/out:dll/UserService.cs /namespace:User
```

As a result the UserService.cs proxy class will be created in the dll folder. Repeat these steps for other services and link all the created proxy classes to the ts.dll library:

```
csc.exe /t:library /out:ts.dll dll\*.cs
```

4. Implement client application. In order to work with SOAP API you should first perform authentication. To do this:

- Create proxy class instance

```
UserService uSrv = new UserService();
```

- Set TrackStudio SOAP Service URL

```
tSrv.Url = "http://localhost:8888/TrackStudio/services/User";
```

- Call the **authenticate** method

```
string sessionId = uSrv.authenticate("root", "root");
```

The session id received as the result of authentication can be used to call other methods.

The code presented below shows the example of the simple .NET client application:

```
using System;
using User;
public class SoapTest {
    public static void Main(string[] args) {
        UserService uSrv = new UserService();
        uSrv.Url = "http://localhost:8888/TrackStudio/services/User";
        string sessionId = uSrv.authenticate("root", "root");
        Console.WriteLine("Session ID is: " + sessionId);
    }
}
```

5. Compile the client application

```
csc.exe /reference:ts.dll SoapTest.cs
```

6. Run the client application

```
C:\soap\dotNET>SoapTest.exe
Session ID is: 297e234cfbf9889500fbf989ee890012
```

Index