TrackStudio Enterprise 3.5

Documentation

Copyright (c) 2002-2006. All rights reserved.

Table of Contents

New Features 1

Installation Guide 3

TrackStudio Updates and Upgrades 3

Upgrading from Version 3.5.x 3

Upgrading from Version 3.2.x 4

Upgrading from Version 3.1.x 5

Upgrading from Version 3.0.x 6

Windows Installation 7

Installing TrackStudio SA for Windows 7

Installing TrackStudio WAR for Windows 7

Running as Windows Service 8

Integrating IIS and TrackStudio 9

Configuring MS Active Directory Authentication 11

Configuring NTLM Authentication 13

UNIX Installation 14

Installing TrackStudio SA for UNIX 14

Installing TrackStudio WAR for UNIX 15

Configuring X11 Server 15

Installing an SSL Certificate 16

Server-specific Notes 17

Creating and Managing the Database 18

Initializing a DB2 Database 18

Initializing an HSQLDB Database 19

Initializing a PostgreSQL Database 19

Initializing an ORACLE Database 20

Initializing an MS SQL Server Database 20

Initializing a MySQL Database 22

Initializing a Firebird Database 22

Creating a Database for Performance Testing 23

Backing Up and Restoring the Database 24

Importing and Exporting the Database 24

Configuring TrackStudio Cluster 26

Integrating IDE and TrackStudio 27

Installing IDEA Plug-in 27

Installing JBuilder Plug-in 27

Installing Eclipse Plug-in 28

Integrating SCM and TrackStudio 28

Integrating E-mail Client and TrackStudio 30

Configuring E-mail Notification 30

Configuring E-mail Submission 31

Using JES for E-mail Integration 33

Integrating Serence KlipFolio and TrackStudio 34

TrackStudio Configuration Properties 35

Changing the TrackStudio URL 40

Adding your Web Site to the TrackStudio Server 40

User's Guide 42

Implementation Guide 42

Sample Project Overview 43

Concepts 48

User Interface Concepts 49

Task Concepts 50

User Concepts 51

Workflow Concepts 51

Message Concepts 52

Category Concepts 52

User Status Concepts 53

Filter Concepts 53

Report Concepts 53

Custom Field Concepts 54

Script Concepts 54

Self-registration Concepts 56

E-mail Template Concepts 56

E-mail Notification Concepts 56

E-mail Submission Concepts 57

CSV Import Concepts 58

Step-by-Step Guides 59

Managing Projects and Bugs 59

Creating a Project 59

Adding a Bug 61

Editing Task Properties 61

Assigning a Task 62

Adding Comments to a Task 64

Changing the Task State 65
Attaching a File to a Task 66
Moving Tasks 67
Linking Tasks 69
Linking Users to a Task 71
Generating a Report 72
Bulk Updating Tasks 74
Closing a Bug 77
Deleting a Task 78
Managing User Accounts 78
Establishing a User Group Account 78
Creating a User Account 81
Creating a Customer Account 82
Editing Account Properties 83
Granting Users Access to a Project 83
Granting Users Access to Other Users 84
Customers Creating Accounts for Themselves 85
Finding a User Account by Login 87
Finding a User Account by Properties 87
Changing a Password 88
Moving a User Account 89
Locking a User Account 89
Deleting a User Account 90
Customizing Task and User Processing 91
Hiding Unused Fields 91
Hiding Menu Items, Tabs or Buttons 93
Adding a Custom Field 94
Defining Who Can View and Edit Custom Fields 96
Restricting Handler List 98
Setting Default Handler for a Project 99
Restricting Who Can Close Task 99
Allowing Group Handler Assignment 100
Defining Which Users Can Create, Edit or Delete Tasks 101
Choosing Localized Interface 101
Defining Task Priorities 102
Creating a Script 103
Calculating a Custom Field Value 104
Implementing Triggers 109
Importing Data from CSV File 113
Searching and Analyzing Tasks 115
Finding a Task 115
Filtering Subtasks by Properties 116

Filtering Tasks Using AND/OR/NOT 117
Sorting Tasks 118
Sorting Tasks by Product 118
Analyzing Tasks Distribution 119
Managing Categories and Workflows 120
Creating a Workflow 120
Creating a Category 124
Defining Category Dependency 125
Setting Workflow for Task Category 126
Managing E-mail Notification and Submission 126
Receiving E-mail Notification when Tasks Change 126
Periodically Receiving Lists of Tasks by E-mail 128
Sending Alert for Overdue Tasks 129
Adding a Task by E-mail 130
Adding a Message by E-mail 131
Customizing E-mail Notification Templates 132

Reference **134**
Task Properties 134
User Properties 135
Message Properties 136
Custom Field Properties 136
Assigned Status Properties 137
Task Filter Properties 138
User Filter Properties 139
Report Properties 140
Full Text Search Reference 141
Category Properties 142
E-mail Notification Rule Properties 143
Filter Subscription Rule Properties 143
E-mail Import Rule Properties 144
Self-registration Rule Properties 144
CSV Import Script Reference 145

Developer's Guide 148

Localizing User Interface **148**

Extending TrackStudio Functionality **149**
Implementing Adapters 149

Building TrackStudio from Source **151**

Integrating with Third-Party Systems **151**
Using Java SOAP 152

Using SOAP from Web Browser 153
Using .NET SOAP 155

Index a

TrackStudio Enterprise 3.5

1 New Features

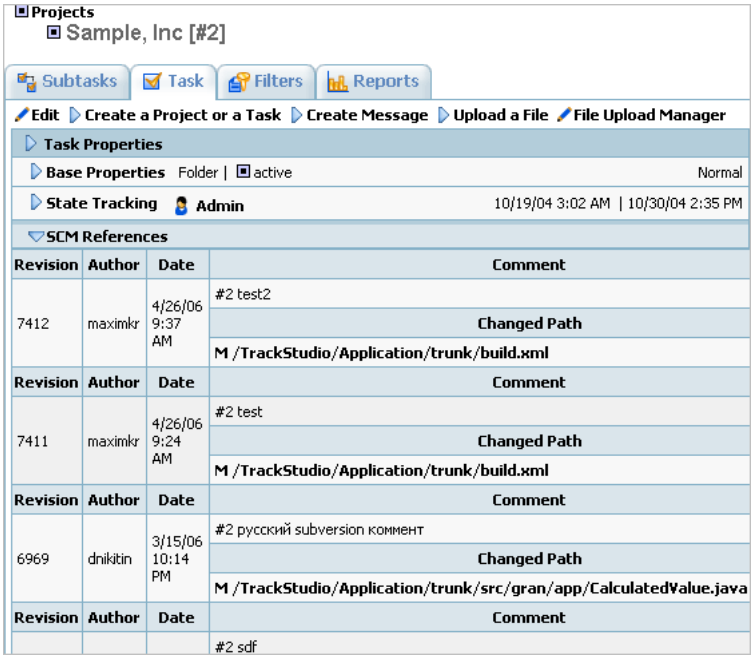
The following topic describes some new features of TrackStudio.

Description

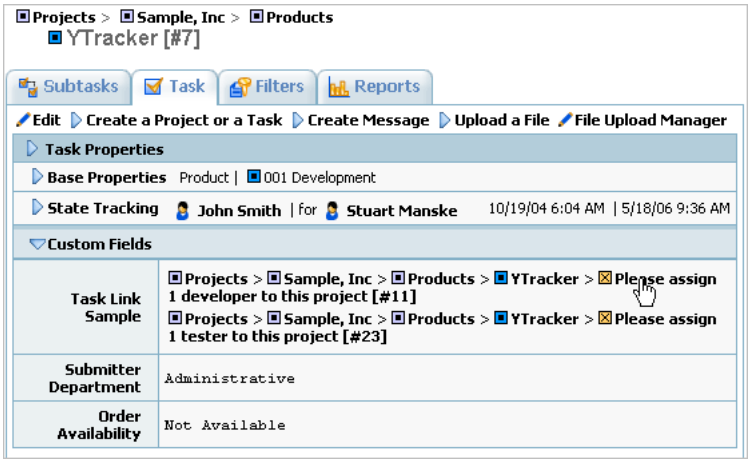
TrackStudio Enterprise is one of the most flexible Java-based issue tracking systems available, supporting the widest range of commercial and open source application servers and DBMSs. TrackStudio Enterprise can be run on any computer platform.

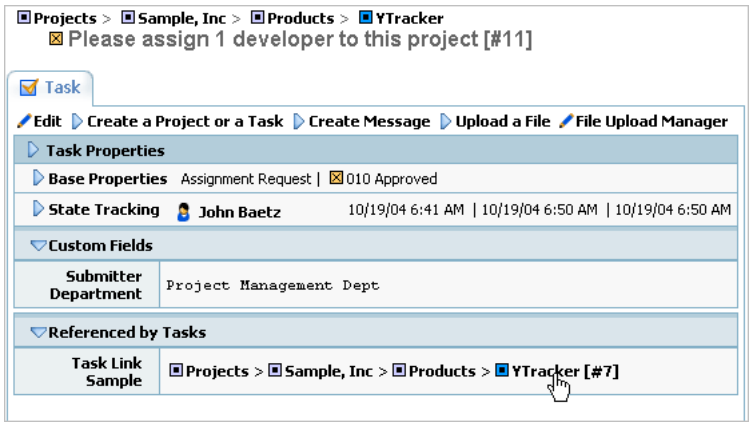
The following major new features have been incorporated in TrackStudio 3.5:

- Totally new **SCM integration** has been implemented.

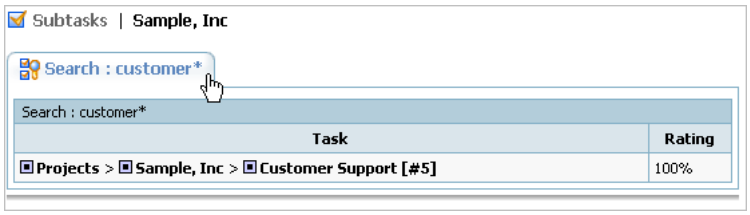


- **Linking** of tasks and users has been improved significantly.

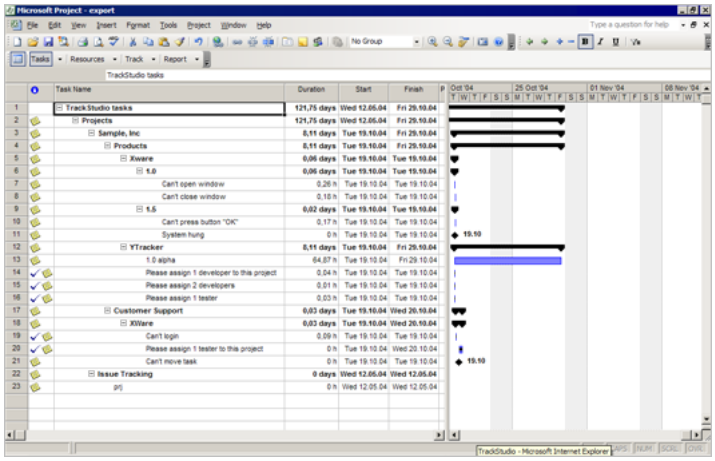




- Simplified **full text search** was implemented.



- **Microsoft Project reporting** has been improved.



2 Installation Guide

2.1 TrackStudio Updates and Upgrades

2.1.1 Upgrading from Version 3.5.x

The following topic describes how to update from TrackStudio 3.5.x to TrackStudio 3.5.y.

Description

To update TrackStudio WAR:

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

To update TrackStudio SA:

1. Download the latest TrackStudio SA.
2. Stop the TrackStudio 3.5.x instance.
3. Install the TrackStudio 3.5.y into a separate directory.
4. Configure the new TrackStudio instance using **sman**.
5. **HSQLDB only:** Copy all database file (tsdb.*) from the TrackStudio 3.5.x home directory into the home directory for the new instance.
6. Copy all files from the TrackStudio 3.5.x upload directory into the **Upload directory** for the new instance. To determine the upload directory path, see the **trackstudio.uploadDir** property in the **trackstudio.properties** file.
7. Delete the contents of the **Index directory**. To determine the index directory path, see the **trackstudio.indexDir** property in the **trackstudio.properties** file.
8. Start TrackStudio 3.5.y
9. Update e-mail templates:

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

Property	Value
Subject	<${task.getProjectAlias()} | ${addval.reason} | ${addval.taskViewFactory.inEmailText(task).getNumber()}>: ${addval.taskViewFactory.inEmailText(task).getName()}

Content Type	text
From	Submitter
Reply To	TrackStudio
E-mail Body	Copy/paste from **templates/common_text.ftl**

Notes

- Minor version updates (3.5.x to 3.5.y) do not require a 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.
- If you are using the default HSQLDB DBMS, your database files look like *tsdb.lck, tsdb.log, tsdb.properties* and *tsdb.script.* Do not delete them.
- Both the **Upload** and **Index directories** can be specified in a flexible manner in **trackstudio.*** configuration files, to set them outside the TrackStudio instance directory. This has implications when upgrading.

2.1.2 Upgrading from Version 3.2.x

The following topic describes how to update from TrackStudio 3.2.x to TrackStudio 3.5.y.

Description

To update TrackStudio WAR:

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

To update TrackStudio SA:

1. Download the latest TrackStudio SA.
2. Stop the TrackStudio 3.2.x instance.
3. Install the TrackStudio 3.5.y into a separate directory.
4. Configure the new TrackStudio instance using **sman**.
5. **HSQLDB only:** Copy all database files (test.*) from the TrackStudio 3.2.x home directory into the home directory for the new instance.
6. Copy all files from the TrackStudio 3.2.x upload directory into the **Upload directory** for the new instance. To determine the upload directory path, see the **trackstudio.uploadDir** property in the **trackstudio.properties** file.
7. Delete the contents of the **Index directory**. To determine the index directory path, see the **trackstudio.indexDir** property in the **trackstudio.properties** file.
8. Start TrackStudio 3.5.y
9. Update e-mail templates:

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

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

Notes

- Upgrading from version 3.2.x do not require a 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.
- If you are using the default HSQLDB DBMS, your database files look like *tsdb.lck, tsdb.log, tsdb.properties* and *tsdb.script.* Do not delete them.
- Both the **Upload** and **Index directories** can be specified in a flexible manner in **trackstudio.*** configuration files, to set them outside the TrackStudio instance directory. This has implications when upgrading.

2.1.3 Upgrading from Version 3.1.x

The following topic describes how to update from TrackStudio 3.1.x. to 3.5.y.

Description

To upgrade the system from TrackStudio 3.1.x to TrackStudio 3.5.y:

1. Stop the TrackStudio 3.1.x instance.
2. Backup your database. The backup will allow you to continue to use version 3.1.x until until any unforeseen problems with TrackStudio 3.5.y are solved.
3. Install the TrackStudio 3.5.y into a separate directory.
4. Upgrade database:
   - **TrackStudio SA**: Run Server Manager from TrackStudio 3.5.y, go to the **Database -> Database Management** tab, and press the **Upgrade Database** button.
   - **TrackStudio WAR**: Execute the update script for your DBMS. Use the appropriate script from **sql\upgrade-31-35** directory.
5. Configure TrackStudio.
   - **TrackStudio SA**: Run Server Manager from TrackStudio 3.5.y.
   - **TrackStudio WAR**: Edit the trackstudio properties files.
6. Copy all files from the TrackStudio 3.1.x upload directory into the **Upload directory** for the new instance. To determine the upload directory path, see the **trackstudio.uploadDir** property in the **trackstudio.properties** file.
7. Delete the contents of the **Index directory**. To determine the index directory path, see the **trackstudio.indexDir** property in the **trackstudio.properties** file.
8. Start TrackStudio 3.5.y
9. Login as **root.**
10. Update e-mail templates:

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

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

Important Information Regarding Upgrading Your Database:

- XML Import/Export allows you to transfer the data of TrackStudio 3.5 between various database types, but it cannot be used to transfer data between various versions of TrackStudio.
- If errors or problems occur while upgrading the database, you should contact us and continue using TrackStudio 3.1.x until the problem is resolved.
- If you are using the default HSQLDB DBMS, your database files look like *tsdb.lck, tsdb.log, tsdb.properties* and *tsdb.script.* Do not delete them.
- Both the **Upload** and **Index directories** can be specified in a flexible manner in **trackstudio.*** configuration files, to set them outside the TrackStudio instance directory. This has implications when upgrading.
- In previous versions, fields of the class **Timestamp** were used for datetime objects in the database. The ability to search datetime objects using greater-than or less-than operators was required, so as of version 3.5, we have changed from **Timestamp** to **Calendar** for all datetime fields. This change affects scripts as well, because they have direct access to the TrackStudio object model. Here are a few examples of how you can upgrade your 3.1 scripts to be compatible with 3.5:
- Version 3.1:

```
task.getSubmitdate().getTime()
task.getUpdatedate().getTime()
```

- Version 3.5:

```
task.getSubmitdate().getTime().getTime()
task.getUpdatedate().getTime().getTime()
```

See Also

- Backing Up and Restoring the Database (see page 24)

2.1.4 Upgrading from Version 3.0.x

The following topic describes how to update from TrackStudio 3.0.x to TrackStudio 3.5.y.

Description

To upgrade the system from TrackStudio 3.0.x to TrackStudio 3.5.y:

1. Download the TrackStudio 3.1 using the following link: http://www.trackstudio.com/products-edownload.html
2. Upgrade TrackStudio database from 3.0.x to 3.1.y using **sman** or database upgrade script.
3. Run TrackStudio to finish the database conversion.
4. Ensure that everything works.
5. Upgrade from TrackStudio 3.1.x to 3.5.y.

2.2 Windows Installation

2.2.1 Installing TrackStudio SA for Windows

The following topic describes TrackStudio Enterprise installation and configuration (Standalone distribution).

Description

To install and configure TrackStudio SA:

1. Run Server Manager (**sman.exe**).
2. Press the **Start** button to run the TrackStudio server.
3. Once the server is running, click the **Login** button.

Use the following to log on:

Role	Description	Login	Password
System Administrator	This user has full rights to configure TrackStudio settings and create global system defaults within the solution for your entire company.	root	root
Issue Tracker Managed Administrator	User who is authorized to manage pre-defined TrackStudio configuration for issue tracking.	itadmin	itadmin
Sample Database Managed Administrator	User who is authorized to manage TrackStudio Sample Database.	jsmith	jsmith

Notes

If you are using the default HSQLDB DBMS, your database files look like *tsdb.lck, tsdb.log, tsdb.properties* and *tsdb.script.* Do not delete them.

2.2.2 Installing TrackStudio WAR for Windows

The following topic describes TrackStudio Enterprise installation and configuration (WAR distribution).

Description

To install and configure TrackStudio WAR:

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** system environment variable to specify the directory name with configuration files (*.properties).
5. Deploy the **TrackStudio.war**. You can also unpack the WAR file and perform the deployment of the directory structure.
6. Run the application server.
7. The application is available at **http://localhost:port/TrackStudio**

Use the following to log on:

Role	Description	Login	Password
System Administrator	This user has full rights to configure TrackStudio settings and create global system defaults within the solution for your entire company.	root	root
Issue Tracker Managed Administrator	User who is authorized to manage pre-defined TrackStudio configuration for issue tracking.	itadmin	itadmin
Sample Database Managed Administrator	User who is authorized to manage TrackStudio Sample Database.	jsmith	jsmith

Remarks

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

See Also

- Server-specific Notes (see page 17)

2.2.3 Running as Windows Service

The following topic describes how to run TrackStudio as a Windows service.

Description

Jetty (TrackStudio SA default application server)

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 the Windows service:

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

To uninstall the Windows service:

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

2.2.4 Integrating IIS and TrackStudio

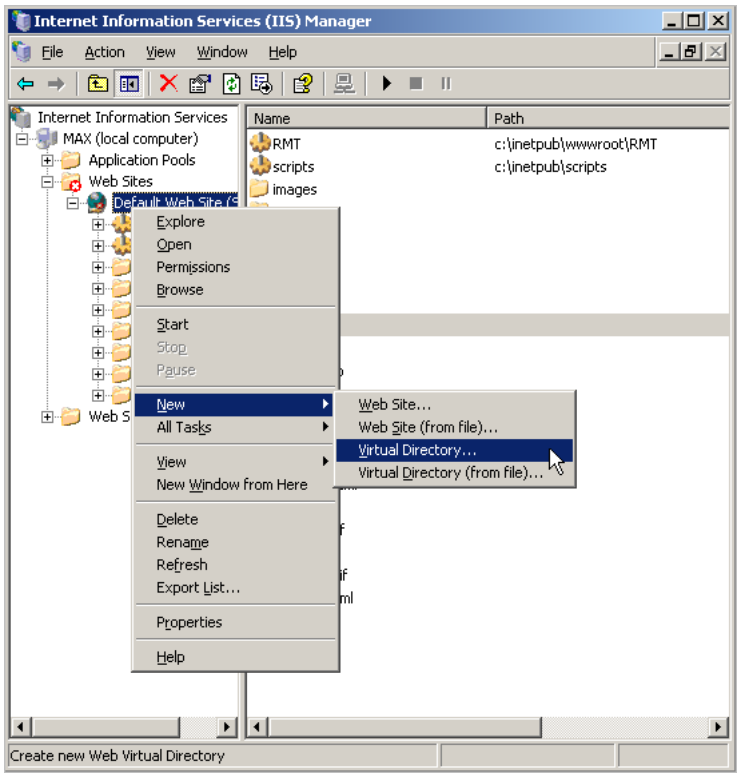
The following topic describes how to integrate TrackStudio with Internet Information Server.

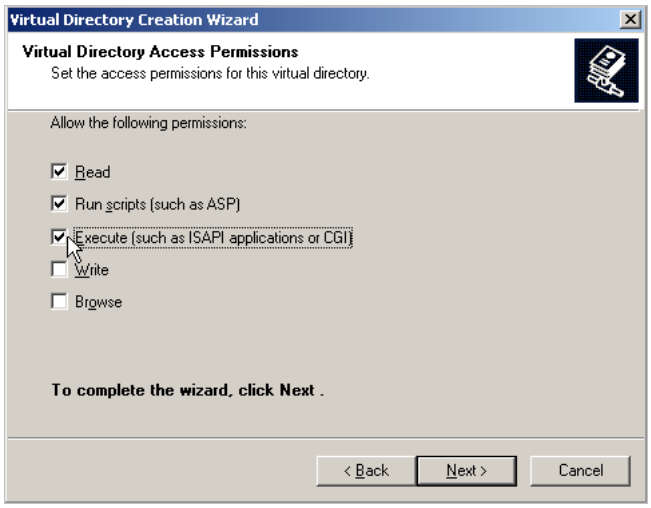
Description

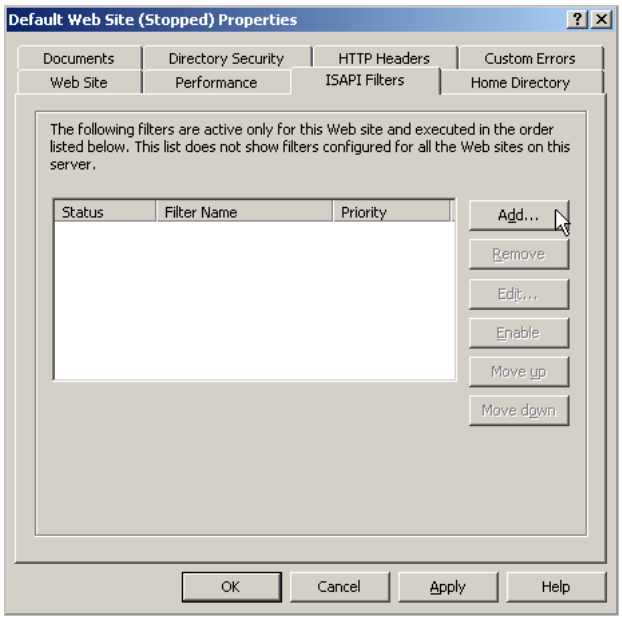
This topic describes how to permit access to TrackStudio via an existing Internet Information Server. TrackStudio can be configured so that when you reference an IIS "virtual folder" (e.g. */TrackStudio*), you will be redirected to TrackStudio.

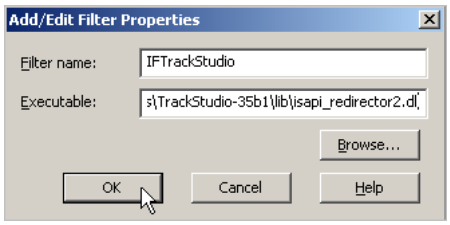
To integrate TrackStudio with Internet Information Server:

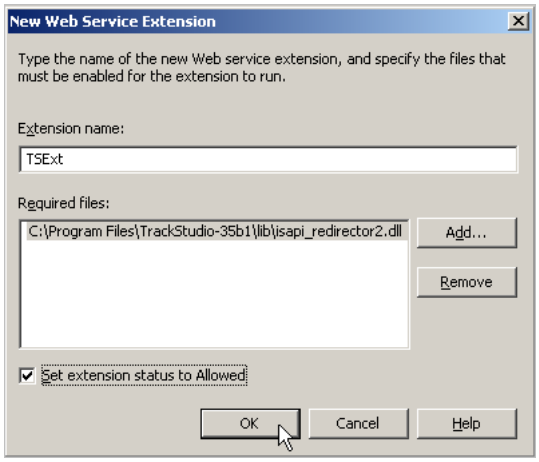
1. Start the IIS administration software (**Start -> Programs -> Administrative Files -> Internet Information Services**).
2. Create a virtual TrackStudio folder for one of the web sites, e.g. for the *Default Web Site* (**Action -> New -> Virtual Directory**).
   - Specify the **<TRACKSTUDIO_HOME>** 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**


Internet Information Services (IIS) Manager
File Action View Window Help
Internet Information Services
MAX (local computer)
Application Pools
Web Sites
Default Web Site
Name
Path
RMT
c:\inetpub\wwwroot\RMT
scripts
c:\inetpub\scripts
images
Explore
Open
Permissions
Browse
Start
Stop
Pause
New
All Tasks
View
New Window from Here
Delete
Rename
Refresh
Export List...
Properties
Help
Web Site...
Web Site (from file)...
Virtual Directory...
Virtual Directory (from file)...
Create new Web Virtual Directory


Virtual Directory Creation Wizard
Virtual Directory Access Permissions
Set the access permissions for this virtual directory.
Allow the following permissions:
Read
Run scripts (such as ASP)
Execute (such as ISAPI applications or CGI)
Write
Browse
To complete the wizard, click Next .
< Back
Next >
Cancel







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. You can safely ignore this.

2.2.5 Configuring MS Active Directory Authentication

The following topic describes how to configure the user authentication via the **Microsoft Active Directory Service**.

Description

To configure user authentication via the Microsoft Active Directory Service:

1. Login into **Microsoft Windows** as *Administrator*
2. Export the LDAP context to a file.

```
ldifde –f ldap.txt
```

3. Open the resulting *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.security.properties:**

```
trackstudio.useLDAP yes
```

5. Set the base DN to **cn=users** for the 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 LDAP (AD) server:

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

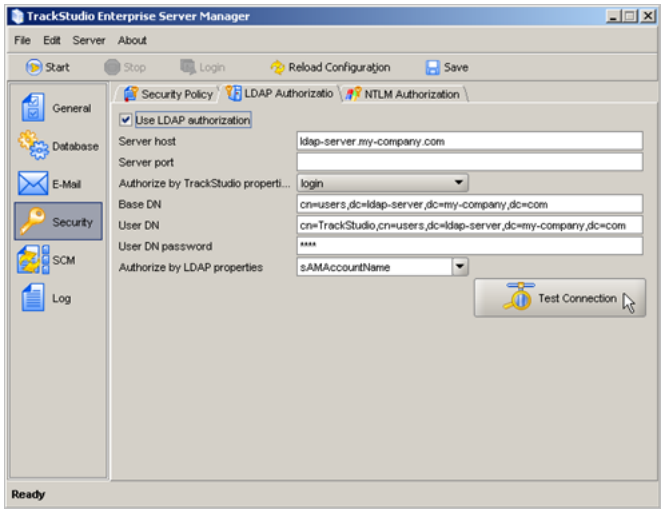
7. To login by **Name** set:

```
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 database query and finds the user in the local database 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 changed in the database, but not in the LDAP server.
- A user can log in if his/her password matches the one stored in the database or the one specified in LDAP. To avoid authorization via the local database, you should remove **com.trackstudio.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

2.2.6 Configuring NTLM Authentication

The following topic describes how to configure the user authentication via **NTLM**.

Description

To configure the user authentication via NTLM:

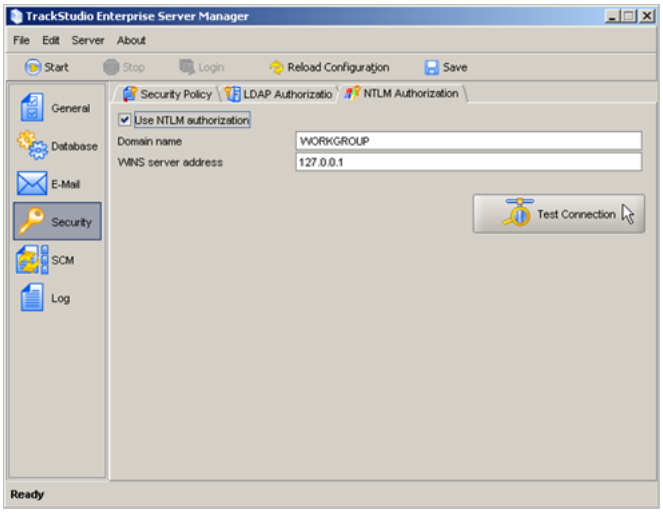
1. Login into **Microsoft Windows** as Administrator.
2. Run the **Configure Your Server** application. (**Control Panel / Administrative tools / Configure Your Server**).
3. Use the **Networking/DHCP** tab to configure a DHCP server if necessary.
4. Use the **Networking/DNS** tab to configure DNS if necessary.
5. Use the **Active Directory** tab to configure a domain controller if necessary.
6. Use the **Windows Components wizard** to install WINS if necessary.
7. Enable NTLM in **trackstudio.security.properties**

```
trackstudio.useNTLM yes
```

8. Specify a domain name and WINS address in **trackstudio.security.properties**

```
jcifs.smb.client.domain=WORKGROUP
jcifs.netbios.wins=192.168.100.1
```

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



How it works:

If **trackstudio.useNTLM** is set to **yes**, TrackStudio will use the NTLM protocol as the mechanism of users' authentication. This mechanism provides the possibility of saving logon data about authorized users using WINS a service, and allows users to avoid entering their login name the next time they access TrackStudio.

Notes

- The NTLM mechanism uses the WINS and DNS services, which means your network must have an accessible host configured as a domain controller. A domain controller can be installed with Windows Server operating systems only. Use either Microsoft Windows 2000 Advanced Server or Microsoft Windows 2003.
- You have to use your TrackStudio **login** in the **Login** window.
- Even if you use NTLM authorization, you will have to register a new user in TrackStudio first.

2.3 UNIX Installation

2.3.1 Installing TrackStudio SA for UNIX

The following topic describes TrackStudio Enterprise installation and configuration (Standalone distribution).

Description

To install and configure TrackStudio SA on the server with X11 running:

1. Start TrackStudio Enterprise Server Manager (**sman**). A GUI should appear.
2. Press the **Start** button to run the TrackStudio server.
3. Once the server is running, click the **Login** button.

To install and configure TrackStudio SA on the server with X11 installed, but not running:

1. Start TrackStudio Enterprise default servlet container (**jetty**).
2. Open the following URL: **http://hostname:8888/TrackStudio**

Use the following to log on:

Role	Description	Login	Password
System Administrator	This user has full rights to configure TrackStudio settings and create global system defaults within the solution for your entire company.	root	root
Issue Tracker Managed Administrator	User who is authorized to manage pre-defined TrackStudio configuration for issue tracking.	itadmin	itadmin
Sample Database Managed Administrator	User who is authorized to manage TrackStudio Sample Database.	jsmith	jsmith

Remarks

If you run TrackStudio in a *nix VPS (Virtual Private Server), you may encounter VM object heap errors, which may be due to how a VPS allocates memory on a shared physical machine. The JVM attempts to make a guess at how much memory it will be able to use based on the information it can gather from the same sources as the tools "free" and "top" use. For example, in a VServer VPS this reports memory on the physical host, and does not reflect the limits in place for each VPS.

The workaround is to explicitly tell the JVM how much memory to use:

```
> sman -J-Xmx256m
> jetty -J-Xmx256m
```

Notes

If you are using the default HSQLDB DBMS, your database files look like *tsdb.lck, tsdb.log, tsdb.properties* and *tsdb.script.*

Do not delete them.

See Also

- Configuring X11 Server (see page 15)

2.3.2 Installing TrackStudio WAR for UNIX

The following topic describes TrackStudio Enterprise installation and configuration (WAR distribution).

Description

To install and configure TrackStudio WAR:

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** system environment variable to specify the directory name with configuration files (*.properties).
5. Deploy the **TrackStudio.war**. You can also unpack the WAR file and perform the deployment of the directory structure.
6. Run the application server.
7. The application is available at **http://localhost:port/TrackStudio**

Use the following to log on:

Role	Description	Login	Password
System Administrator	This user has full rights to configure TrackStudio settings and create global system defaults within the solution for your entire company.	root	root
Issue Tracker Managed Administrator	User who is authorized to manage pre-defined TrackStudio configuration for issue tracking.	itadmin	itadmin
Sample Database Managed Administrator	User who is authorized to manage TrackStudio Sample Database.	jsmith	jsmith

Remarks

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

See Also

- Configuring X11 Server (see page 15)
- Server-specific Notes (see page 17)

2.3.3 Configuring X11 Server

The following topic describes how to configure X11 to run TrackStudio.

Description

TrackStudio does not contain graphical libraries for generating colors, fonts or other AWT information. Java relies on your system's libraries for providing such information, and so 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 a GUI interface and graphics

card already exist by default.

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:

- Install the X11 Server and set the DISPLAY environment variable.
- 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 5.x, you would specify these options in the **catalina.bat** or **catalina.sh** file for **CATALINA_OPTS**.

2.3.4 Installing an SSL Certificate

The following topic describes how to install an SSL certificate for jetty.

Description

To install SSL certificate for jetty:

1. Create a keystore using **keytool**, supplied with the Sun JDK. When creating a certificate, you must specify *keystorePassword* and *keyPassword.*

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

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 your CSR to a Certificate Authority such as Verisign or XRamp, and purchase a SSL certificate. The CA will return a **cert.crt** file to you. 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:

1. Create a Certificate Authority by running:

```
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 a demo cert under MS Internet Explorer. Use a certificate from a Certificate Authority such as Verisign or XRamp to solve this issue.

2.4 Server-specific Notes

The easiest way to get TrackStudio running is to use the TrackStudio SA distribution, which comes preconfigured with everything you need. However if you already have a Java application server running, TrackStudio can be deployed within it.

Description

JBoss 4.0.3SP1 notes:

Set the **log4j.defaultInitOverride** system property by adding

```
-Dlog4j.defaultInitOverride=true
```

to the **JAVA_OPTS** line your **run.bat** (Windows) or **run.conf** (Linux) file.

Resin 3.0.17 notes:

After the **TrackStudio.war** deployment, create the **[RESIN_HOME]/webapps/WEB-INF/lib** directory. Unpack the **TrackStudio.war** somewhere and copy the following files into this directory:

- commons-beanutils.jar
- commons-collections-2.1.1.jar
- commons-digester.jar
- commons-discovery.jar
- commons-logging.jar
- log4j-1.2.12.jar

Weblogic 9.0 notes:

Uncomment the following line in **trackstudio.hibernate.properties**:

```
# hibernate.query.factory_class org.hibernate.hql.classic.ClassicQueryTranslatorFactory
```

Set character encoding in the **web.xml** if required:

```
<jsp-config>
...
  <jsp-property-group>
  ...
    <page-encoding>UTF-8</page-encoding>
  </jsp-property-group>
  ...
</jsp-config>
```

Jetty 5.1.x notes:

- Delete **WEB-INF/lib/xercesImpl-2.6.2.jar** from **TrackStudio.war**.
- Copy **WEB-INF/lib/log4j-*.jar** to Jetty's **ext** directory.

2.5 Creating and Managing the Database

2.5.1 Initializing a DB2 Database

The following topic describes how to configure TrackStudio for use with a DB2 database management system.

Description

1. Create user tablespace and temp system tablespace.
2. Open a DB2 command window (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 -> Database Connectivity** tab, and enter the JDBC connection properties.
   - **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect org.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 -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Execute **sql\install\trackstudio-db2.sql:**

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

2.5.2 Initializing an HSQLDB Database

The following topic describes how to configure TrackStudio for use with an HSQLDB database management system.

Description

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

```
hibernate.dialect org.hibernate.dialect.HSQLDialect
hibernate.connection.url jdbc:hsqldb:file:tsdb;shutdown=true
hibernate.connection.driver_class org.hsqldb.jdbcDriver
hibernate.connection.username sa
hibernate.connection.password
```

2. Initialize the database:
   - **TrackStudio SA:** Go to the **Database -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Execute **sql\install\trackstudio-hsql.sql:**

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

2.5.3 Initializing a PostgreSQL Database

The following topic describes how to configure TrackStudio for use with a PostgreSQL database management system.

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

```
hibernate.dialect org.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 -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Execute **sql\install\trackstudio-pgsql.sql:**

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

Notes

- To backup the database execute:

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

- To restore the database execute:

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

2.5.4 Initializing an ORACLE Database

The following topic describes how to configure TrackStudio for use with an ORACLE database management system.

Description

1. Create the Tablespace.
2. Create a TrackStudio user.
3. Grant **DBA** and **Resource** role to the created user.
4. Configure the database connection properties. 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.
   - **TrackStudio SA:** Go to the **Database -> Database Connectivity** tab, and enter the JDBC connection properties.
   - **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect org.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 -> 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
```

2.5.5 Initializing an MS SQL Server Database

The following topic describes how to configure TrackStudio for use with a Microsoft SQL Server database management system.

Description

1. Start **Enterprise Manager.**
2. Create the database.
3. Configure the database connection properties:
   - **TrackStudio SA:** Go to the **Database -> Database Connectivity** tab, enter the JDBC connection properties.
   - **TrackStudio WAR:** Edit the **trackstudio.hibernate.properties**:

```
hibernate.dialect org.hibernate.dialect.SQLServerDialect
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 -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Start **Query Analyzer** and execute **sql\install\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 Microsoft SQL Server. If you need to store national characters in Microsoft SQL Server - please consider using a national character encoding (TIS-620 for Thai characters, for example). Change data types:

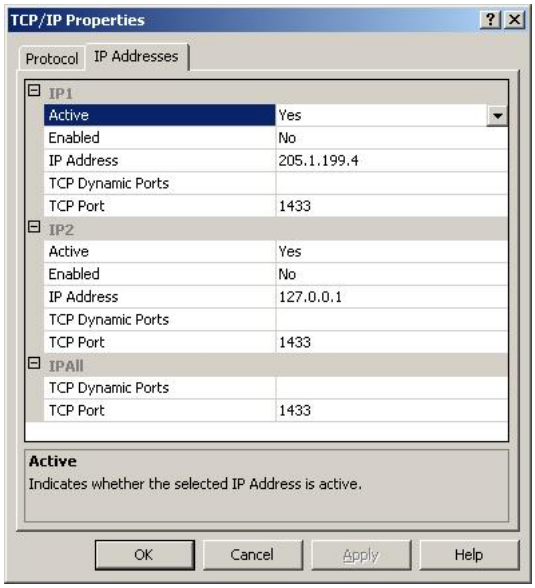
- varchar to nvarchar
- text to ntext
- char to nchar

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.

Connecting to **Microsoft SQL Server 2005 Express** (SSE):

1. Install SSE using **Mixed Mode**. Input strong password for **SA** (SSE will not allow for a null password) make sure to also include the SQL Browser as part of the install, it with save time later.
2. Using **MS SQL Server Management Studio Express** create an empty database.
3. In order to allow remote connection to SSE start **SQL Server Surface Area Configuration**:
   1. Select **Surface Area Configuration for Services and Connection.**
   2. Select *server instance* -> **Database Engine** -> **Remote Connections**.
   3. Select **Local and remote connections**.
   4. Select **Using both TCP/IP and named pipes**.
   5. Select **Apply** then **OK**.
4. Using **MS SQL Server Configuration Manager**, in the right pane:
   1. Double-click **TCP/IP** and select **IP Addresses** tab.
   2. Blank out **TCP Dynamic ports**.
   3. Input *1433* in **TCP Port.**
   4. Click **Apply** then **OK.**
   5. Exit **Configuration Manager**.
5. In **MS SQL Server Management Studio Express**:
   1. Right click database instance.
   2. On pull-down menu select **Restart.**
   3. Exit **Server Management Studio Express**.

6. Stop TrackStudio.
7. Using **TrackStudio Server Manager**, don't forget to input strong password for **SA** in when selecting **MS SQL** database.



2.5.6 Initializing a MySQL Database

The following topic describes how to configure TrackStudio for use with a MySQL database management system.

Description

1. For MySQL 4.1 (Win) start MySQL Command Line Client or start mysql.exe the from command line:

```
shell> mysql -u root
```

2. Create the database:

```
mysql> create database trackstudio character set utf8;
mysql> commit;
mysql> use trackstudio;
```

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

```
hibernate.dialect org.hibernate.dialect.MySQLInnoDBDialect
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql://localhost/trackstudio?useUnicode=true
                            &characterEncoding=UTF-8&autoReconnect=true
hibernate.connection.username root
hibernate.connection.password
```

4. Initialize the database:
   - **TrackStudio SA**: Go to the **Database -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Execute following command from command line:

```
shell>mysql -u root -p trackstudio <"trackstudio-mysql.sql"
```

2.5.7 Initializing a Firebird Database

The following topic describes how to configure TrackStudio for use with a Firebird database management system.

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. **TrackStudio SA:** exit from **isql**:

```
SQL> exit;
```

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

```
hibernate.dialect org.hibernate.dialect.FirebirdDialect
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 -> Database Management** tab and click the **Create Database** button.
   - **TrackStudio WAR**: Execute **sql\install\trackstudio-firebird.sql:**

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

Notes

Before upgrading the database, execute the following:

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

2.5.8 Creating a Database for Performance Testing

The following topic describes how to create large TrackStudio databases to use as benchmarks.

Description

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

To prepare a database:

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

```
> massive/massive
```

To create a test database:

1. Run TrackStudio Server Manager (**sman**).
2. Select the **Database -> Database Connectivity** tab.
3. Enter the JDBC connection properties of the empty database (without any tables).
4. Select the **Database -> Database Management** tab.
5. Enter the generated XML file name
6. Click the **Create Database** button.

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 TrackStudio, create the file **skipindex.flag** in the directory that is specified in the **trackstudio.indexDir** parameter in **trackstudio.properties**.

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

2.5.9 Backing Up and Restoring the Database

The following topic describes how to backup and restore the TrackStudio database.

Description

To backup the TrackStudio database:

1. Stop TrackStudio.
2. Use DBMS utilities or TrackStudio Server Manager (available in TrackStudio SA only) to backup the database. To backup HSQLDB database, copy *tsdb.** from **TrackStudio Home** directory.
3. Backup the TrackStudio Upload Directory. Check the **trackstudio.uploadDir** property in the **trackstudio.properties** file to determine the upload directory path.
4. Backup TrackStudio configuration files:
   - **trackstudio.properties**
   - **trackstudio.adapters.properties**
   - **trackstudio.hibernate.properties**
   - **trackstudio.license.properties**
   - **trackstudio.log4j.properties**
   - **trackstudio.mail.properties**
   - **trackstudio.security.properties**
   - **trackstudio.scm.properties**
5. Start TrackStudio

To restore TrackStudio database:

1. Stop TrackStudio.
2. Restore the database.
3. Restore the **Upload directory** content.
4. Delete the contents of the **Index directory**.
5. Restore the configuration files.
6. Start TrackStudio.

2.5.10 Importing and Exporting the Database

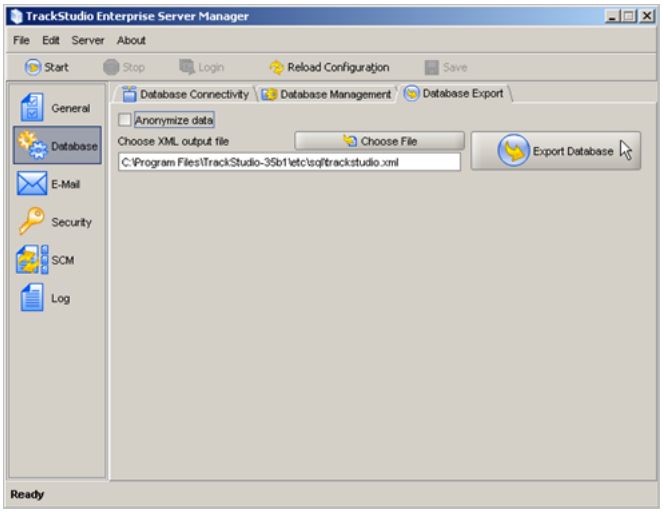
The following topic describes how to transfer the data stored in the TrackStudio database to another DBMS.

Description

To export the data into XML:

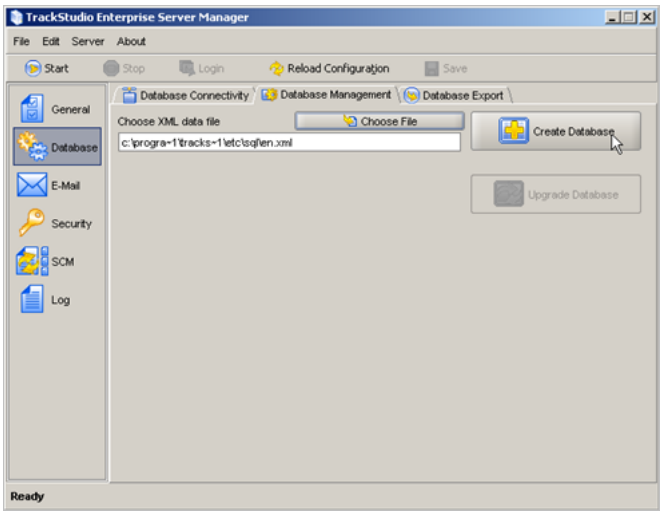
1. Stop TrackStudio, if it is running.
2. Start Server Manager (available in TrackStudio SA only).
3. Select the **Database -> Database Connectivity** tab.
4. Enter the JDBC connection properties.
5. Select the **General** tab.
6. Specify the character encoding of the database.
7. Select the **Database -> Database Export** tab.
8. Specify the name of the file to which the data will be exported.
9. Mark the **Anonymize data** checkbox to remove sensitive information from export file you send to TrackStudio Support.
10. 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 -> Database Connectivity** tab.
4. Enter the JDBC connection properties of the empty database (without any tables).
5. Select the **General** tab.
6. Specify the character encoding of the database.
7. Select the **Database -> Database Management** tab.
8. Specify the name of the imported XML file in the **Choose XML data file** 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.

2.6 Configuring TrackStudio Cluster

The following topic describes how to configure TrackStudio to be launched on several application servers grouped into a cluster.

Description

For 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 a cache for data processing. The cache stores 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 update the objects in 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 all nodes. Set **trackstudio.cluster** to **yes** and specify the same **trackstudio.cluster.name** for all cluster nodes. Set cluster node IP address if required.
3. Specify the name of the directory which will be used to store uploaded files. All instances must use the same directory in which to store uploads. You can use shared disk in Windows or NFS in UNIX.
4. Specify the name of the directory which will be used to store full text search index files. Each instance must have a local copy of index**.**
5. Configure database connections. All instances must use the same database.
6. Configure the other settings and launch TrackStudio on all cluster nodes. While loading, TrackStudio displays messages about active nodes in the cluster:

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
GMS: address is TMK-12X3:4390
-----------------------------------------------------
```

2.7 Integrating IDE and TrackStudio

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

Description

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

2.7.1 Installing IDEA Plug-in

The following 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** into this directory.
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** window).

2.7.2 Installing JBuilder Plug-in

The following 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 open an 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.

2.7.3 Installing Eclipse Plug-in

The following 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.

2.8 Integrating SCM and TrackStudio

The following topic describes how to integrate TrackStudio with SCM tools.

Description

TrackStudio supports integration with both of CVS and Subversion version control systems. SCM integration lets one see SCM commit information relevant to each task and browse last commits for your repository. You can assign multiple different SCM repositories for each of your projects.

To assign a repository connection for your project:

TrackStudio WAR:

1. Open a **trackstudio.scm.properties** file
2. Set **trackstudio.scm.useSCM** property.
3. For each of your projects create **trackstudio.scm.<name>.tasknum**, where tasknum- task number of the your project. SCM integration will be available for subtasks of the specified task.

4. Fill in other properties.

TrackStudio SA:

1. Start TrackStudio Server Manager.
2. Open the **SCM** tab.
3. Mark the **Enable SCM integration** checkbox.
4. Click the **Add** button.
5. Enter the **Connection name**.
6. Enter the **Task #** of your project. SCM integration will be available for subtasks of the specified task.
7. Select the repository **Type**.
8. Enter the **Root** of the repository.
9. Enter **Login** and **Password**.
10. **CVS only:** Specify **Module** and **Character encoding**.
11. Click the **Test Connection** button.
12. Click the **OK** button to save connection settings.

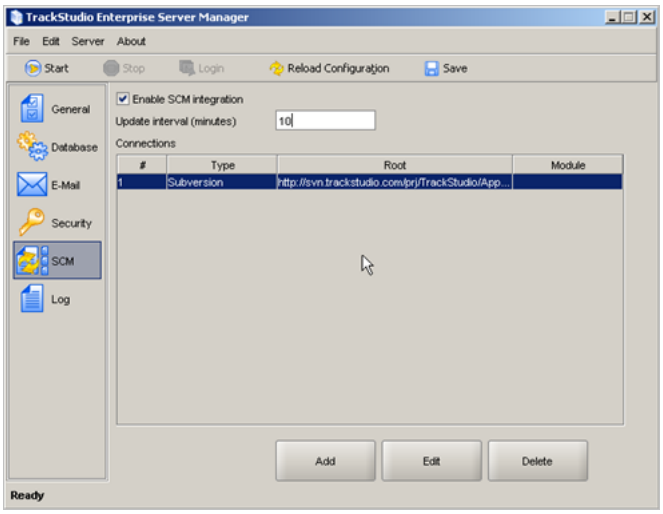
To import CVS/Subversion message into TrackStudio commit your files. Message will be added to the tasks specified in the message body.

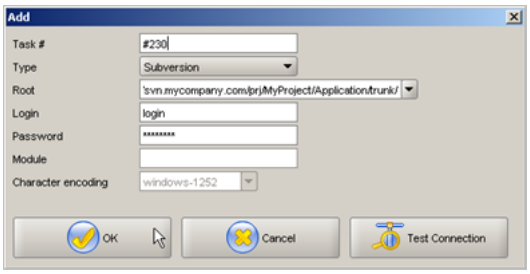
Subversion:

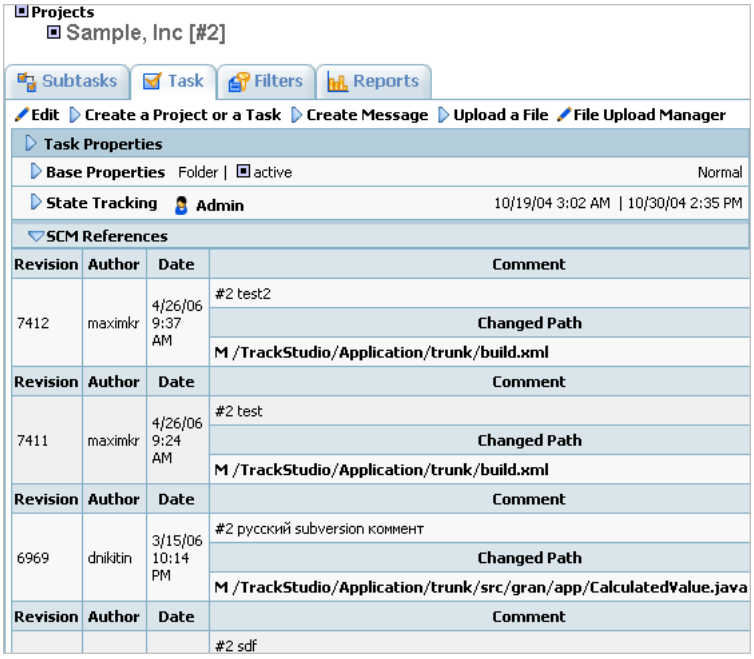
```
$ svn commit -m "This message should be added to the task #1 and #2."
```

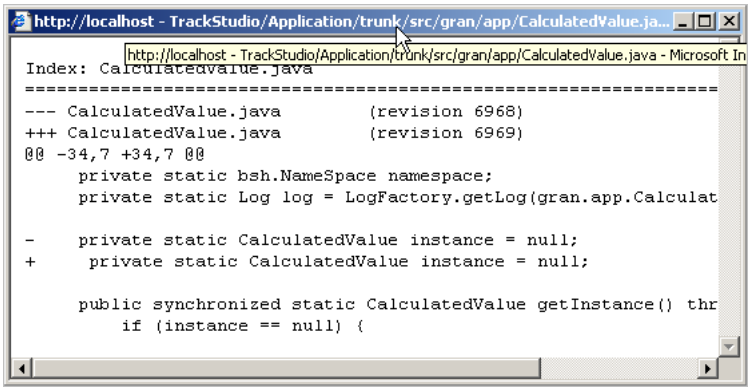
CVS:

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









2.9 Integrating E-mail Client and TrackStudio

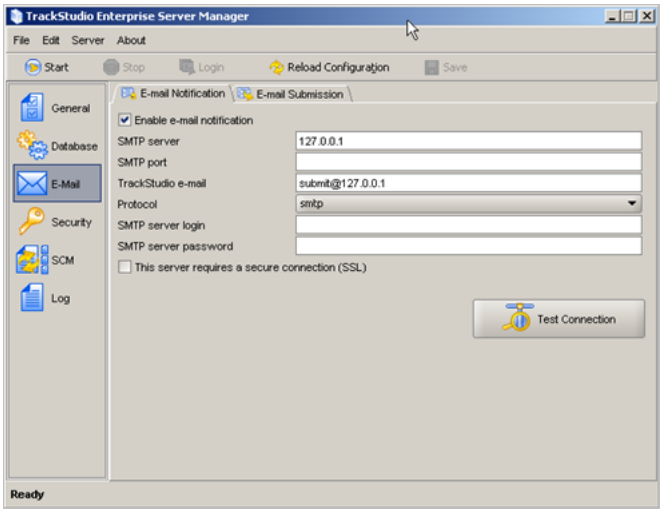
2.9.1 Configuring E-mail Notification

The following 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 for your users using the **Current User -> User... -> Edit** link. Do not use TrackStudio e-mail specified in **trackstudio.mail.properties** as user e-mail.
3. Create the e-mail notification rule using the **Current Task -> E-mail Notification Rules...** menu item. The rule determines which tasks send e-mail notifications, while the current task determines for which project the e-mail notification is enabled.



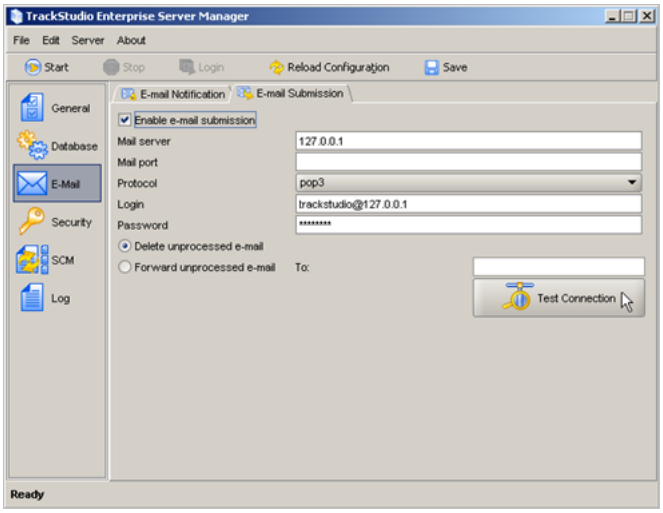
2.9.2 Configuring E-mail Submission

The following 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. **Optional:** Create accounts for the users you wish to have use this submission.
3. **Optional:** Create an e-mail submission rule to create new tasks by e-mail.



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. Below 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 for the software or visit the vendor sites.

Mozilla Messenger

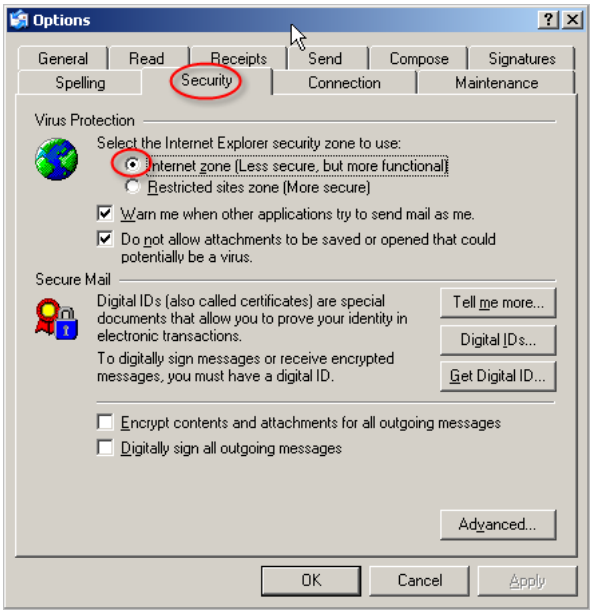
1. From the menu bar, choose **Edit**, then **Preferences.**
2. Select **Advanced** from the list of options, then **Scripts & Plugins.**
3. Click the box **Enable JavaScript for Mail and News.**

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 cause a worse problem if enabled. 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 cannot 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 130)
- Adding a Message by E-mail (see page 131)

2.9.3 Using JES for E-mail Integration

The following topic describes how to configure e-mail notification and e-mail submission for working with the 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 a 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. Uncomment 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 -> E-mail Notification** tab in Server Manager.
5. Check the **Enable e-mail notification** check box. Do not change any other parameters.
6. Click the **Start** button to run TrackStudio.
7. Log in.
8. Specify your e-mail address in the user settings (the **Current User -> User... -> Edit** link).
9. Use the **Current Task -> E-mail-Notification Rules...** menu item to create an e-mail notification rule for a user.

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 test it to make sure it works properly.
2. Run Server Manager (**sman**).
3. Go to the **E-mail -> E-mail Submission** tab in Server Manager.

4. Check the **Enable e-mail submission** check box. Do not change any other 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 on to its intended recipient.
7. Use the **Current Task -> E-mail Import Rules...** menu item to configure e-mail submission rules for a task.
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 126)
- Adding a Task by E-mail (see page 130)

2.10 Integrating Serence KlipFolio and TrackStudio

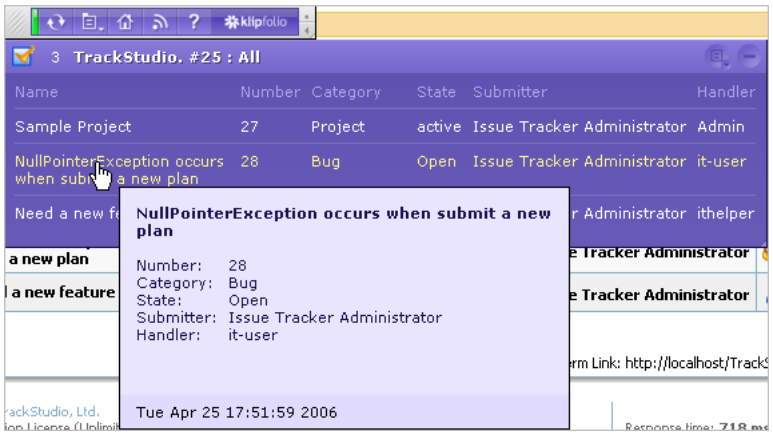
The following topic describes how to simplify tasks monitoring by integrating TrackStudio and Serence KlipFolio.

Description

To simplify tasks monitoring, TrackStudio can be integrated with Serence KlipFolio personal dashboard. The KlipFolio provides different kinds of alerts such as toast (Windows tray pop-ups that retract), sounds, mail, etc.

To monitor subtasks in Serence KlipFolio:

1. Run the Serence KlipFolio.
2. Select a task for which subtasks should be monitored.
3. Click the **Reports** tab.
4. Create a new **List** report.
5. Select the **KlipFolio Format**.
6. Click the **Generate Report** button. TrackStudio creates a Klip and opens it with Serence KlipFolio.



To configure alerts in KlipFolio:

1. Right click the Klip and then click the **Klip Setup** item in the popup menu.
2. Use the **Alerts** tab to configure the parameters of the alerts.



See Also

- Serence KlipFolio

2.11 TrackStudio Configuration Properties

The following section describes the TrackStudio configuration files.

Description

TrackStudio look for configuration files in the following order:

- **WEB-INF** subdirectory in **TrackStudio.war.** Use the **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 the **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 TrackStudio, the following property files are loaded at startup:

File	Server Manager tab	Description
trackstudio.properties	**General**	A main configuration file.
trackstudio.hibernate.properties	**Database**	Database connection configuration file.
trackstudio.mail.properties	**E-mail**	E-mail notification and e-mail submission configuration file.
trackstudio.security.properties	**Security**	LDAP & NTLM properties and security policy configuration.
trackstudio.scm.properties	**SCM**	Source Code Management systems integration.
trackstudio.log4j.properties	**N/A**	Debug logging properties.

trackstudio.license.properties	**N/A**	TrackStudio license file. Do not modify this file.
trackstudio.adapters.properties	**N/A**	TrackStudio adapters configuration. Do not modify this file.

You should restart TrackStudio after configuration files modification.

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.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.cluster.bind	**Specify network interface (checkbox)**	Mark to bind cluster node to specific network interface.	*yes* *no*
trackstudio.cluster.bindTo	**Specify network interface (dropdown)**	TrackStudio cluster node IP address.	*192.168.1.100*
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.skinPath	**N/A**	Skin path.	*/skins/defaultSkin*
trackstudio.defaultLocale	**Default locale**	Default locale. Used before user is logged in (Login/Registration/Forgot password pages).	*en*

trackstudio.script	**N/A**	Import packages for scripts.	*java.lang.Boolean*
trackstudio.maxDescriptionLength	**N/A**	Maximum task/message description length.	*60000*
trackstudio.startupDelay	**N/A**	Startup delay in seconds. Used to wait for DBMS startup.	*0*
trackstudio.gzipHTTP	**Compress HTTP content**	Compress HTTP content.	*yes* *no*

trackstudio.hibernate.properties

Property	Server Manager (Database tab)	Description	Example
hibernate.dialect	**Select DBMS**	SQL dialect.	*org.hibernate.dialect.HSQLDialect*
hibernate.connection.url	**URL**	JDBC connection URL.	*jdbc:hsqldb:file:test;shutdown=true*
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 tab)	Description	Example
trackstudio.sendMail	**Enable e-mail notification**	Enable e-mail 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 e-mail 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 any 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.com*
mail.debug	**N/A**	SMTP/POP3/IMAP debug logs.	*true* *false*

trackstudio.security.properties

Property	Server Manager (Security tab)	Description	Example
trackstudio.security.password.min	**Minimum password length**	Minimum password length.	*0*
trackstudio.security.password.maxage	**Maximum password age (days)**	Maximum password age in days. 0 means "unlimited".	*0*
trackstudio.security.password.complex	**Password must meet complexity requirements**	Enforce password complexity requirements to prevent dictionary attack.	*0*
trackstudio.security.password.history	**Enforce password history**	Amount of remembered passwords to prevent password reuse by users. 0 means "don't check", maximum value - 8.	*0*
trackstudio.security.password.changefirst	**User must change password at first login**	User must change password at first login.	*yes* *no*
trackstudio.security.password.case	**Use case insensitive account names**	Use case insensitive account names (logins).	*yes* *no*
trackstudio.loginAsAnotherUser	**Enable logon as another user**	Allowed login as another user. To login as subordinate user, use their name and your password.	*yes* *no*
trackstudio.hideSessionId	**Hide session ID from URL**	Set to store the session ID in the cookie.	*yes* *no*
trackstudio.useNTLM	**Use NTLM Authorization**	Use NTLM Authorization.	*yes* *no*

jcifs.netbios.wins	**WINS server address**	WINS server address for name resolution.	*127.0.0.1*
jcifs.smb.client.domain	**Domain name**	Domain name.	*WORKGROUP*
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 user authentication.	*cn=users,dc=ldap-server,* *dc=my-company, dc=com*
ldap.userDN	**User DN**	Specifies the user DN, which is connected to the LDAP server. Objects (users, groups, and 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-server,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.	*name* *login*
ldap.loginAttrLDAP	**Authorize by LDAP properties**	Specifies the property which should be used to search for the user on the LDAP server. For example, if the **ldap.loginAttrLDAP** is *cn*, the common name is used to search for the user.	*displayName* *sAMAccountName*

trackstudio.scm.properties

Property	Server Manager (SCM tab)	Description	Example
trackstudio.scm.useSCM	**Enable SCM integration**		*yes* *no*

trackstudio.scm.updateInterval	**Update interval (minutes)**	Update interval (minutes)	*10*
trackstudio.scm.<name>.type	**Type**	SCM type	*Subversion* *CVS*
trackstudio.scm.<name>.tasknum	#	Task number. SCM integration will be available for subtasks of the specified task.	2 5
trackstudio.scm.<name>.root	**Root**	SCM root	*http\://svn.mycompany.com/prj/MyProject/Application/trunk/*
trackstudio.scm.<name>.login	**Login**	Login	
trackstudio.scm.<name>.password	**Password**	Password	
trackstudio.scm.<name>.module	**Module**	Module (CVS only)	project
trackstudio.scm.<name>.encoding	**Character encoding**	Character encoding (CVS only)	UTF-8

2.12 Changing the TrackStudio URL

The following topic describes how to change the TrackStudio URL in TrackStudio SA

Description

To change the TrackStudio URL in TrackStudio SA:

1. Open the **etc/jetty.xml.**
2. Find the **Configure the Contexts** section and add script into this section that maps TrackStudio URL to site content as shown below:

```
  <!-- ============================================================= -->
  <!-- Configure the Contexts                                        -->
  <!-- ============================================================= -->
<Call name="addWebApplication">
    <Arg>/NewTrackStudioURL</Arg>
    <Arg>./webapps/TrackStudio</Arg>
    <Set name="defaultsDescriptor">org/mortbay/jetty/servlet/webdefault.xml</Set>
</Call>
```

2.13 Adding your Web Site to the TrackStudio Server

The following topic describes how to add your own web site to the TrackStudio SA web server (jetty).

Description

To add your own web site to the TrackStudio SA web server:

1. Create a directory like *webapps/MySite.*
2. Add your web site content into this directory.

3. Use a URL such as *http://localhost:8888/MySite* to access your site.

3 User's Guide

3.1 Implementation Guide

The following topic describes how to configure TrackStudio.

Description

We recommend that you complete the following steps to use TrackStudio for issue tracking:

Step	How to
1. Login as **itadmin/itadmin**.	
2. Use the **Current User -> Users List... -> Create a User** pulldown to create user accounts for your developers, testers, and customers.	Creating a User Account (see page 81)
3. Use the **Current Task -> Subtasks... -> Create a Project or a Task** pulldown to create your project hierarchy.	Creating a Project (see page 59)
4. Login as **root/root**.	
5. Use the **Current User -> User... -> Change Password** link to change the **root's** password.	Changing a Password (see page 88)
6. Use the **Current User -> Users List...** menu item to choose jsmith's account and use the **Edit** link to deactivate it (all subordinated accounts will be deactivated automatically).	Locking a User Account (see page 89)

We recommend that you complete the following steps to create brand new TrackStudio configuration for your company:

Step	How to
1. Login as **root/root.**	
2. Use the **Current User -> Users List... -> Create a User** pulldown to create a managed administrator.	Creating a User Account (see page 81)
3. Click the **Change Password** link to set the password for the managed administrator.	Changing a Password (see page 88)
4. Use the **Current Task -> Subtasks... -> Create a Project or a Task** pulldown to create the root folder for your projects.	Creating a Project (see page 59)
5. Use the **Current Task -> Access Control Rules... -> Assigned Statuses -> Grant Access** pulldown to give the managed administrator permission to manage your projects.	Granting Users Access to a Project (see page 83)
6. Login as the managed administrator.	

7. Use the **Current User -> Statuses... -> Create a Status** pulldown to create user groups and set permissions for each user group.	Establishing a User Group Account (see page 78)
8. Use the **Current User -> Users List... -> Create a User** pulldown to create user accounts for your developers, testers, and customers.	Creating a User Account (see page 81)
9. Use the **Current Task -> Workflows... -> Create a Workflow** link to create workflows for your projects and bugs. Set workflow permissions for your user statuses.	Creating a Workflow (see page 120)
10. Use the **Current Task -> Categories... -> Create a Category** link to create categories for your projects and tasks. Set category permissions for your user statuses.	Creating a Category (see page 124)
11. Use the **Current Task -> Subtasks... -> Create a Project or a Task** pulldown to create your project hierarchy.	Creating a Project (see page 59)
12. Use the **Current Task -> Access Control Rules... -> Assigned Statuses -> Grant Access** pulldown to allow your users to view projects and submit bugs.	Granting Users Access to a Project (see page 83)
13. Use the **Current Task -> Custom Fields... -> Create a Custom Field** and the **Current Task -> Workflows -> Custom Fields -> Create a Custom Field** pulldowns to create custom fields for your projects and bugs.	Adding a Custom Field (see page 94)
14. Use the **Current Task -> Filters... -> Create a Filter** link to customize task filters.	Filtering Subtasks by Properties (see page 116)
15. Login as **root/root**.	
16. Use the **Current User -> User... -> Change Password** link to change the **root's** password.	Changing a Password (see page 88)
17. Use the **Current User -> Users List...** menu item to choose jsmith's account and use the **Edit** link to deactivate it (all subordinated accounts will be deactivated automatically).	Locking a User Account (see page 89)
18. 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 72)

3.2 Sample Project Overview

The following topic describes the TrackStudio demo project **Sample, Inc**.

Description

Organizational Structure

Suppose we have a company - *Sample Inc.* - with the following functional structure:

- *R&D Department*
- *QA Department*
- *Customer Support Department*

- *Project Management Department*
- *Sales Department*

Employees in the departments have at least two managers (i.e. their line manager and the department manager). Each department manager is only responsible for his or her own employee. Each line manager is only responsible for his or her 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*

Sample, Inc. employees, their login ID and password (the same), and user status (group):

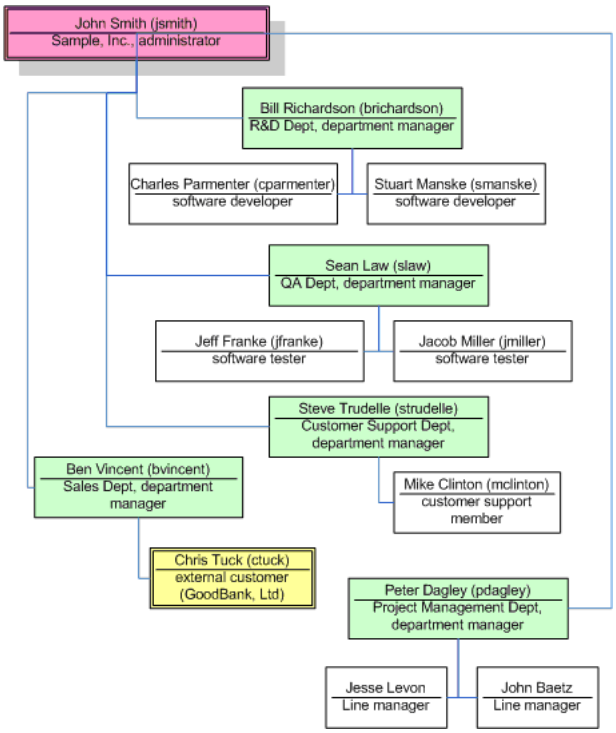
- *John Smith* - jsmith - administrator
  - *Peter Dagley* - pdagley - Project Management Department Manager
    - *John Baetz* - jbaetz - Line Manager for *YTracker*
    - *Jesse Levon* - jlevon - Line Manager for *XWare*
  - *Steve Trudelle* - strudelle - Customer Support Department Manager
    - *Mike 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:

- an additional custom field to hold the customer addresses. You can confirm this by logging in as *bvincent* (password *bvincent*) and opening the menu **Current User -> Custom Fields...**
- a customer self-registration rule, to allow all new customers to create bugs in the *Customer Support* project. Confirm this by opening the menu **Current User -> Self-registration Rules...**, while logged in as *Ben Vincent*.

Samples, Inc also has an existing customer - *Chris Tuck*, from *GoodBank*. He can login as *ctuck/ctuck* and add a bug report into the system. In addition, he can create subordinate users from his company to add bug reports. You can limit how many accounts each customer can create, and set expiration dates for your customer accounts, to match the validity periods listed in your support contracts (for example, one year).

A user's password in the demo database is the same as his/her login. Login as *root/root* or *jsmith/jsmith* to examine the demo database.



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 yet.

YTracker Team

User Name	Role (User Status)
John Baetz	*project manager*
Jeff Franke	*software tester*
Stuart Manske	*software developer*

XWare Team

User Name	Role (User Status)
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.

Group Permissions

Now let's examine how user group permissions are defined. Login as *jsmith* and open the **Current User -> Statuses...** menu item. Using user groups, you can control every menu item and button the users in that group can view or use, and set which fields can be viewed or edited. 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, with one workflow state and 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-based 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 team is waiting for a response from customer.

Take a closer look at the permissions defined for the *Issue Lifecycle* workflow:

Message Type	Description	Permissions
001 Resolve	Resolve a bug	• Any administrator, department manager, line manager, or customer support member can *resolve* tasks. • Testers or developers can *resolve* tasks (i.e. use the *resolve* message) only when they are the task handler. • External customers cannot see *resolve* messages from team members or add such messages.
002 Verify	Verify a bug	• Any administrator, department manager, or line manager can *verify* tasks (i.e. use *verify* messages). • Software testers can *verify* tasks only when they are the task handler. • Support staff, developers, and customers cannot add *verify* messages. • Customers cannot see *verify* messages.
003 Close	Close a bug	• Any users except software developers, testers, and customers can *close* tasks. • Customers cannot see *close* messages from other users.
004 Reopen	Reopen a bug	• Any users except software developers, testers or customers can *close* tasks. • Customers cannot see *close* messages from other users.
000 Note	Add a note	• All 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 a user creates a bug, he or she does not choose a workflow directly. Instead, the user will choose task categories such as *Software Bug* that are connected to a given workflow. Several categories can be connected to the same workflow, and, for each category you can specify who can create, view, edit or delete tasks within the category. In addition, you can specify the category's dependencies - for example, *Folder* can contain *Software Bug*, but *Software Bug* cannot contain *Folder*.

Assigning Employees to Products

In our demo, product managers cannot assign developers or testers directly to projects - they need approval from department managers. You can see this in task #11, where *YTracker's* project manager, *John Baetz*, asks the R&D department manager, *Bill Richardson*, about being assigned one developer for his project. *Bill* approves it, and assigns *Stuart Manske* to *YTracker* as a software developer. Confirm this in the **Current Task -> Access Control Rules...** menu item, and click the **Assigned Statuses** tab for *YTracker* #7. Department managers can use the *My Assignment Requests* filter to find their assignment requests.

Using Calculated Custom Fields

In our implementation for this demo, 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 the *Company* field is a user field, it cannot be referenced by a task filter directly. However, we can resolve this by creating a task custom field that uses a script which will return the *Company* field.

Login as *jsmith/jsmith* and go to the **Current User -> Scripts...** menu item. Look at the *getDepartment* script, which returns department names when the task submitter is you or your subordinate user, or *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 have connected the *getDepartment* script to the *Submitter Department* task custom field (see the **Current Task -> Custom Fields...** menu item for 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 within each state 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. Login as an administrator, department manager and software tester, and compare the output of this report for each.

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

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.

3.3 Concepts

The following topic describes TrackStudio Concepts.

Description

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

TrackStudio is a hierarchical database of objects. There are 17 main types of object that you need to be aware of:

- Tasks (items)
- Users
- Workflows
- Messages
- Categories
- User groups (also called user "statuses")
- Filters
- Reports
- Custom fields
- Triggers
- E-mail templates
- Scripts
- Self-registration rules
- Access control rules
- E-mail notification rules
- Filter subscription rules
- E-mail import rules

In brief, tasks (items) are objects whose state is tracked and updated by users of the system. Every task is of a particular category, and categories are objects such as *Issues*, *Risks*, *Incidents*, *Software Bugs* and so on. When you create a category, you assign it a workflow, and therefore categories are a kind of "selectable workflow instance" which you make available in the TrackStudio folder structure for selection by users. This is where TrackStudio becomes an extremely powerful management tool. The workflow behind the category defines the states that are allowed for a task (item), the task's transitions between states, what type of user can transition the item to another state, and the custom fields that can be captured for the task.

Users are grouped into statuses which can be described as a typical role grouping. Statuses are used to control the

permissions users have in the system. The administrator status is built-in, and you can create others to assign permissions to. Access control rules govern what user or status can do what, within a given task.

In terms of interactive control of TrackStudio, filters are a key component. Filters can be created which return lists of tasks and subtasks based on a set of criteria, and might then be selected to return lists of "all open issues" or "my open issues" for example, that might also be fed in part by custom scripts. Filters also feed and control important components of TrackStudio such as e-mail notifications or subscriptions, reports, and even full-text search.

3.3.1 User Interface Concepts

The following topic describes basic user interface concepts.

Description

1.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. To synchronize the **Navigation Tree** with the current task or user in the main frame, use the small icon near the TrackStudio logo, on the left.

For tasks, you can see the task name and task number, and the number of subtasks. For users you can see the user name, and the number of subordinated users.

2.Main Menu

Use the main menu to manage projects and tasks, build reports, etc.

Use the input field in the right part of the **Main Menu** to:

- jump to a task by its **number, alias** or **content**.
- jump to a user by the user **login ID**.

3.Full Path

Nearly all operations in TrackStudio are performed against the current task or user. The **Full Path** field under the **Main Menu** shows the path to the current task or user, and just below the field, some information related to the current user or task, depending on the working mode.

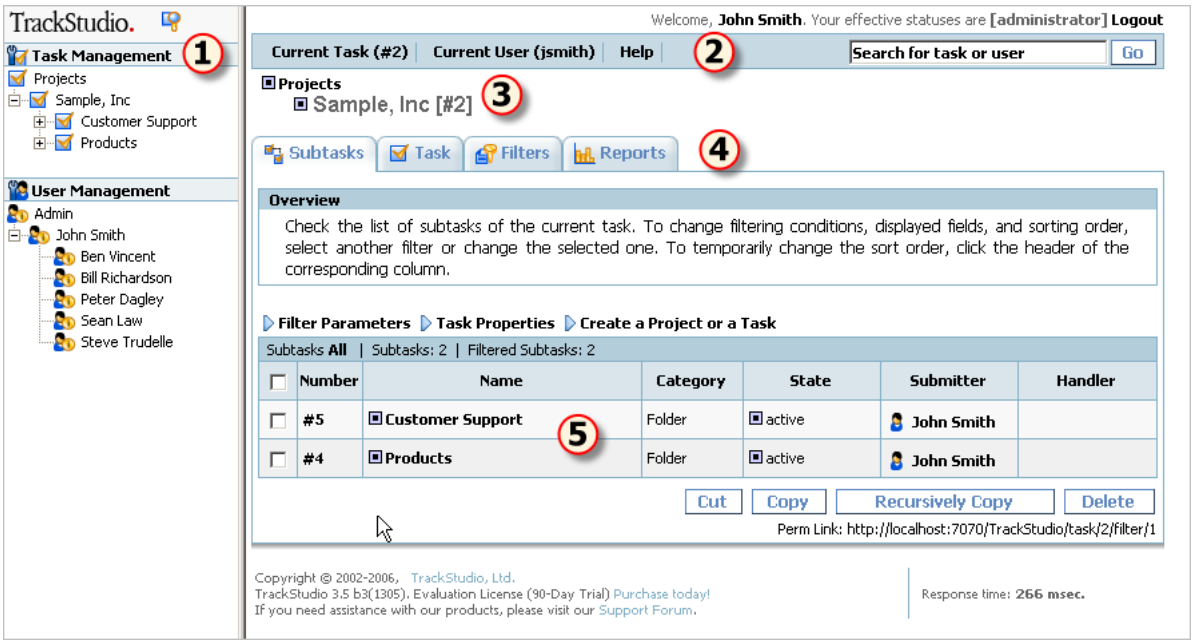
4.Tabs

TrackStudio pages with common functionality are grouped in a tabbed interface. You can control which tabs are accessible for each of your users.

5.Object lists

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

- To create an object, use the corresponding pulldown or link above the objects list.
- To view or edit object properties, click the object name.
- To delete an object, select the checkbox beside it and click the **Delete** button.



3.3.2 Task Concepts

A "task" in TrackStudio is a generic description for an object to be tracked, such as projects, bugs, defects, versions, components, modules and so on, and allow the state of those objects to be tracked.

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* contain 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 tasks, 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 operating system, etc.
- You can make individual copies of the task for each operating system. 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 sub-versions of each bug (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 bug (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).

3.3.3 User Concepts

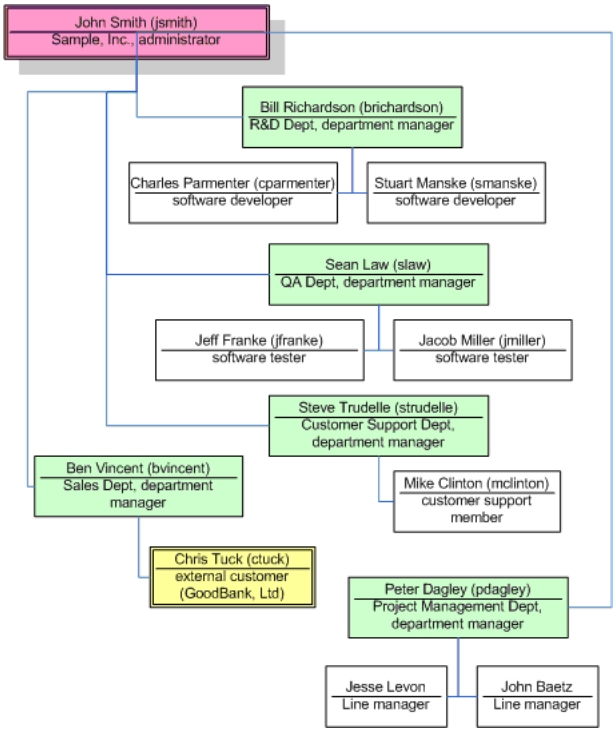
A “user” is a user of TrackStudio, with a logon id, password and certain access rights. Use users to manage your employees, departments and customers.

Description

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



3.3.4 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 allows you to set the rules for changing task states, and contains a set of priorities, states message types, transitions, and custom fields.

Priorities

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

States

A state is the position in a workflow where a given task currently resides. While a task is in a certain state, it corresponds to a handler, who is the user responsible for this stage in the process of working with this task. This handler user must do his work before changing the state 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-Based 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 associated with the specific workflow.

3.3.5 Message Concepts

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

Description

Use messages to

- Change the task state. To do this, select the Message Type that will change the task from its current state into the one you need. Task states and transitions between them depend on the task workflow, which is in turn 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 to work on a task (for example, you have code that needs to be tested or you need to ask another developer a question), specify the appropriate user for that task in the Handler field.
- Monitoring the progress of 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.

3.3.6 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 task types by creating a link between tasks in a particular area of the hierarchy, and workflows. When creating a category, you can specify its possible subcategories and the user groups who can create, view 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.

3.3.7 User Status Concepts

Use user status to assign permissions to users.

Description

User statuses represent user groups, and each status can have a specific set of privileges. A user can be included in several different groups, and the user status set can be different for different projects.

The list of **effective statuses** for a task is determined on the basis of the user's **own default statuses**, and any **assigned statuses**. To perform a particular action, the action should be permitted for at least one of the user's effective statuses. You can set users' default statuses when creating user accounts, or, you can use the **Current User -> User... -> Edit** page to edit an existing user's default status. Use the **Current Task -> Access Control Rules... -> Assigned Statuses** tab to assign statuses for tasks and use the **Current User -> Access Control Rules... -> Assigned Statuses** tab to assign statuses for users.

To configure what tabs users can view, use the **Current User -> Statuses...** menu item.

- To configure what buttons and menu items users can press, use the **Current User -> Statuses...** menu item.
- To configure what standard task and user fields users can view and edit, use the **Current User -> Statuses...** menu item.
- To configure what custom fields users can view and edit, use the **Current Task -> Custom Fields... -> Permissions** or **Current User -> Custom Fields... -> Permissions** tab.
- For workflow-based custom fields you can specify user accounts which can view and edit custom field values in the **Add Message** pulldown. Click the **Current Task -> Workflows...** menu item, choose custom field and use the **Message Type Permissions** tab to configure it.
- To configure what tasks users can create, view, modify and delete, use the **Current Task -> Categories... -> Permissions** tab.
- To configure what messages users can view and add, click the **Current Task -> Workflows...** menu item, choose message type and use the **Permissions** tab.

3.3.8 Filter Concepts

Use filters for filtering users, tasks and messages, setting 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 if you have no access to the task.

3.3.9 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 Time**) users 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, RTF, plain text or Microsoft Excel format.

3.3.10 Custom Field Concepts

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

Description

TrackStudio supports user-based, task-based and workflow-based custom fields:

- User-based custom fields are available for the user they are created for, and for subordinated users.
- Task-based custom fields are available for the task they are created for, and for all subtasks.
- Workflow-based custom fields are available for tasks using a specific workflow only.

Users can populate custom field values manually, or they can be calculated dynamically using **Task / Custom Field Value** and **User / Custom Field Value** scripts. Use calculated custom fields to evaluate complex filter conditions. TrackStudio can calculate custom field values each time it accesses the task, or, can cache values and reset them when the task changes. Do not cache custom fields that access other tasks to evaluate its value, since TrackStudio cannot recalculate custom field values in this case. Do not use calculated custom fields to execute scripts when tasks change - use triggers instead.

If you add non-calculated custom fields to TrackStudio after tasks have been created and later in their life cycle, such tasks will contain no values for the custom fields even if a default is defined.

Use scripts of **Task / Custom Field Lookup** and **User / Custom Field Lookup** types to create a list of possible values for custom fields of the **String** type.

To configure what custom fields users can view and edit, use the **Current Task -> Custom Fields... -> Permissions** or **Current User -> Custom Fields... -> Permissions** tab. For workflow-based custom fields you can specify users which can view and edit custom field value in the **Add Message** pulldown. Click the **Current Task -> Workflows...** menu item, choose custom field and use the **Message Type Permissions** tab to configure it.

See Also

- Script Concepts (see page 54)
- Adding a Custom Field (see page 94)
- Calculating a Custom Field Value (see page 104)
- Custom Field Properties (see page 136)

3.3.11 Script Concepts

Use scripts to calculate custom field values, define triggers, define CSV import rules and define **String** custom field lookup values.

Description

TrackStudio uses a 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 use in your scripts classes defined in the **trackstudio.script** property in the **trackstudio.properties** file.

The following constants are also available:

Constant	Type	Value	Description
DAYS	long	86400000	msec/day
HOURS	long	3600000	msec/hour
MINUTES	long	60000	msec/minute
SECONDS	long	1000	msec/second

Task / Custom Field Value and User / Custom Field Value:

Use scripts of these types to calculate the values of custom fields. To create a calculated custom field, you should create a custom field and then specify a script for it. No results are saved to the database for a calculated field.

An object of the **SecuredTaskBean** class available in the **task** variable can be used in a script of the **Task / Custom Field Value** type. This object corresponds to the task the custom field value is calculated for. Use this object to access task properties.

An object of the **SecuredUserBean** class available in the **user** variable can be used in a script of the **User / Custom Field Value** type.

The type of the value returned by the script depends on the custom field type. 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 make sure that the result matches the required type. For example, to get the result of date calculation in milliseconds (long), convert it to the **Date** type:

```
new Date(milliseconds)
```

Task / Custom Field Lookup and User / Custom Field Lookup

Use scripts of these types to create a list of possible values for custom fields of the **String** type.

An object of the **SecuredTaskBean** class available in the **task** variable can be used in a script of the **Task / Custom Field Lookup** type, while an object of the **SecuredUserBean** class available in the user variable can be used in a script of the **User / Custom Field Lookup** type. The script must return a list of strings that will be offered to the user specifying a value for a custom field of the **String** type:

```
List list = new ArrayList();
list.add("value1");
list.add("value2");
return list;
```

Trigger / *

Triggers are special types of scripts that are defined to execute automatically before, in place of or after data modifications. They can be executed automatically on the **Create Task**, **Add Message**, **Update Task** triggering actions.

There are three different types of triggers in TrackStudio. They are **BEFORE** triggers, **INSTEAD OF** triggers and **AFTER** triggers. These triggers differ from each other in terms of their purpose and when they are fired.

- Use **BEFORE** triggers for data validation and correction before editing existing tasks, or adding new tasks or messages. **BEFORE** triggers execute before the triggering action.
- Use **INSTEAD OF** triggers to avoid new task or message creation or task updating. **INSTEAD OF** trigger replaces the normal triggering action with the actions defined in the trigger. For example, if an **Add Message / INSTEAD OF** trigger exists and a message is added, TrackStudio will not add a message to the task, but rather execute the trigger, which may or may not add a message to the task.
- Use **AFTER** triggers to generate change history, move tasks to another project after editing existing tasks, or adding new tasks or messages. **AFTER** triggers execute following the triggering action. You cannot change properties of the created or edited object using the **AFTER** trigger.

You can define a **BEFORE**, an **INSTEAD OF** and an **AFTER** trigger on the same object for the same operation.

Trigger Type	Parameter	Returns
Trigger / Create Task / *	**SecuredTaskTriggerBean** task	**SecuredTaskTriggerBean**
Trigger / Edit Task / *	**SecuredTaskTriggerBean** task	**SecuredTaskTriggerBean**
Trigger / Add Message / *	**SecuredMessageTriggerBean** message	**SecuredMessageTriggerBean**

To notify user about invalid data submission, throw the **UserMessageException** exception. In this case, event processing will be interrupted and user will be returned to the initial data entry form with error message.

CSV Import

Use a script of this type to import objects from a CSV file. Source data is stored in the **inputMap** object of the **Map** class. Each field in the header line is a key while the value is the corresponding field from the line being processed in the CSV file. This script must return **Map** or a collection of objects of the **Map** class. The corresponding TrackStudio object will be created for each object in this collection.

See Also

- Custom Field Concepts (see page 54)
- CSV Import Concepts (see page 58)

3.3.12 Self-registration Concepts

Use self-registration rules to allow users register in the system without the participation of the administrator.

Description

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

3.3.13 E-mail Template Concepts

Use e-mail templates to customize e-mail notification and filter subscription e-mails.

Description

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

3.3.14 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 subscriptions using the **Current Task -> Filter Subscription Rules...** menu item.

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 new file uploaded to the task.
- a message is added to the task.

Activate event-based e-mail notification on the **Current Task -> E-mail Notification Rules...** menu item. The e-mail notification will be sent when the task to which rule connected or one of its subtasks are modified. The email notification system checks the filtering conditions a bit different from the usual routine.

When you create a new task, update an existing one or upload a file:

The notification will be sent if the task meets the filtering parameters for tasks. If there are no filtering parameters specified, the e-mail notification will be sent when any task is modified. When a task is created or modified, filtering parameters 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**
- **Bulk Processing Tool**

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

When you create a new message:

TrackStudio checks whether the task meets task filtering conditions, or any 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, but 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 in user.
- When processing filter subscription rules, it means the subscribed user.
- When processing e-mail notification rules, it means the user who has created or modified the task, uploaded the file, or added a message.

3.3.15 E-mail Submission Concepts

Use e-mail submission to create new tasks, add messages to existing tasks and upload files by sending e-mails.

Description

If e-mail submission is enabled in **trackstudio.mail.properties**, TrackStudio checks checks the designated mailbox at regular intervals, retrieves any e-mails from the designated mailbox and tries to import it as a message or task.

How it works:

1. If the e-mail comes from a known user and contains the correct task number in the **Subject**, the system imports the e-mail as a new message with the default message type.
2. E-mail import rules are checked and evaluated one after the other, in the order specified by the **Order** rule property. If the e-mail meets the rule requirements, the system imports the e-mail as a new task.
3. 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.
4. If the e-mail cannot be imported, it is either deleted from the queue or forwarded to the specified e-mail address.

3.3.16 CSV Import Concepts

The following topic describes how to import data from a CSV file to TrackStudio.

Description

Use CSV import to import tasks, messages and users to TrackStudio. Before importing data, prepare a file with data in the CSV format and a script of the **CSV Import** type.

Source CSV File Requirements

Prepare a file with data in the Microsoft Excel CSV format. The CSV file must contain a header line. To get an example of a correct CSV file, export tasks into the CVS format using a report of the the **List** type.

CSV Import Script Requirements

The script is executed for each line of CSV file. Each line is parsed and stored in the **inputMap** object of the **Map** class. Each field in the header line is a key while the value is the corresponding field from the line being processed in the CSV file.

To create one TrackStudio object out of one line in a CSV file, the script must return an object of the **Map** class that describes the object being created. To specify the type of the object being created, place an item with the **CSVImport.OBJECT_TYPE** key and one of the following values into the collection:

Value	Created Object
CSVImport.TASK_TYPE	Task
CSVImport.MESSAGE_TYPE	Message
CSVImport.USER_TYPE	User

To create several TrackStudio objects, the script must return a collection of objects of the **Map** class. Objects will be imported in the order they are arranged in the collection.

Custom Field Data Import

To import the values of custom fields, specify an object of the **Map** class containing the {*Custom field ID*,*Custom field value*} pairs as the value of the **TASK_UDF_MAP/USER_UDF_MAP/MESSAGE_UDF_MAP** key. The custom field value must be specified as a line. To separate several values of custom fields of the **Multi List**, **User** and **Task** types, use ";" character.

Date Formatting

To convert a date from a **String** into an object of the **Calendar** class, use the **parseToCalendar()** method of the **DateFormatter** class. In case the date locale in the CSV file differs from the logged user's locale, specify the locale of the CSV file in the *locale* variable:

```
String locale = "fr_FR";
DateFormatter df = new DateFormatter(sc.getUser().getTimezone(), locale);
taskMap.put(CSVImport.TASK_SUBMIT_DATE, inputMap.get("Submit Date") != null ?
   df.parseToCalendar((String) inputMap.get("Submit Date")) : null);
```

3.4 Step-by-Step Guides

3.4.1 Managing Projects and Bugs

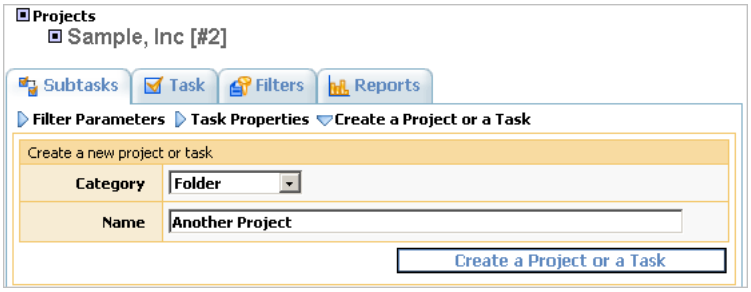
3.4.1.1 Creating a Project

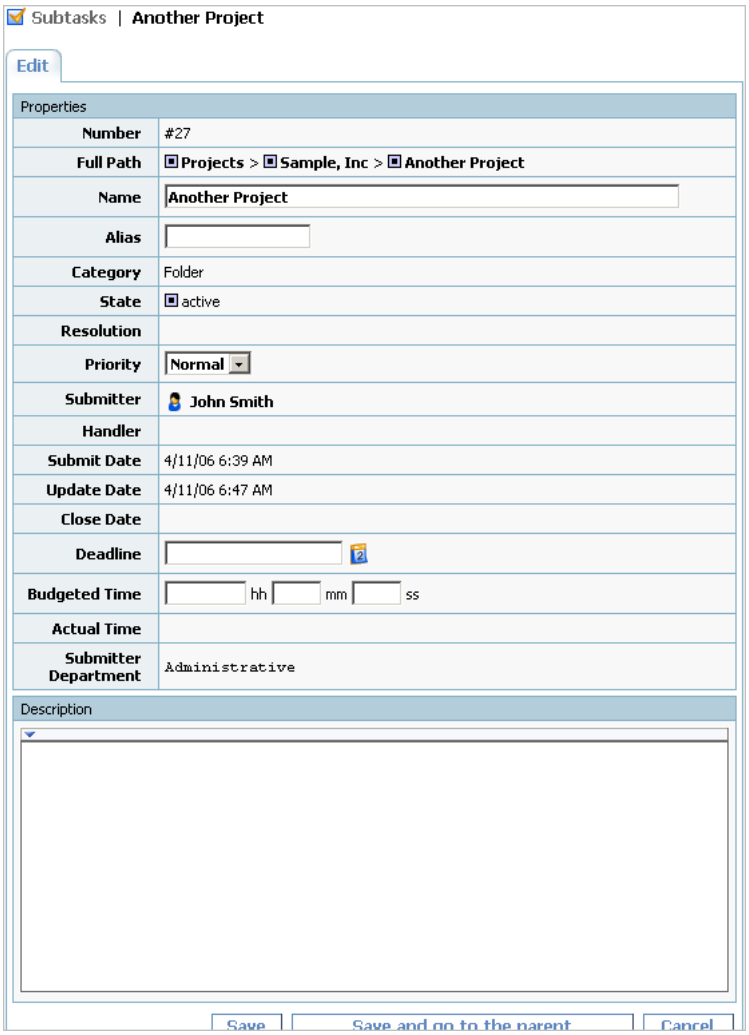
The following topic describes how to create a project.

Description

To create a project:

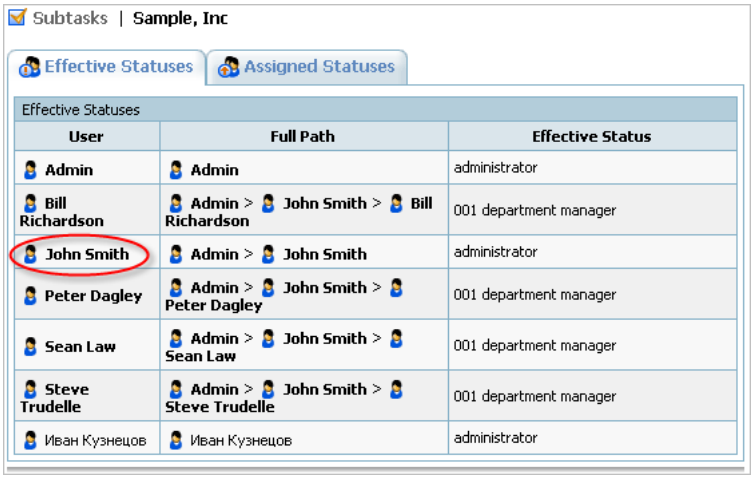
1. Choose a parent task for your project.
2. Click the **Subtasks** tab.
3. Expand the **Create a Project or a Task** pulldown control.
4. Select a category for the project that will allow you to create the subtasks you need in this category.
5. Enter the project name.
6. Click the **Create a Project or a Task** button.
7. Fill in project properties.
8. 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 permissions, use the **Current Task -> Access Control Rules...** menu item.



See Also

- Task Concepts (see page 50)
- Finding a Task (see page 115)
- Defining Category Dependency (see page 125)

- Defining Which Users Can Create, Edit or Delete Tasks (see page 101)
- Task Properties (see page 134)

3.4.1.2 Adding a Bug

The following topic describes how to add a bug or issue.

Description

To add a bug:

1. Choose a parent project for your bug.
2. Expand the **Create a Project or a Task** pulldown control.
3. Select the bug category.
4. Enter the bug summary.
5. Click the **Create a Project or 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 permissions, use the **Current Task -> Access Control Rules...** menu item.

See Also

- Task Concepts (see page 50)
- Finding a Task (see page 115)
- Defining Category Dependency (see page 125)
- Defining Which Users Can Create, Edit or Delete Tasks (see page 101)
- Task Properties (see page 134)

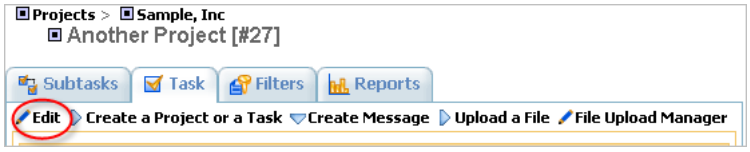
3.4.1.3 Editing Task Properties

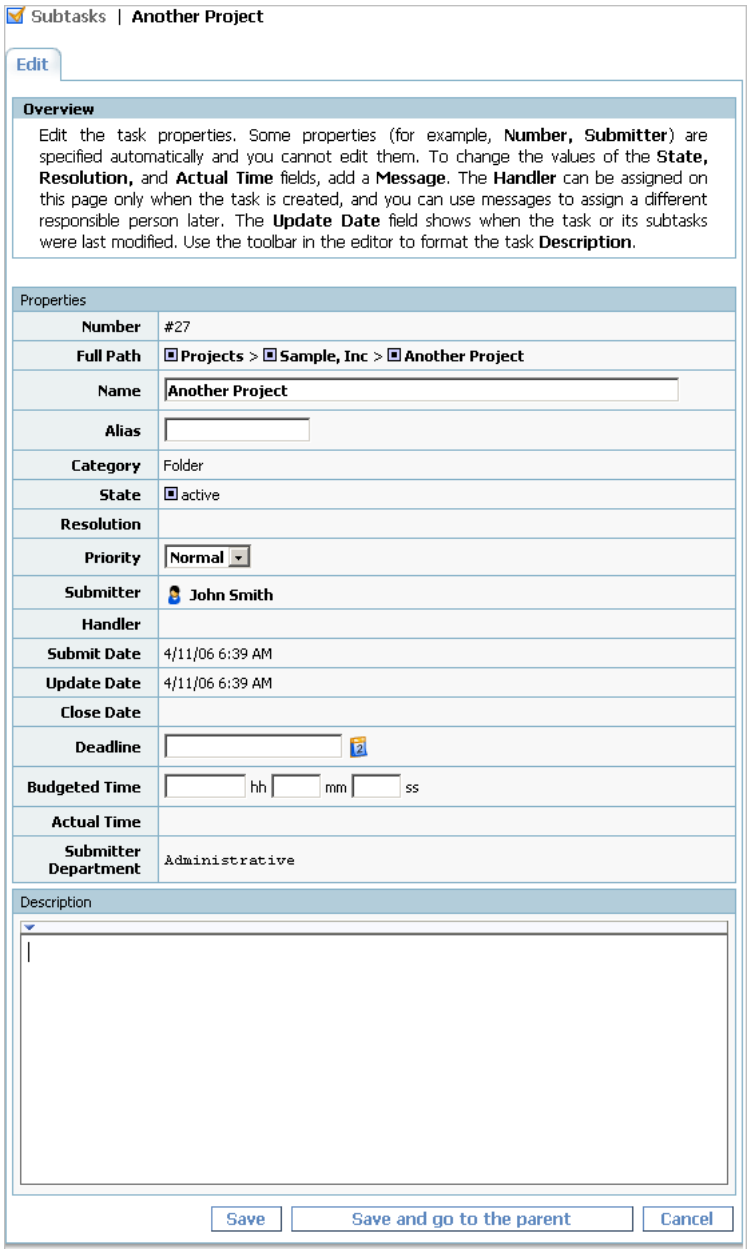
The following topic describes how to edit task properties.

Description

To edit task properties:

1. Choose a task you want to edit.
2. Click the **Task** tab.
3. Click the **Edit** link.
4. Update task properties.
5. Click the **Save** button.





See Also

- Task Properties (see page 134)
- Bulk Updating Tasks (see page 74)
- Receiving E-mail Notification when Tasks Change (see page 126)

3.4.1.4 Assigning a Task

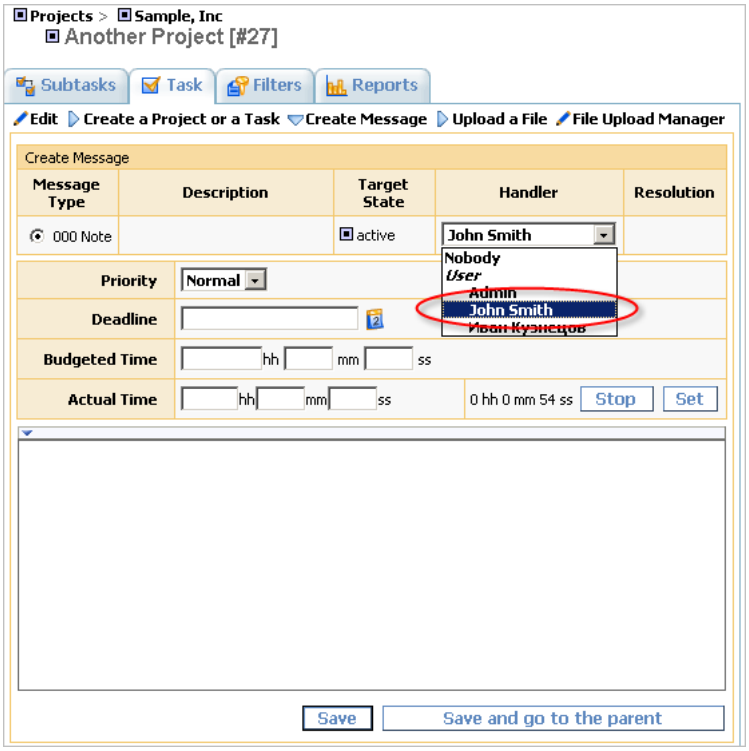
The following topic describes how to assign a task.

Description

To assign an existing task:

1. Click the **Task** tab.
2. Expand the **Create Message** pulldown.
3. Choose a new task handler.

4. Click the **Save** button.



Conditions

You cannot assign a task if:

- You do not have permissions to add messages for the task or change the task handler.
  1. Click the **Current User** -> **Statuses...** menu item.
  2. Select the status and check permissions.
- There are no message types available that move the task to the desired target state, or you do not have permissions to add messages of the required type. To check available message types and message type permissions:
  1. Click the **Current Task -> Workflows...** menu item.
  2. Click the task's workflow.
  3. Click the **Message Types** tab.
  4. Click the message type for which you want to view permissions.
  5. Click the **Permissions** tab.
- There are no available handlers for this task. To see the available handlers click the **Current Task -> Access Control Rules...** menu item.

See Also

- Task Concepts (see page 50)
- Adding Comments to a Task (see page 64)
- Setting Default Handler for a Project (see page 99)
- Restricting Handler List (see page 98)
- Receiving E-mail Notification when Tasks Change (see page 126)
- Task Properties (see page 134)
- Message Properties (see page 136)

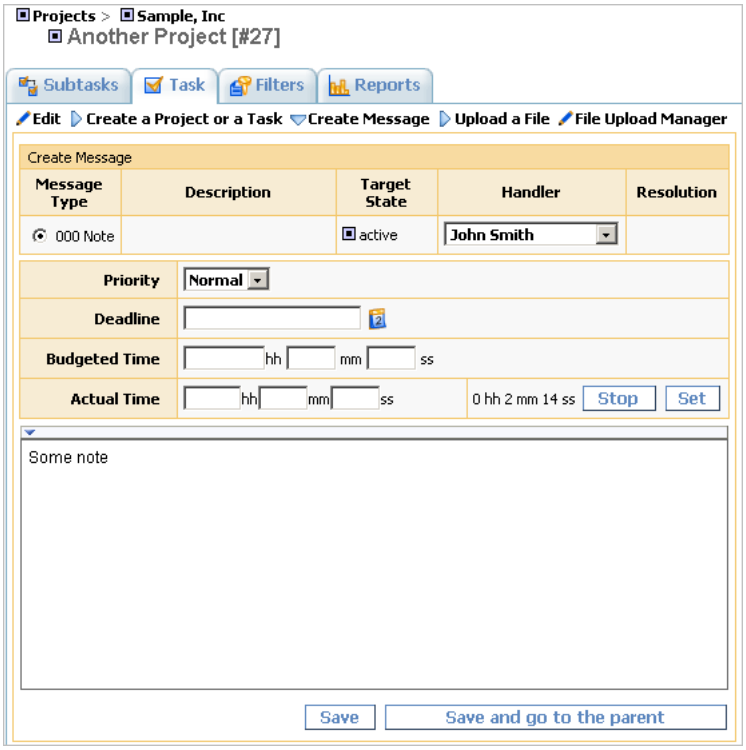
3.4.1.5 Adding Comments to a Task

The following topic describes how to add comments to a task using messages.

Description

To add comments to a task:

1. Click the **Task** tab.
2. Expand the **Create Message** pulldown.
3. Fill in the message description with the comment information.
4. **Optional:** Fill in other message properties.
5. Click the **Save** button.



Conditions

You cannot add a comment to a task if:

- You do not have permissions to add messages for the task (**Current User** -> **Statuses...** menu item).
- There are no message types available that move the task to the desired target state. To check available message types:
  1. Click the **Current Task -> Workflows...** menu item.
  2. Click the task's workflow.
  3. Click the **Message Types** tab.
- You do not have permissions to add messages of the required type. To check message type permissions:
  1. Click the **Current Task -> Workflows...** menu item.
  2. Click the task's workflow.
  3. Click the **Message Types** tab.
  4. Click the message type for which you want to view permissions.
  5. Click the **Permissions** tab.

See Also

- Task Concepts (see page 50)
- Setting Default Handler for a Project (see page 99)
- Restricting Handler List (see page 98)
- Receiving E-mail Notification when Tasks Change (see page 126)
- Task Properties (see page 134)
- Message Properties (see page 136)

3.4.1.6 Changing the Task State

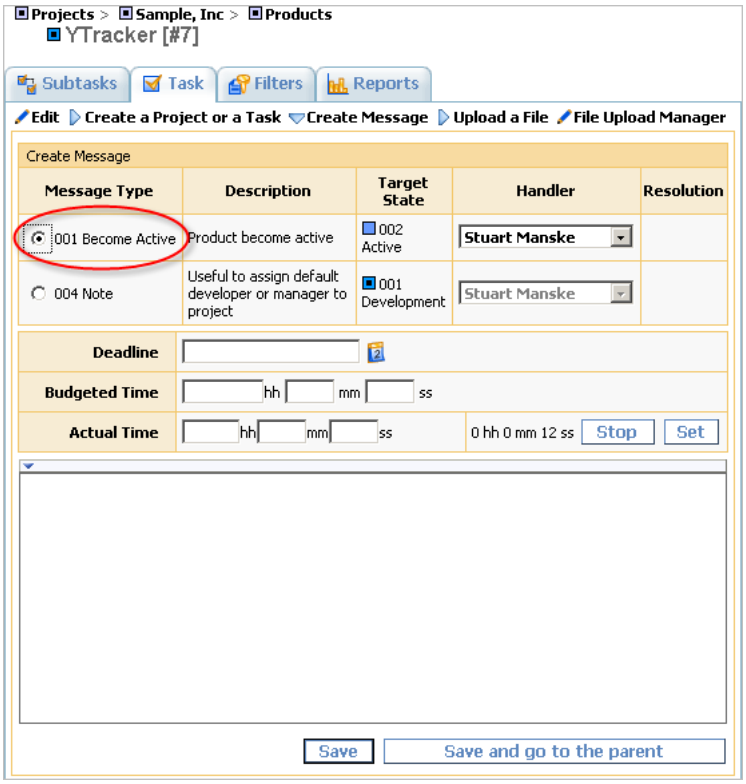
The following topic describes how to change the task state.

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** tab.
2. Expand the **Create Message** pulldown.
3. Choose the message type that will move your task to the required target state.
4. **Optional:** Fill in other message properties.
5. Click the **Save** button.



Conditions

You cannot change task state if:

- You do not have permissions to add messages to the task (**Current User** -> **Statuses...** menu item).
- There are no message types available that move the task to the desired target state. To check available message types:

1. Click the **Current Task -> Workflows...** menu item.
2. Click the task's workflow.
3. Click the **Message Types** tab.

- You do not have permissions to add messages of the required type. To check message type permissions:
  1. Click the **Current Task -> Workflows...** menu item.
  2. Click the task's workflow.
  3. Click the **Message Types** tab.
  4. Click the message type for which you want to view permissions.
  5. Click the **Permissions** tab.

See Also

- Task Concepts (see page 50)
- Adding Comments to a Task (see page 64)
- Restricting Handler List (see page 98)
- Restricting Who Can Close Task (see page 99)
- Receiving E-mail Notification when Tasks Change (see page 126)
- Closing a Bug (see page 77)
- Task Properties (see page 134)
- Message Properties (see page 136)

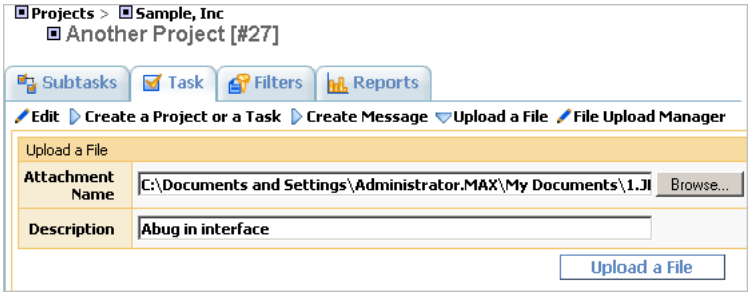
3.4.1.7 Attaching a File to a Task

The following topic describes how to attach a file to a task.

Description

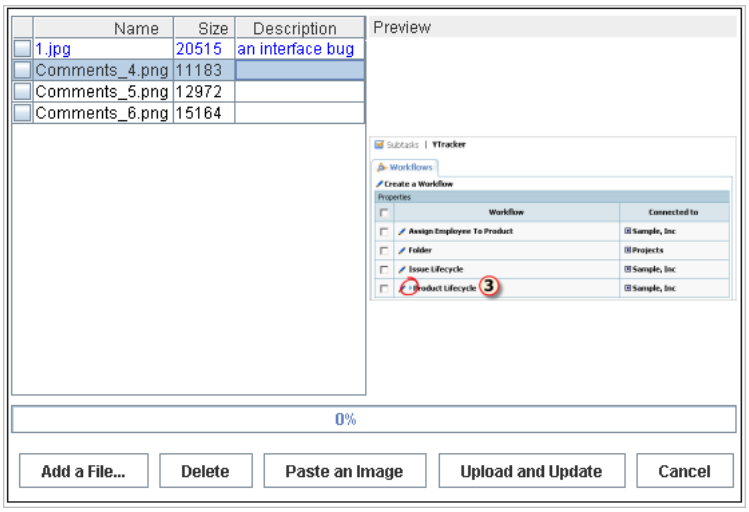
To attach a file to a task:

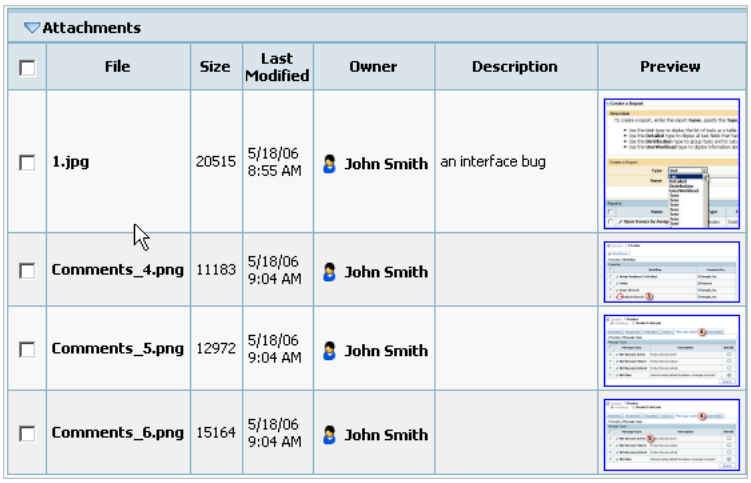
1. Click the **Task** tab.
2. Expand the **Upload a File** pulldown.
3. Fill in the attachment name and description.
4. Click the **Upload a File** button.



To attach a screenshot to a task (Microsoft Windows only):

1. Click the **Task** tab.
2. Click the **File Upload Manager** link.
3. Copy screenshot to clipboard using **Alt-PrtSc**.
4. Click the **Paste an Image** button.
5. Click the **Upload and Update** button.





Conditions

You cannot attach a file to the task if:

- You do not have permissions to upload files for this task (**Current User** -> **Statuses...** menu item).
- The file is larger than the maximum upload file size as defined by the TrackStudio administrator (**trackstudio.maxUploadSize** property in **trackstudio.properties**).

See Also

- TrackStudio Configuration Properties (see page 35)

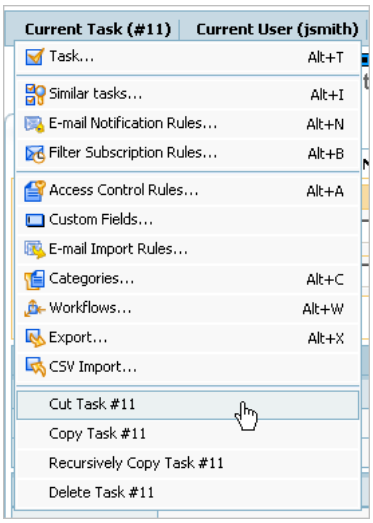
3.4.1.8 Moving Tasks

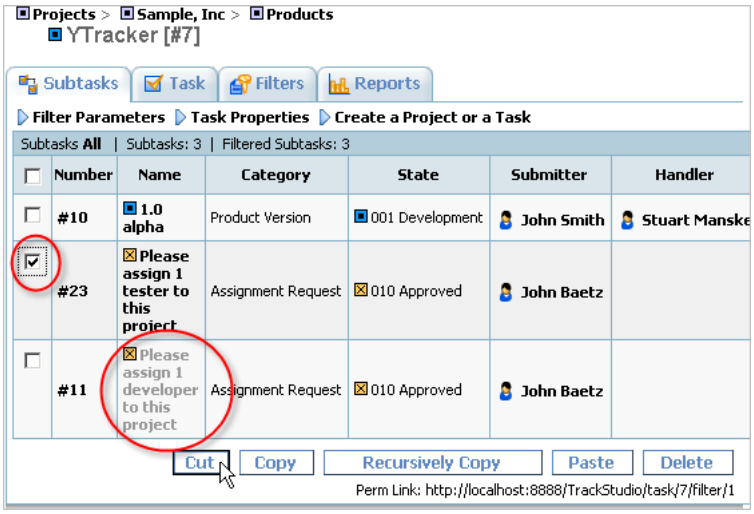
The following topic describes how to move a task.

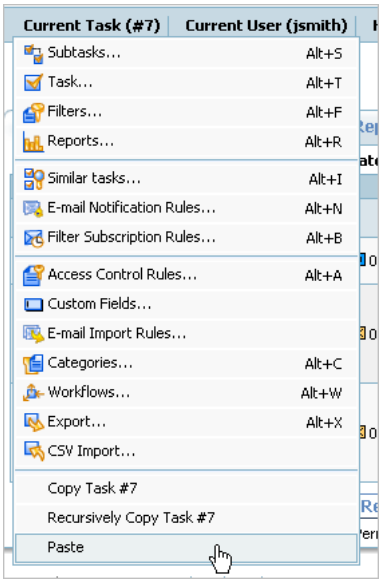
Description

To move a task:

1. Choose the task you want to move.
2. Click the **Current Task -> Cut Task** menu item.
3. Click the task that will be a new parent for the moved task.
4. Click the **Current Task -> Paste** menu item.







See Also

- Task Concepts (see page 50)
- Task Properties (see page 134)

3.4.1.9 Linking Tasks

The following topic describes how to link tasks.

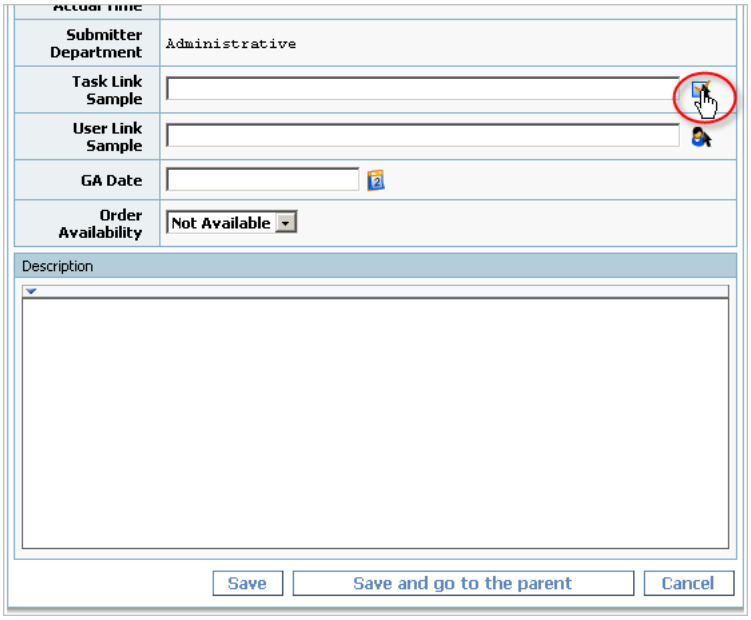
Before You Begin

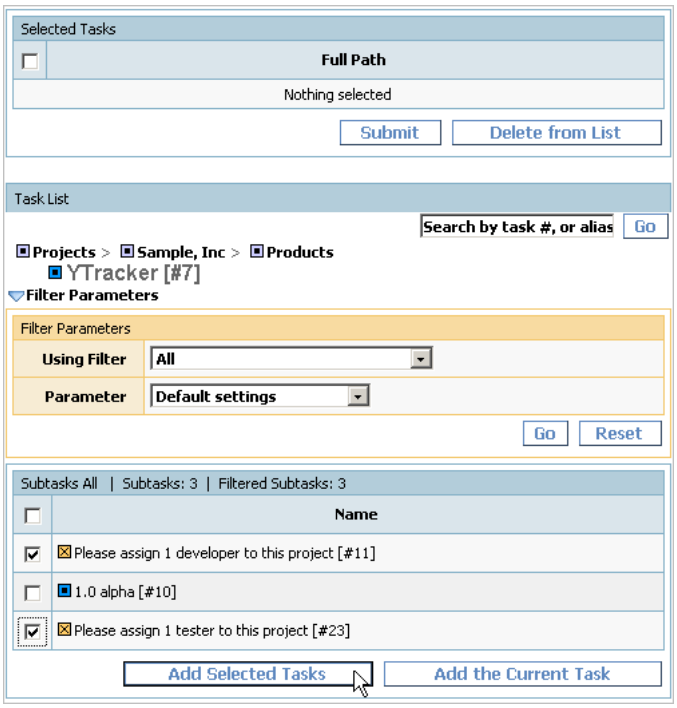
- Create a custom field with **Task** type.

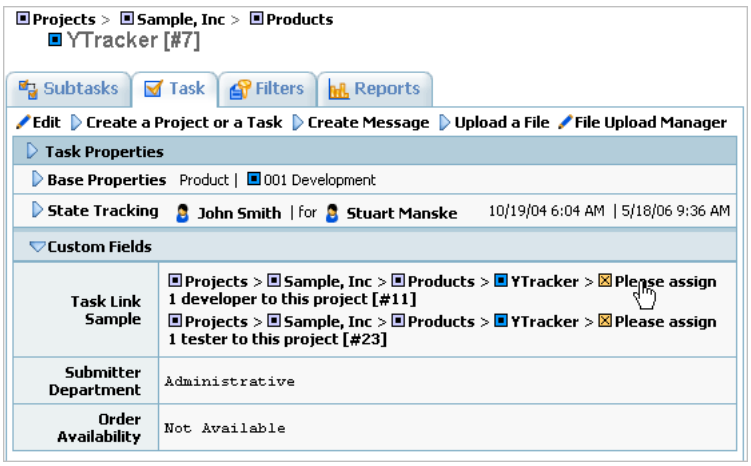
Description

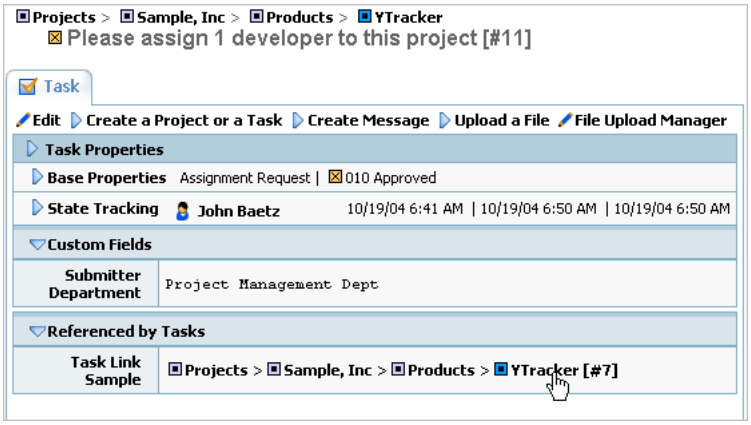
To link tasks:

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









See Also

- Task Concepts (see page 50)
- Custom Field Concepts (see page 54)
- Adding a Custom Field (see page 94)
- Custom Field Properties (see page 136)

3.4.1.10 Linking Users to a Task

The following topic describes how to link users to a task.

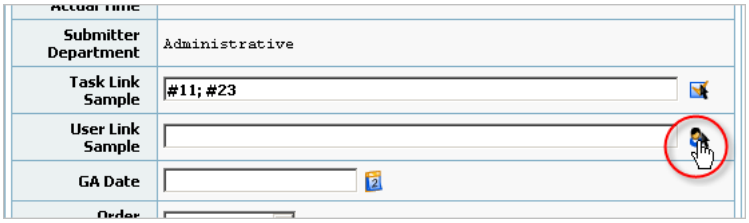
Before You Begin

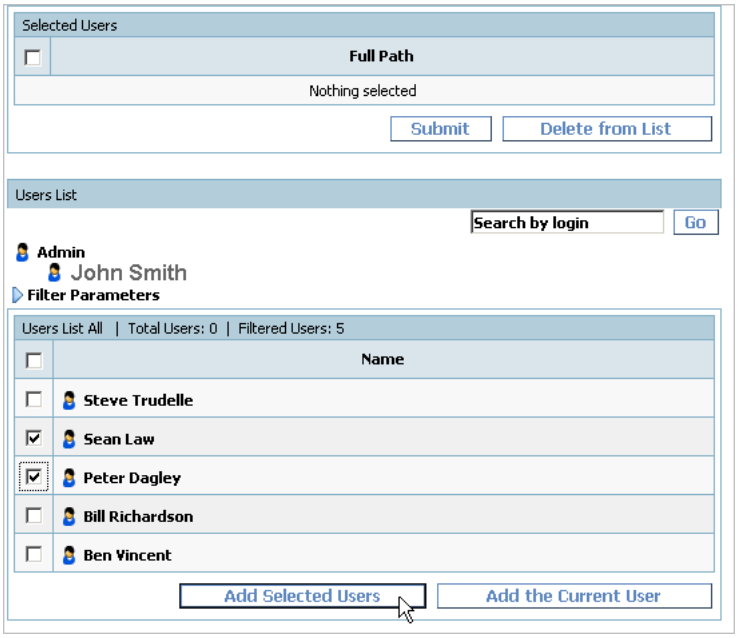
- Create a custom field with **User** type.

Description

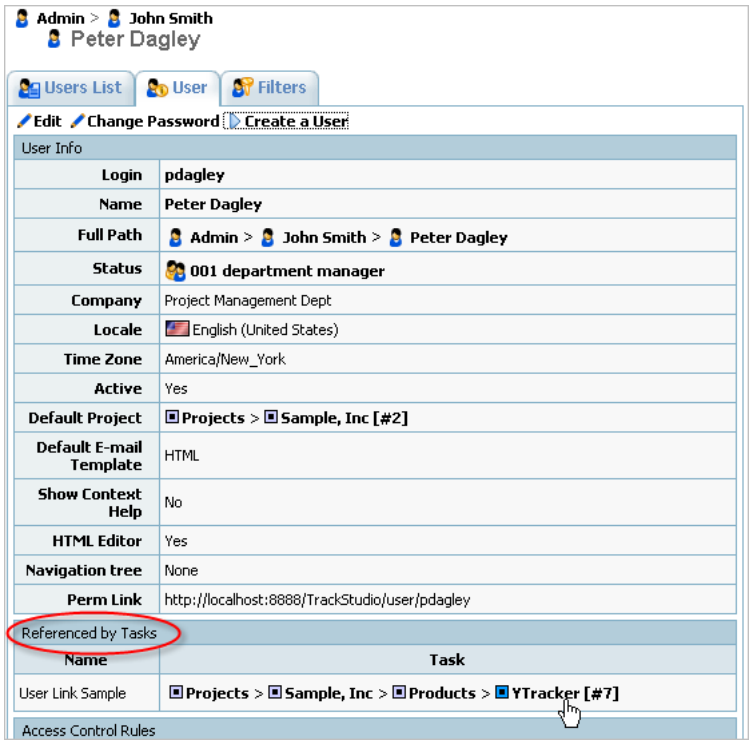
To link users to a task:

1. Click the **Task** tab.
2. Click the **Edit** link.
3. Use the custom field of **User** type to associate users with this task.
4. Click the **Save** button.









See Also

- Task Concepts (see page 50)
- Custom Field Concepts (see page 54)
- Adding a Custom Field (see page 94)
- Custom Field Properties (see page 136)

3.4.1.11 Generating a Report

The following topic describes how to generate a report.

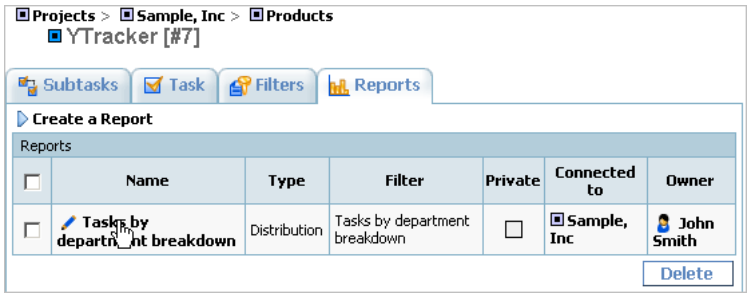
Before You Begin

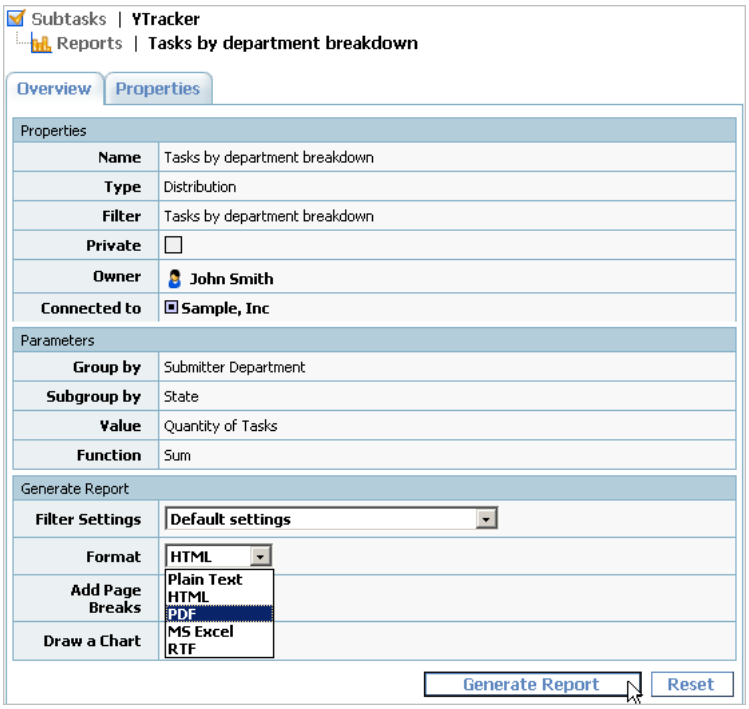
- Create a filter to select tasks for the report.

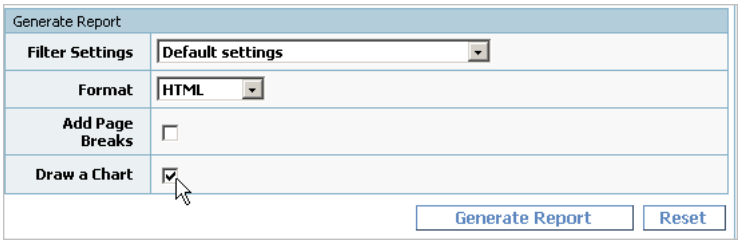
Description

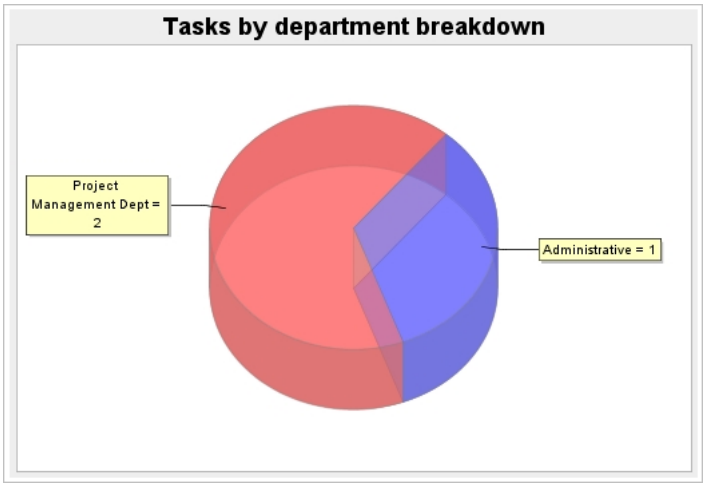
To generate a report:

1. Click the **Current Task -> Reports...** menu item.
2. Choose existing reports or create a new one.
3. Fill in report properties.
4. Click the **Generate Report** button.




Subtasks | YTracker
Reports | Tasks by department breakdown
Overview
Properties
Properties
Name
Tasks by department breakdown
Type
Distribution
Filter
Tasks by department breakdown
Private
Owner
John Smith
Connected to
Sample, Inc
Parameters
Group by
Submitter Department
Subgroup by
State
Value
Quantity of Tasks
Function
Sum
Generate Report
Filter Settings
Default settings
Format
HTML
Plain Text
HTML
PDF
MS Excel
RTF
Add Page Breaks
Draw a Chart
Generate Report
Reset


Generate Report
Filter Settings
Default settings
Format
HTML
Add Page Breaks
Draw a Chart
Generate Report
Reset

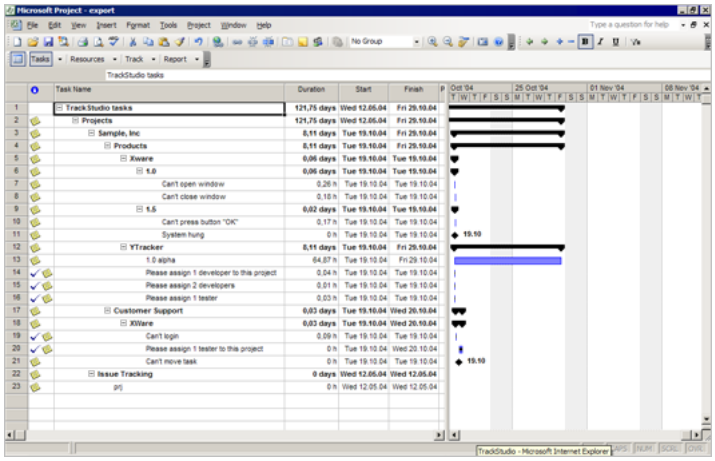

Tasks by department breakdown
Project Management Dept = 2
Administrative = 1

Tasks by department breakdown

Filter	Tasks by department breakdown
Task	Projects > Sample, Inc [#2]
Type	Distribution

Submitter Department	State	Sum(Quantity of Tasks)
Administrative	001 Development (Product Lifecycle)	2.0
	002 Active (Product Lifecycle)	2.0
	003 Mature (Product Lifecycle)	2.0
	active (Folder)	4.0
	sum	10.0
GoodBank, Ltd	000 New (Issue Lifecycle)	1
	003 Closed (Issue Lifecycle)	1
	sum	2.0
Project Management Dept	010 Approved (Assign Employee To Product)	4.0
	sum	4.0
QA Dept	000 New (Issue Lifecycle)	2.0
	001 Resolved (Issue Lifecycle)	1
	002 Verified (Issue Lifecycle)	1
	sum	4.0
Summary (sum)	Administrative	10.0
	GoodBank, Ltd	2.0
	Project Management Dept	4.0
	QA Dept	4.0
	sum	20.0

Page:1 Created 4/12/06 9:19 AM Powered by TrackStudio



See Also

- Filtering Tasks Using AND/OR/NOT (see page 117)
- Analyzing Tasks Distribution (see page 119)

3.4.1.12 Bulk Updating Tasks

The following topic describes how to update multiple tasks.

Before You Begin

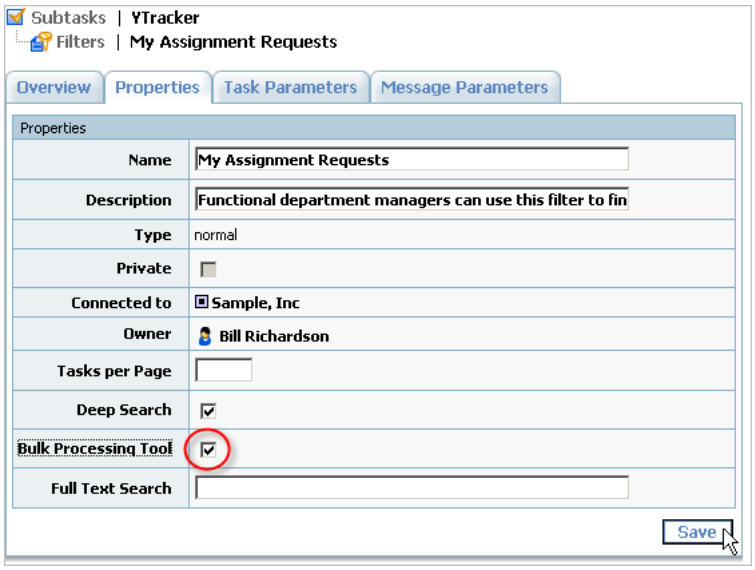
1. Choose the **Filters** tab.
2. Choose an existing filter or create a new one.
3. Edit the filter, and check the **Bulk Processing Tool** setting.
4. Click the **Save** button.

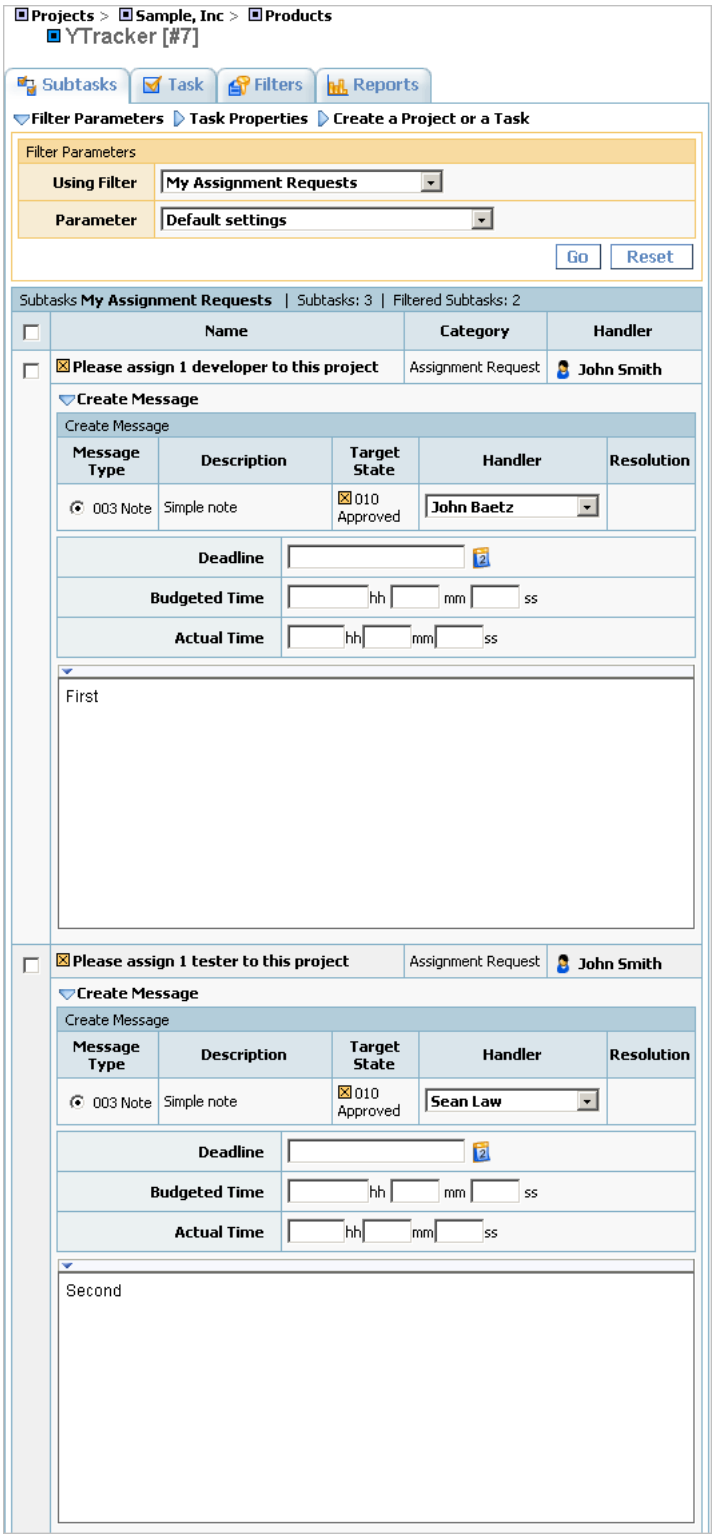
Description

To update multiple tasks:

1. Choose the **Subtasks** tab.

2. Use your filter to show the task list.
3. Expand the **Create Message** pulldown for the tasks that should be updated.
4. Fill in message properties for the tasks. To add a message using the **Bulk Processing Tool** you should specify a non-empty **Description** for the message.
5. Click the **Save** button.





See Also

- Editing Task Properties (see page 61)
- Message Properties (see page 136)
- Task Filter Properties (see page 138)

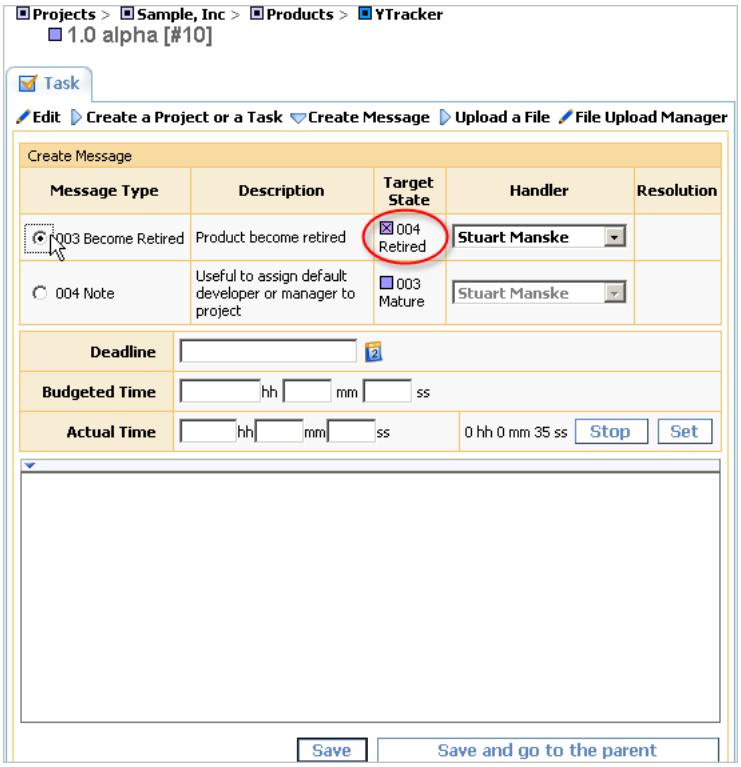
3.4.1.13 Closing a Bug

The following topic describes how to close bugs.

Description

To close a bug:

1. Click the **Task** tab.
2. Expand the **Create Message** pulldown.
3. Choose the appropriate message type.
4. **Optional:** Fill in other message properties.
5. Click the **Save** button.



Conditions

You can't close a task if:

- You do not have permissions to add messages for the task (**Current User** -> **Statuses...** menu item).
- The option is unavailable in the current task state. To see available message types:
  1. Click the **Current task -> Workflow...** menu item.
  2. Click the task's workflow.
  3. Click the **Message Types** tab.

See Also

- Task Concepts (see page 50)
- Adding Comments to a Task (see page 64)
- Restricting Handler List (see page 98)
- Restricting Who Can Close Task (see page 99)
- Changing the Task State (see page 65)

- Task Properties (see page 134)
- Message Properties (see page 136)

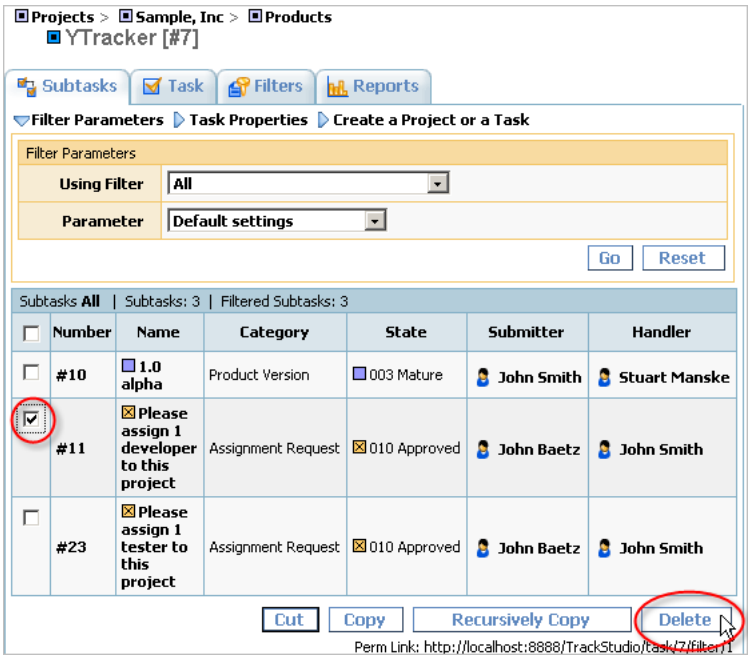
3.4.1.14 Deleting a Task

The following topic describes how to delete a task.

Description

To delete a task:

1. Choose the task you want to delete.
2. Click the **Current Task -> Delete Task** menu item.



Conditions

You cannot delete a task if:

- You do not have permissions to delete the task (Use **Current User** -> **Statuses...** and **Current Task -> Access Control Rules...** menu items to check the permissions).

3.4.2 Managing User Accounts

3.4.2.1 Establishing a User Group Account

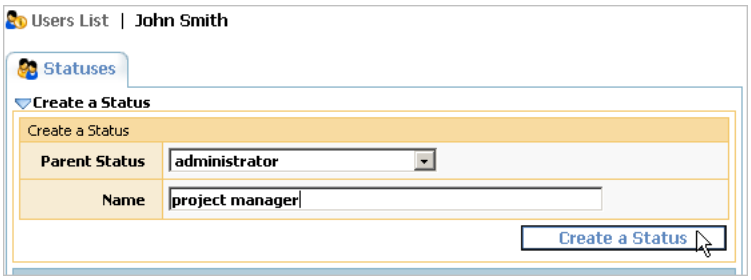
The following topic describes how to create a user group.

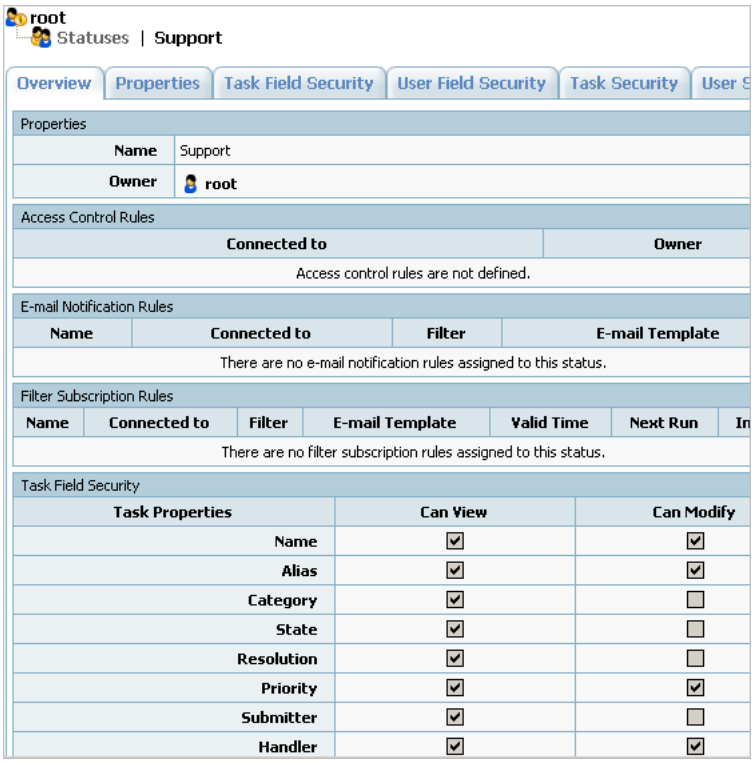
Description

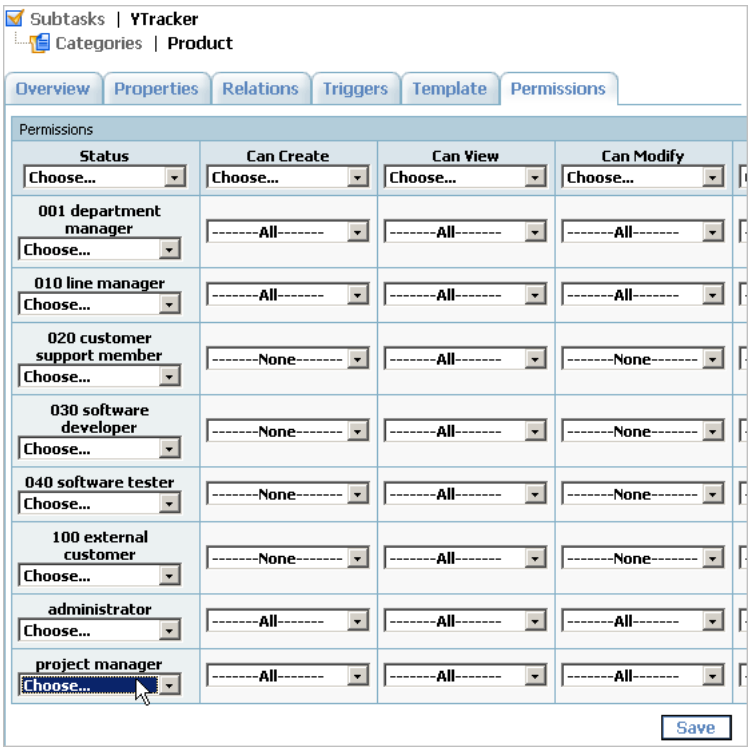
To create a user group:

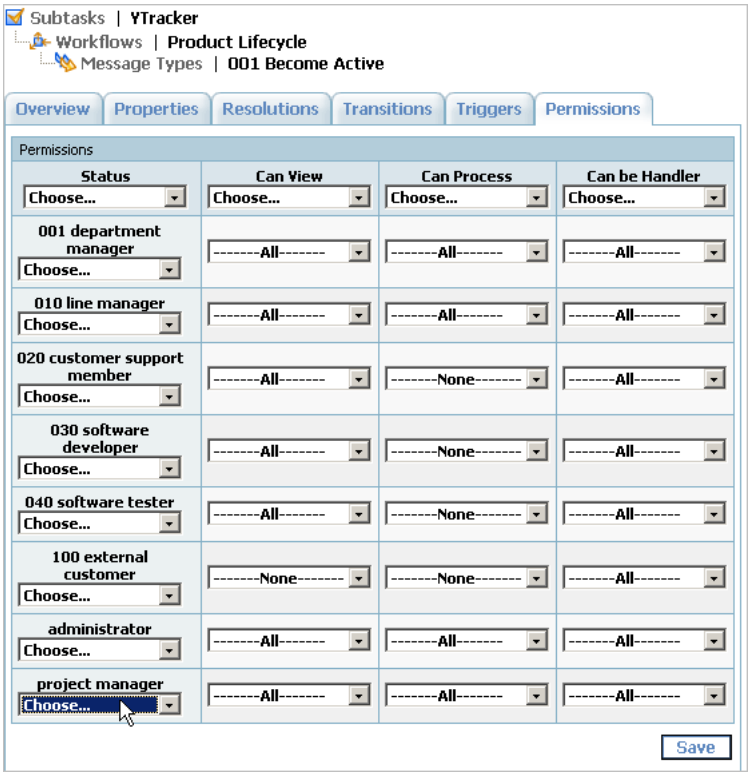
1. Click the **Current User -> Statuses...** menu item**.**
2. Expand the **Create a Status** pulldown.
3. Enter the status **Name**.

4. Click the **Create a Status** button.
5. Set permissions for users who hold this status.
6. Click the **Save** button.
7. Click the **Current Task -> Categories...** menu item.
8. For each category that should be available for the user, click the **Permissions** tab to specify which users can view, create, delete or edit tasks. To view a task, users should be able to view all parent tasks.
9. Click the **Current Task -> Workflows...** menu item.
10. Click the **Message Types** tab.
11. For each message type, click the **Permissions** tab to specify which users can create and view messages of that type.
12. Define custom field permissions if required.









See Also

- User Status Concepts (see page 53)
- Defining Which Users Can Create, Edit or Delete Tasks (see page 101)
- Restricting Who Can Close Task (see page 99)
- Restricting Handler List (see page 98)
- Defining Who Can View and Edit Custom Fields (see page 96)

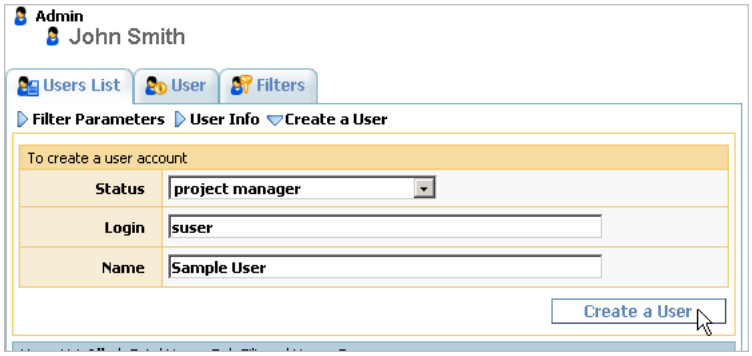
3.4.2.2 Creating a User Account

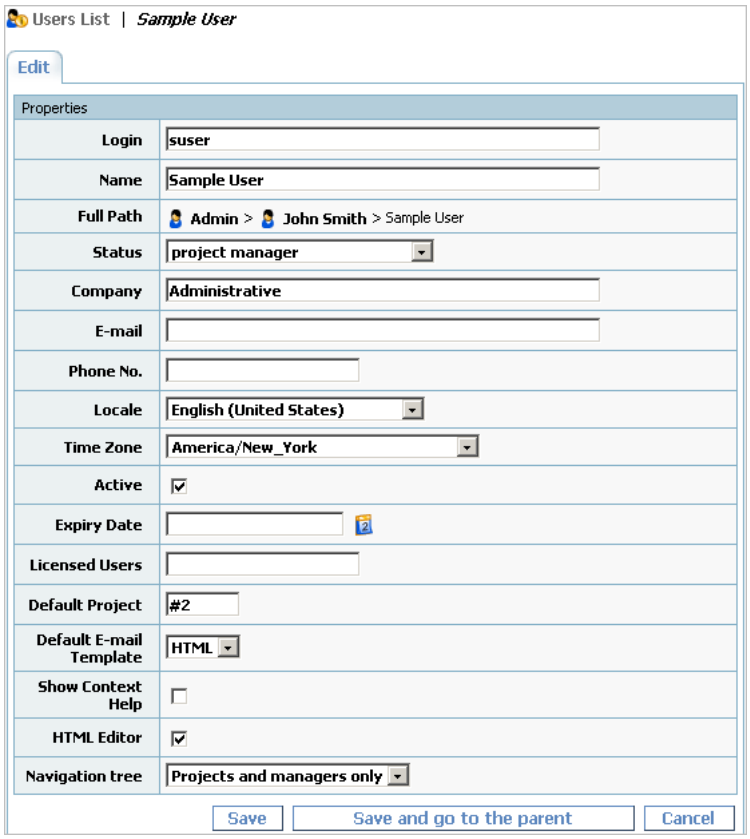
The following topic describes how to create an account.

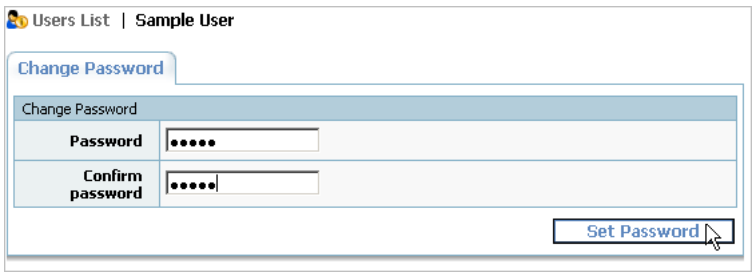
Description

To create a user account:

1. Click the **Current User -> User...** menu item.
2. Fill in the user's **Login** and **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 **Change Password** link.
7. Type the new password twice.
8. Click the **Set Password** button.
9. Specify the tasks and projects to which the user will have access by using the **Current Task -> Access Control Rules...** menu item.







See Also

- User Concepts (see page 51)
- User Status Concepts (see page 53)
- Granting Users Access to a Project (see page 83)
- Defining Which Users Can Create, Edit or Delete Tasks (see page 101)
- Restricting Who Can Close Task (see page 99)
- Restricting Handler List (see page 98)
- Choosing Localized Interface (see page 101)
- User Properties (see page 135)

3.4.2.3 Creating a Customer Account

The following topic describes how to create a customer account.

Description

To create a customer account:

1. Create a user account.

2. Grant access to your customer account for customer support members.

See Also

- Creating a User Account (see page 81)
- Granting Users Access to Other Users (see page 84)

3.4.2.4 Editing Account Properties

The following topic describes how to edit a user account.

Description

To edit user account properties:

1. Click the **Current User -> User...** menu item.
2. Click the **Edit** link.
3. Enter user account properties.
4. Click the **Save** button.

See Also

- User Properties (see page 135)

3.4.2.5 Granting Users Access to a Project

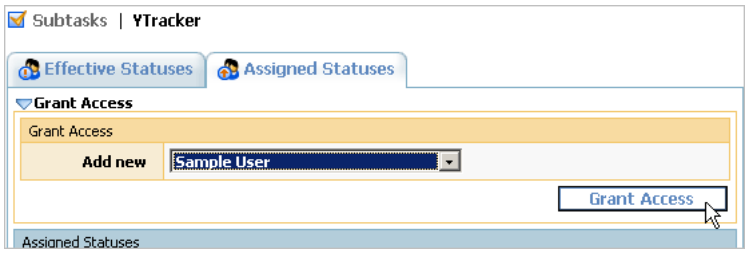
The following topic describes how to grant users access to a project.

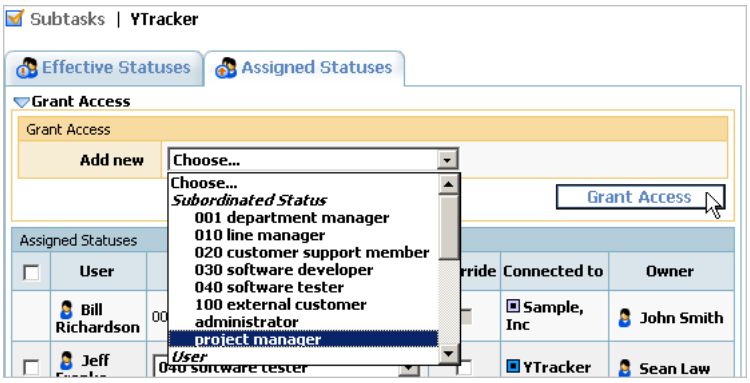
Description

To grant access to a project:

1. Select a project for which you want to give your subordinate users access.
2. Click the **Current Task -> Access Control Rules...** menu item.
3. Click the **Assigned Statuses** tab.
   - To apply the user's own status as the effective status for the current task (a *smanske* user must have the *software developer* status for the current task), grant access for your user with the default parameters.
   - To extend the list of inherited statuses (a *smanske* user must have both the *software developer* and *line manager* statuses), grant access for your user, and assign the necessary statuses to them. The user's own status and any statuses assigned to the parent tasks will be inherited automatically - you shouldn't assign them.
   - To completely redefine the list of effective statuses (a *smanske* user must have only the *line manager* status), grant access for your user, assign any necessary statuses, and mark the **Override** checkbox as needed. In this case, the user's own status and any inherited assigned statuses specified for the parent tasks are not taken into account, when TrackStudio determines effective status for the user.
   - To make a task accessible for all subordinate users whose own status is the same (all users with a status of *software developer* can access the current task), grant access for this status.

Users can use tasks which lie between the root task and accessible tasks only for navigating. Users can neither view the description of such tasks nor modify them.





See Also

- User Status Concepts (see page 53)

3.4.2.6 Granting Users Access to Other Users

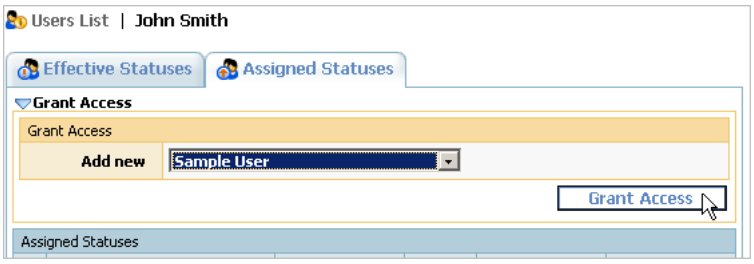
The following topic describes how to grant users access to other users.

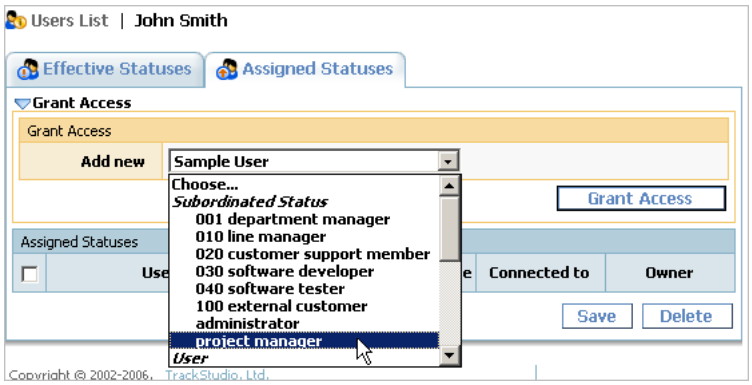
Description

To grant access to users:

1. Select a subordinate user, access to whom you are going to grant.
2. Click the **Current User -> Access Control Rules...**
3. Click the **Assigned Statuses** tab.
   - To apply the user's own status as the effective status for the current user (a *smanske* user must have the *software developer* status for the user *ctuck*), grant access for your user with the default parameters.
   - To extend the list of inherited statuses (a *smanske* user must have both the *software developer* and *line manager* statuses for the user *ctuck*), grant access for your user and assign the necessary statuses to them. The user's own status and any statuses assigned to the parent users will be inherited automatically - you shouldn't assign them.
   - To completely redefine the list of effective statuses (a *smanske* user must have only the *line manager* status for the user *ctuck*), grant access for your user, assign any necessary statuses, and mark the **Override** checkbox as needed. In this case, the user's own status and any inherited assigned statuses specified for the parent users are not taken into account, when TrackStudio determines effective status for the user.
   - To make a user accessible for all subordinate users whose own status is the same (all users with a status of *software developer* can access the user *ctuck*), grant access for this status.

Users can use users which lie between the root user and accessible users only for navigating. Users can neither view properties of such users nor modify them.





See Also

- User Status Concepts (see page 53)

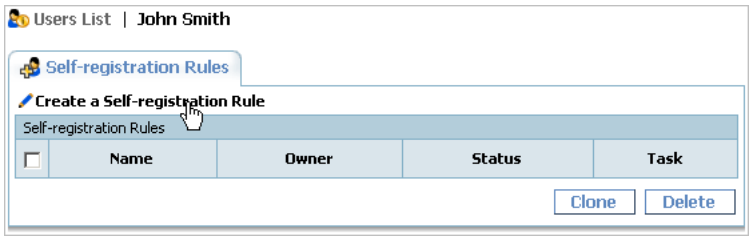
3.4.2.7 Customers Creating Accounts for Themselves

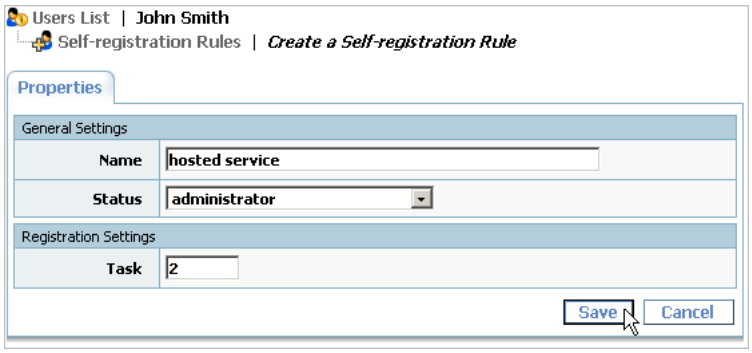
The following topic describes how to give customers access to create their own accounts.

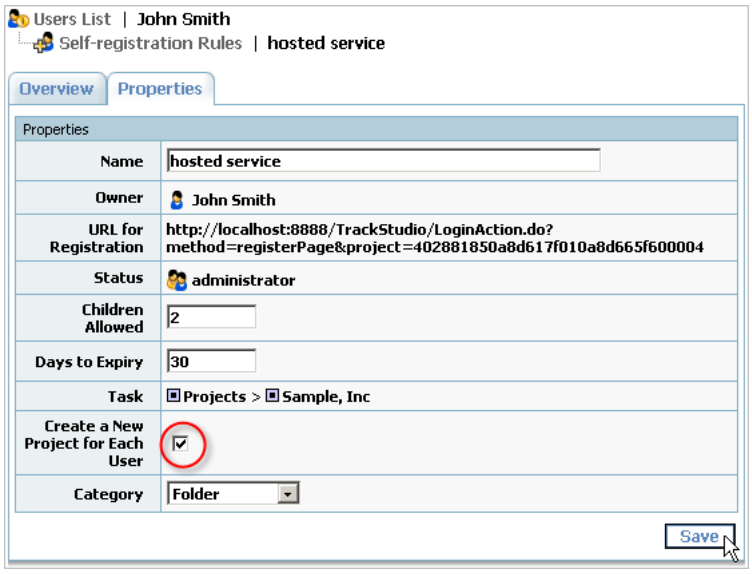
Description

To allow customers to create accounts for themselves:

1. Enable e-mail notification in **trackstudio.mail.properties**. You can also use the Server Manager.
2. Click the **Current User -> Self-registration Rules...** menu item.
3. Click the **Create a Registration Rule** link.
4. Fill in registration rule properties.
5. Click the **Save** button.
6. Click the **Properties** tab.
7. **Optional:** Specify additional options for the created registration rule.
8. Click **the Save** button.









Example 1

Suppose we need to organize a hosted service. To do that, new users should register with *administrator* status. 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*
Days to Expiry	*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
Days to Expiry	<empty>
Create a New Project for Each User	Off

See Also

- Self-registration Concepts (see page 56)
- Self-registration Rule Properties (see page 144)

3.4.2.8 Finding a User Account by Login

The following topic describes how to find a user account by login.

Description

To find a user by login:

1. Enter the user login into the input field in the right section of the **Main Menu**.
2. Click the **Go** button.

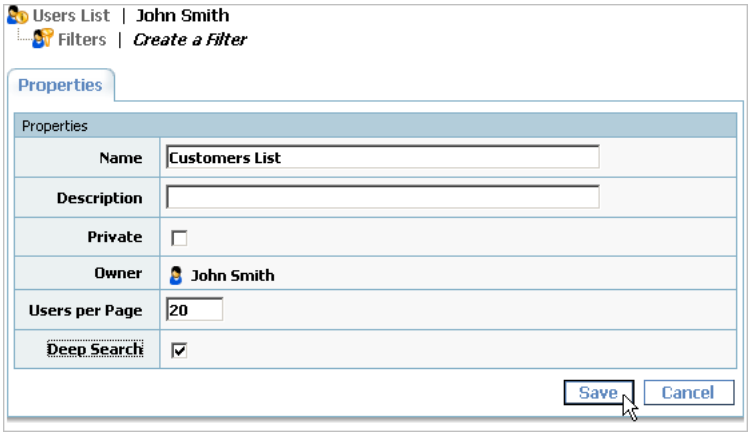
3.4.2.9 Finding a User Account by Properties

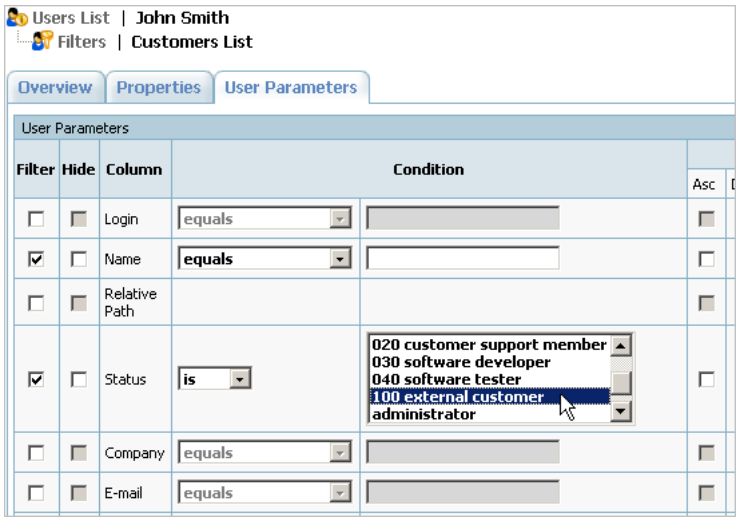
The following topic describes how to find a user account by properties.

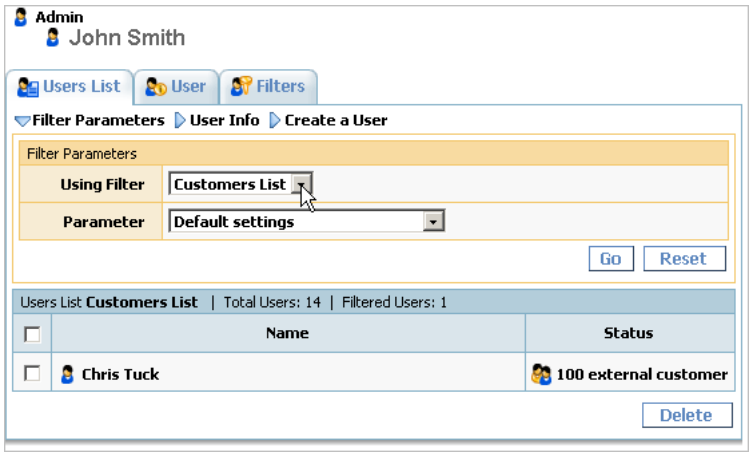
Description

To find a user account by properties:

1. Click the **Current User -> Filters...** menu item.
2. Create a new filter or select an existing one.
3. Click the **Filter Parameters** tab.
4. Fill in user search conditions.
5. Click the **Save** button.
6. Click the **Current User -> Users List...** menu item.
7. Choose the filter.







See Also

- Filter Concepts (see page 53)
- User Properties (see page 135)
- User Filter Properties (see page 139)

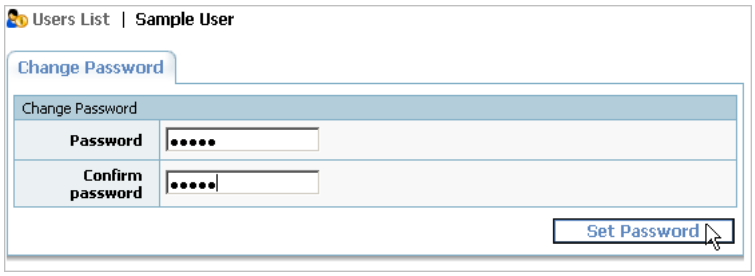
3.4.2.10 Changing a Password

The following topic describes how to change your password.

Description

To change a password:

1. Click the **Current User -> User...** menu item.
2. Click the **Change Password** link.
3. Type new password twice.
4. Click the **Set Password** button.



Conditions

- You cannot change LDAP user passwords from TrackStudio.

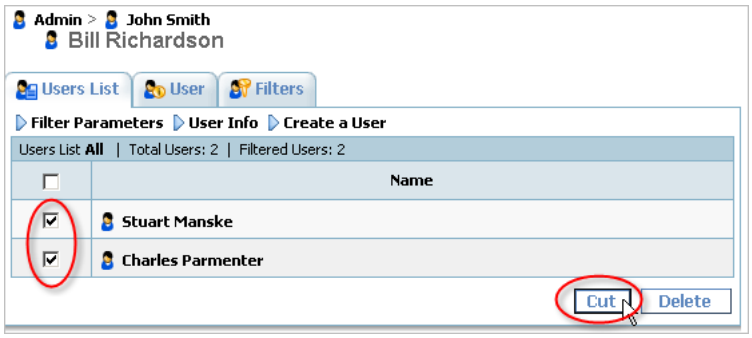
3.4.2.11 Moving a User Account

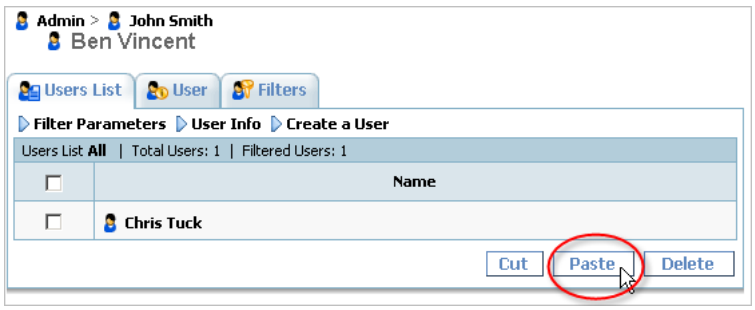
The following topic describes how to move a user account.

Description

To move a user account:

1. Choose a user account to move.
2. Click the **Current User -> Cut User** menu item.
3. Choose a new manager for the user.
4. Click the **Current User -> Paste** menu item.





Conditions

- You cannot move your own account.

3.4.2.12 Locking a User Account

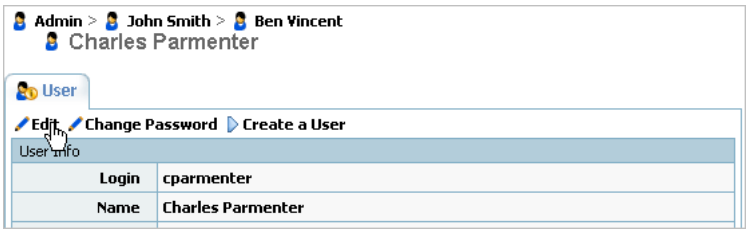
The following topic describes how to lock a user account.

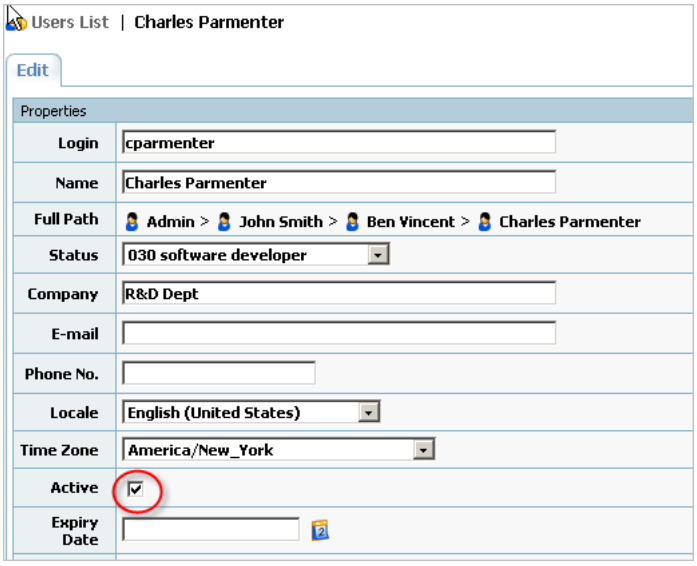
Description

To lock a user account:

1. Choose user account.
2. Click the **Current User -> User...** menu item
3. Click the **Edit** link.
4. Mark the **Active** flag off.
5. Click the **Save** button.

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





Conditions

- You cannot lock your own account.

See Also

- User Properties (see page 135)

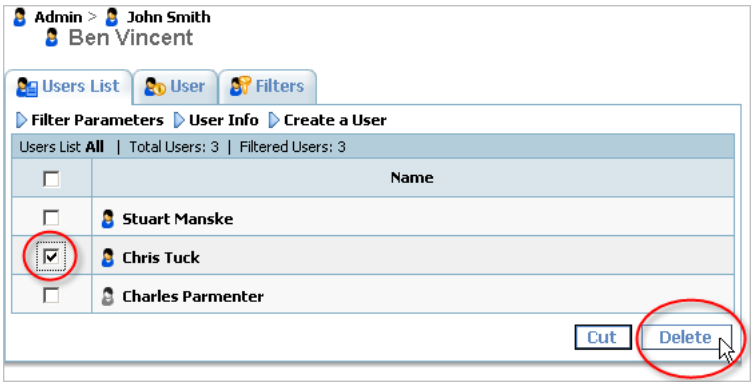
3.4.2.13 Deleting a User Account

The following topic describes how to delete a user account.

Description

To delete user account:

1. Choose a user account.
2. Click the **Current User -> Users List...** menu item.
3. Select user to delete.
4. Click the **Delete** button.



Conditions

- You must have permission to delete a user (see **Current User -> Statuses...**).
- You cannot delete a user who has subordinate users. Consider locking the account instead.
- You cannot delete your own account.

See Also

- Locking a User Account (see page 89)

3.4.3 Customizing Task and User Processing

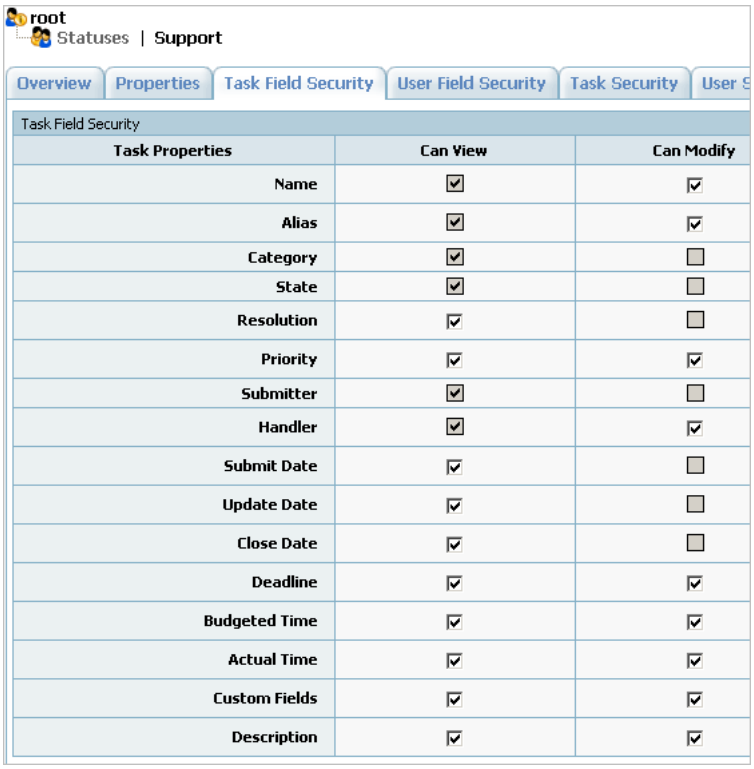
3.4.3.1 Hiding Unused Fields

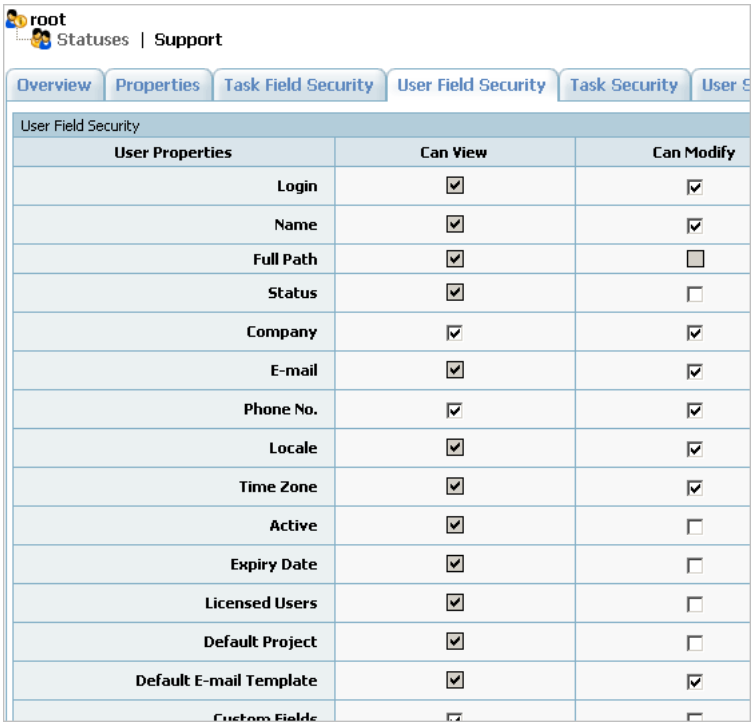
The following topic describes how to hide unused fields.

Description

To hide unused fields:

1. Click the **Current User -> Statuses...** menu item.
2. Select the user status for which you want to hide fields.
3. Click the **Task Field Security** or the **User Field Security** tab.
4. Mark required checkboxes off.
5. Click the **Save** button.





Conditions

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

Example

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

1. Click the **Current User -> Statuses...** menu item.
2. Select the user status for which you want to hide fields.
3. Click the **Task Field Security** tab.

4. Mark checkboxes off for the following fields:
   - **Budgeted Time** / Can View
   - **Actual Time** / Can View
   - **Deadline** / Can View

See Also

- Adding a Custom Field (see page 94)

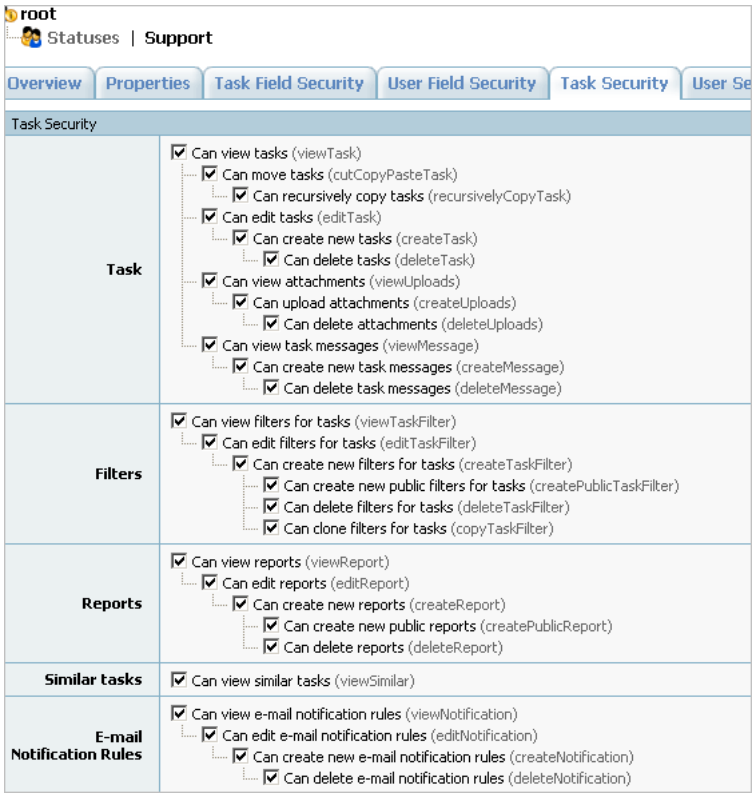
3.4.3.2 Hiding Menu Items, Tabs or Buttons

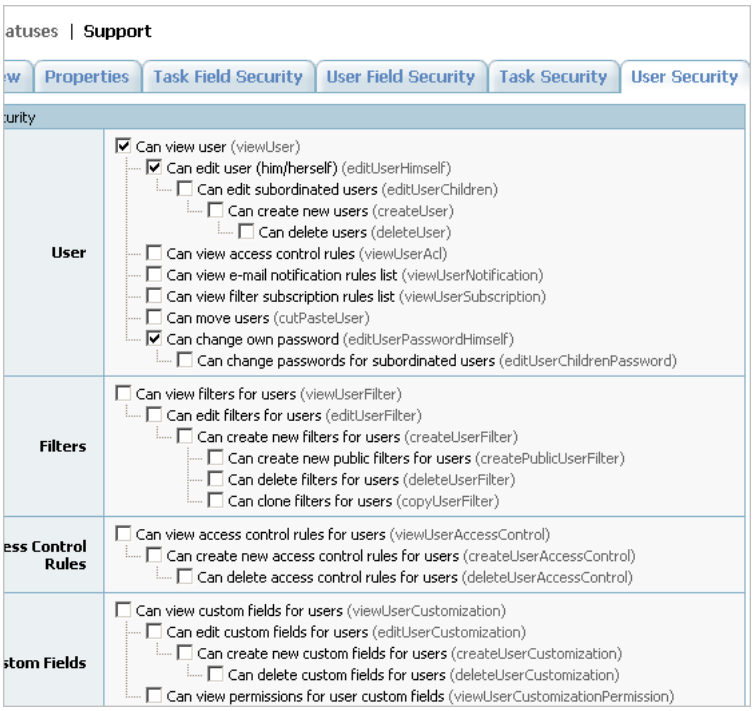
The following topic describes how to hide menu items, tabs, or buttons from a user.

Description

To hide a menu item, a tab or a button:

1. Click the **Current User -> Statuses...** menu item.
2. Select the user status for which you want to hide the menu item, the tab, or the button.
3. Click the **User Security** or the **Task Security** tab.
4. Check checkboxes off for extra items.
5. Click the **Save** button.





Conditions

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

Example

To hide the **Current Task -> Workflows...** menu item:

1. Click the **Current User -> Statuses...** menu item.
2. Select the user status for which you want to hide fields.
3. Click the **Task Security Tab** tab.
4. Mark checkbox off for the following item:
   - **Can view workflows (viewWorkflow)**

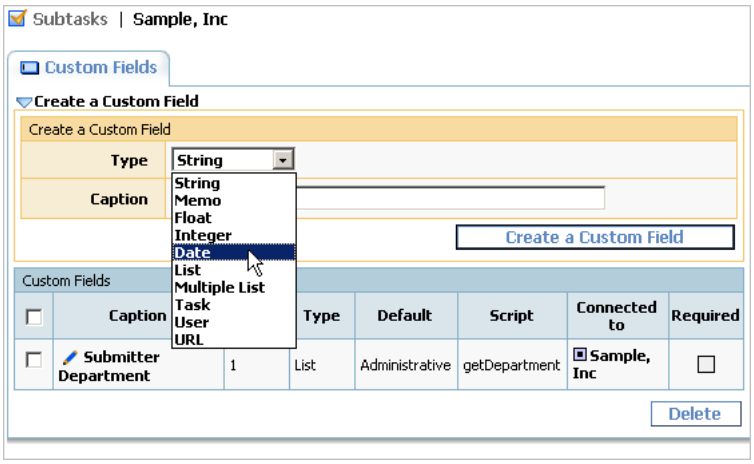
3.4.3.3 Adding a Custom Field

The following topic describes how to add a custom field.

Description

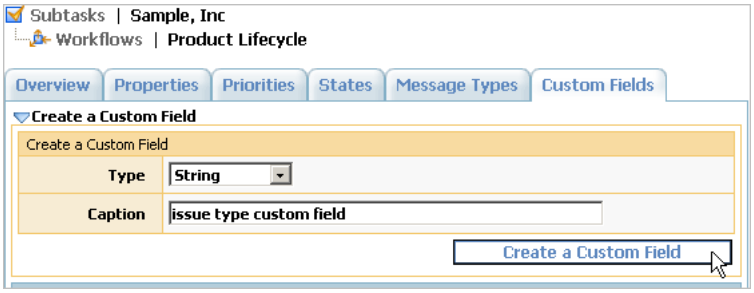
To add a custom field for a task and its subtasks:

1. Click the **Current Task -> Custom Fields...** menu item.
2. Expand the **Create a Custom Fields** pulldown.
3. Enter the custom field information.
4. Click the **Create a Custom Fields** button.



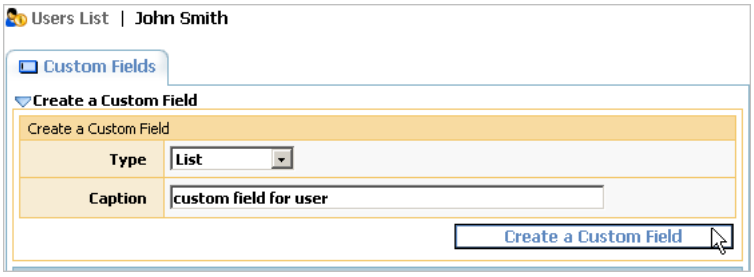
To add a custom field for all tasks that use the specified workflow:

1. Click the **Current Task -> Workflows...** menu item.
2. Chose a workflow.
3. Click the **Custom Fields** tab.
4. Expand the **Create a Custom Fields** pulldown.
5. Enter the custom field information.
6. Click the **Save** button.



To add a custom field for the user and subordinated users:

1. Click the **Current User -> Custom Fields...** menu item.
2. Expand the **Create a Custom Fields** pulldown.
3. Enter the custom field information.
4. Click the **Save** button.



See Also

- Custom Field Concepts (see page 54)
- User Status Concepts (see page 53)
- Script Concepts (see page 54)
- Calculating a Custom Field Value (see page 104)

- Hiding Unused Fields (see page 91)
- Custom Field Properties (see page 136)

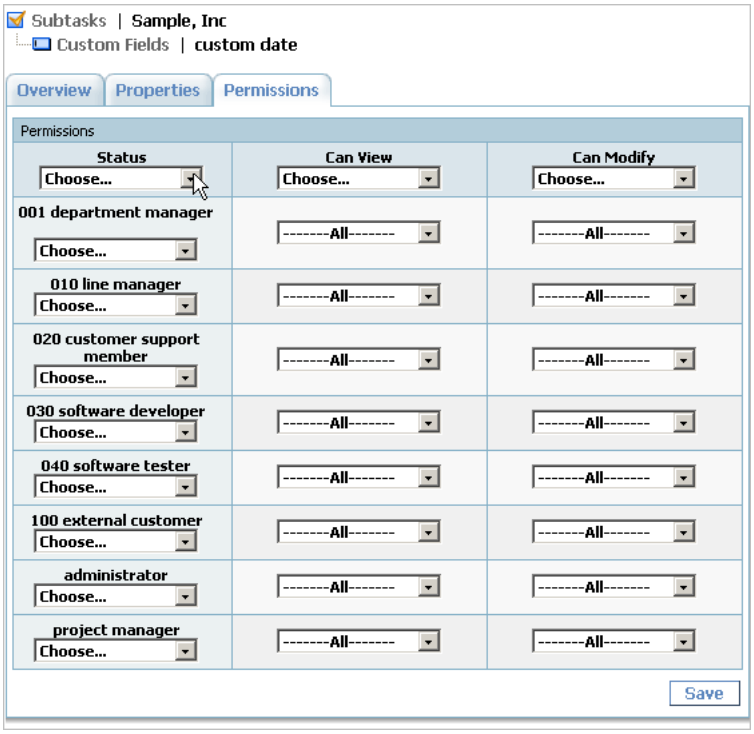
3.4.3.4 Defining Who Can View and Edit Custom Fields

The following topic describes how to define who can view and edit custom fields.

Description

To define who can view or edit a custom field connected to a task:

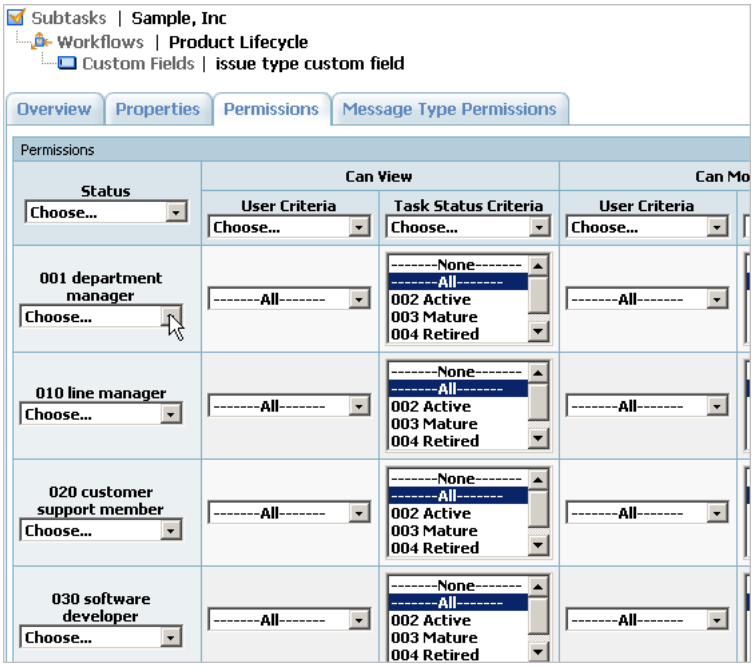
1. Select the task containing the custom field.
2. Click the **Current Task -> Custom Fields...** menu item.
3. Choose a custom field.
4. Click the **Permissions** tab.
5. Define which users can view and edit the custom field.
6. Click the **Save** button.

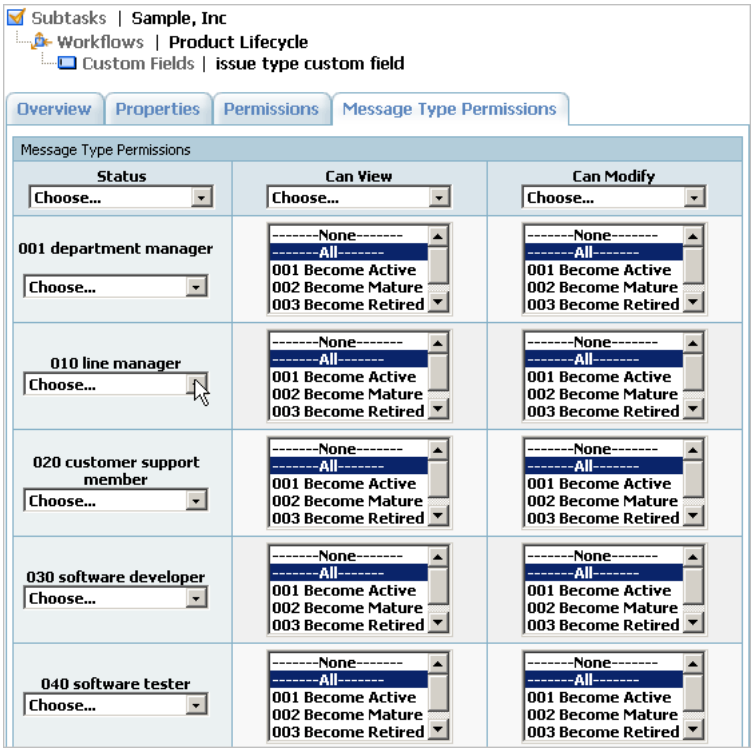


To define who can view or edit a custom field connected to a workflow:

1. Click the **Current Task -> Workflows...** menu item.
2. Choose the workflow which contains the custom field.
3. Click the **Custom Fields** tab.
4. Choose a custom field.
5. Click the **Permissions** tab.
6. Define which users can view and edit the custom field for each task state.
7. Click the **Save** button.
8. Click the **Message Type Permissions** tab.
9. Define which users can view and edit custom field using the **Add Message** pulldown.

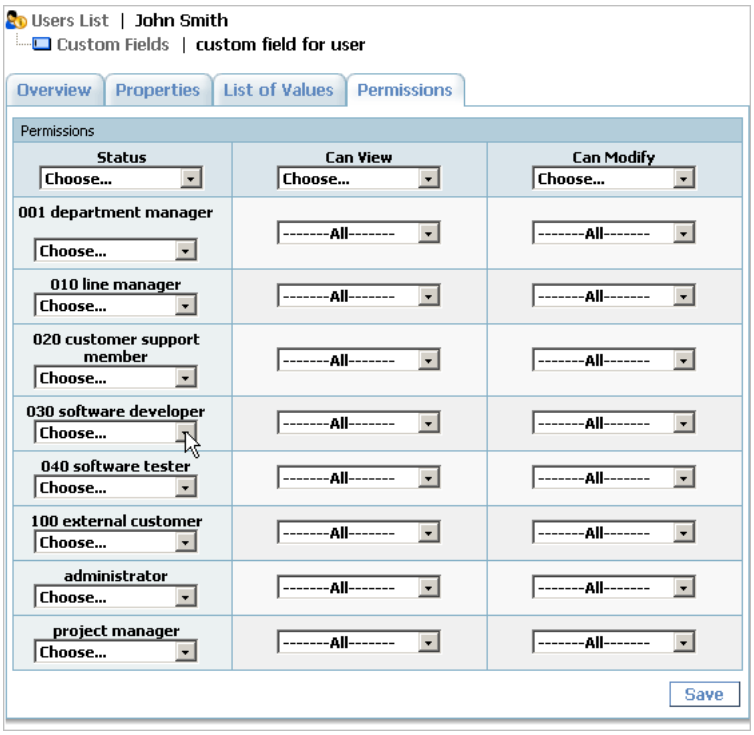
10. Click the **Save** button.





To define who can view or edit a custom field connected to a user:

1. Select the user that owns the custom field.
2. Click the **Current User -> Custom Fields...** menu item.
3. Choose a custom field.
4. Click the **Permissions** tab.
5. Define which users can view and edit the custom field.
6. Click the **Save** button.



See Also

- Custom Field Concepts (see page 54)
- User Status Concepts (see page 53)

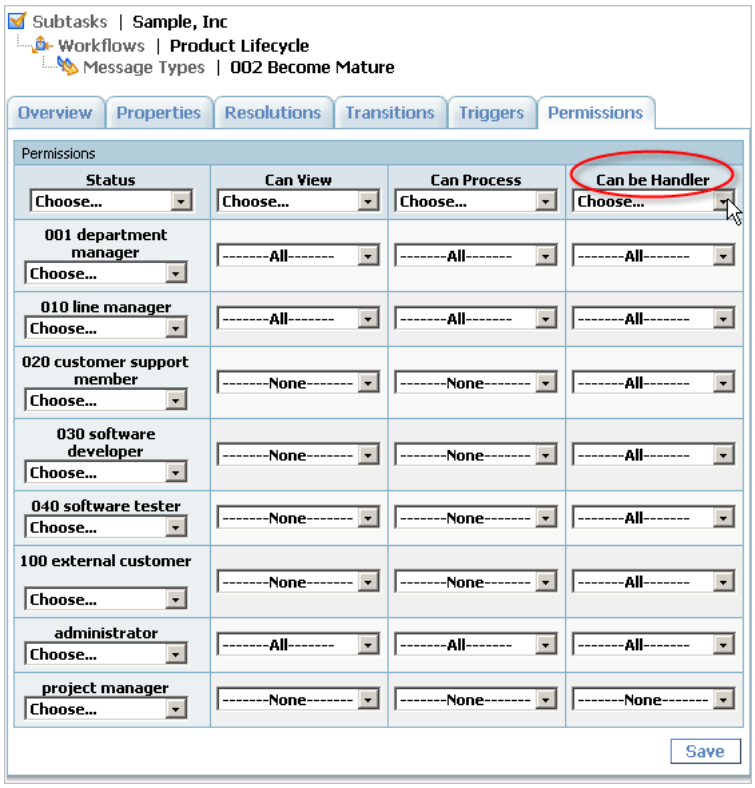
3.4.3.5 Restricting Handler List

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

Description

To restrict handler list:

1. Click the **Current Task -> Workflows...** menu item.
2. Choose the workflow for which you want to restrict the handler list.
3. Click the **Message Types** tab.
4. Select one of the message types in the list.
5. Click the **Permissions** tab.
6. Set the **Can be Handler** permission.
7. Click the **Save** button.



Conditions

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

See Also

- Creating a Workflow (see page 120)

3.4.3.6 Setting Default Handler for a Project

The following 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 62)

3.4.3.7 Restricting Who Can Close Task

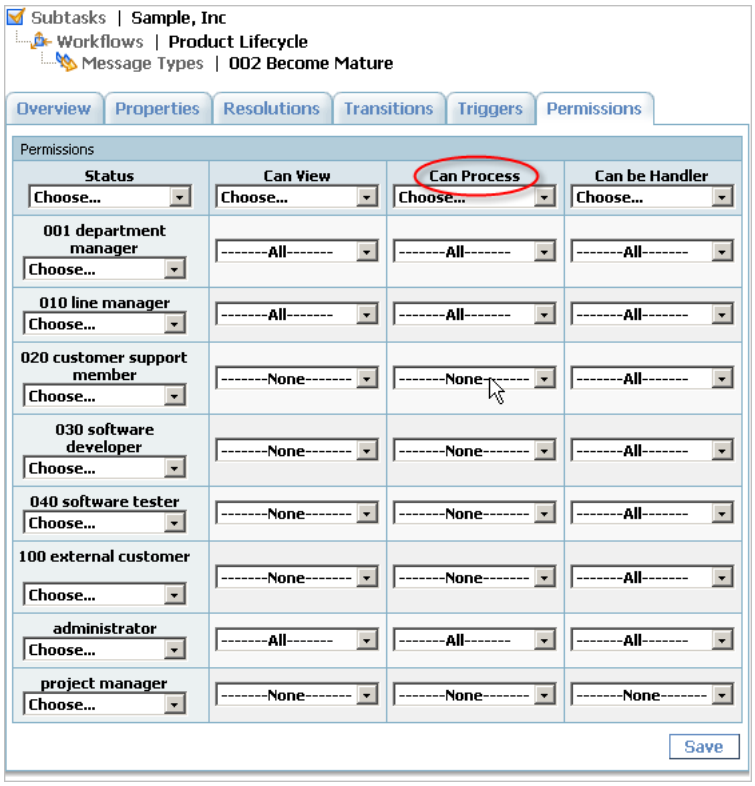
The following topic describes how to restrict access for closing tasks.

Description

To define which users can close tasks:

1. Click the **Current Task -> Workflows...** menu item.
2. Choose the workflow for which you want to restrict access for task closure.

3. Click the **Message Types** tab.
4. For each message type that moves tasks to the final state:
   1. Click the message type.
   2. Click the **Permissions** tab.
   3. Set the **Can Process** permission.
   4. Click the **Save** button.



See Also

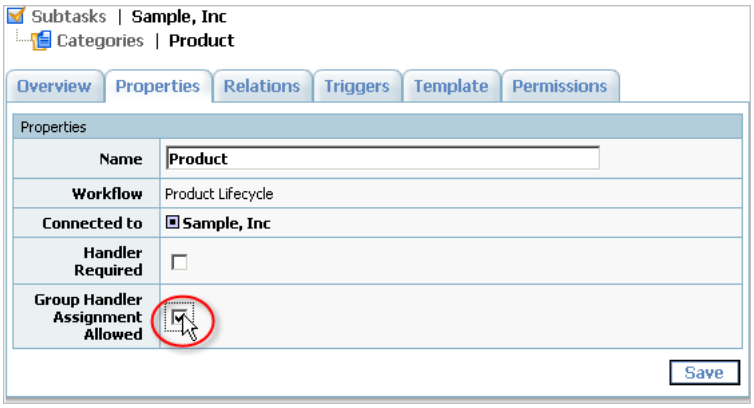
- Closing a Bug (see page 77)
- Creating a Workflow (see page 120)

3.4.3.8 Allowing Group Handler Assignment

The following topic describes how to allow task assignment to user status.

Description

1. Use the **Current Task -> Access Control Rules...** menu item to add user statuses that can be assigned.
2. Use the **Current Task -> Categories...** tab to set **Group Handler Assignment Allowed** for required categories.



3.4.3.9 Defining Which Users Can Create, Edit or Delete Tasks

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

Description

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

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



See Also

- User Status Concepts (see page 53)

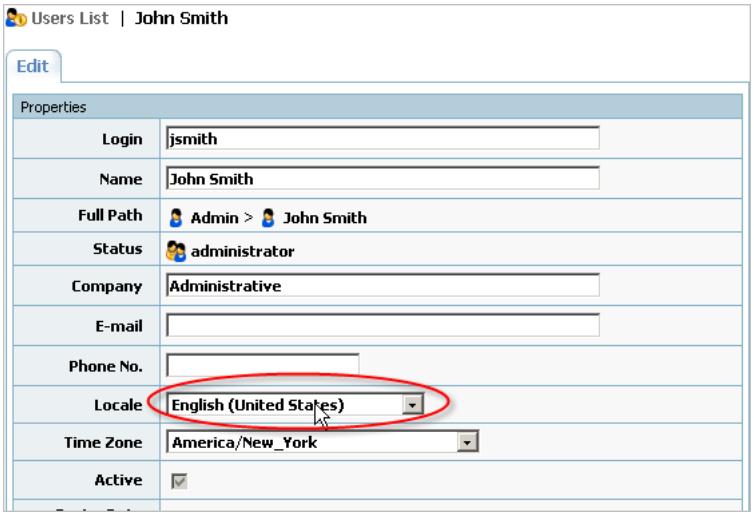
3.4.3.10 Choosing Localized Interface

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

Description

To choose the localized version of user interface:

1. Click the **Current User -> User...** menu item.
2. Click the **Edit** link.
3. Choose **Locale**.
4. 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 148)

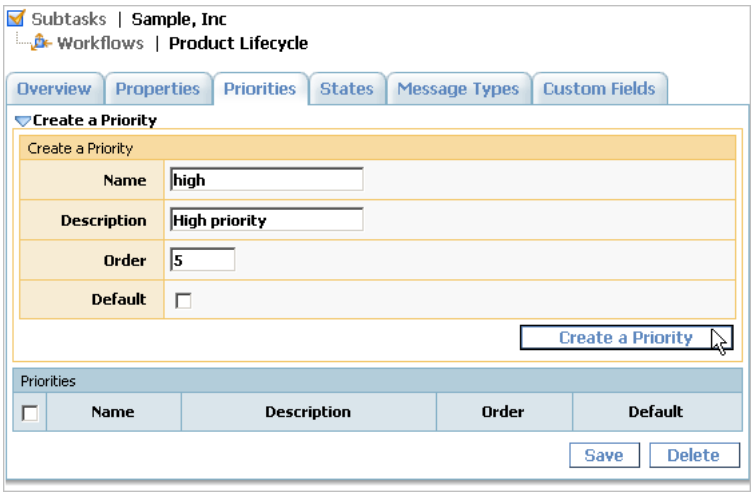
3.4.3.11 Defining Task Priorities

The following topic describes how to define available task priorities.

Description

To define available task priorities:

1. Click the **Current Task -> Workflows...** menu item.
2. Select a workflow from the list.
3. Click the **Priorities** tab.
4. Expand the **Create a Priority** pulldown.
5. Fill in priority parameters.
6. Click the **Create a Priority** button.



Notes

- Use priorities to specify the importance of the task.
- To set or change the priority of a task, edit it or add a message.

See Also

- Workflow Concepts (see page 51)

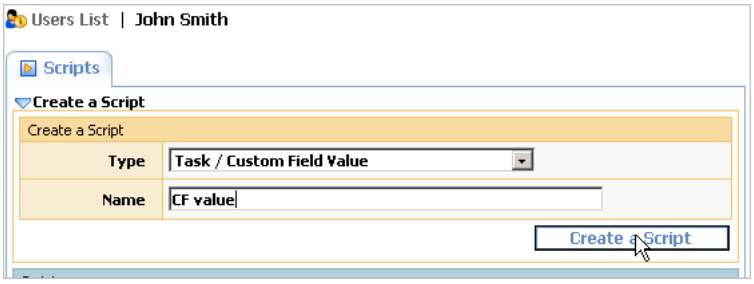
3.4.3.12 Creating a Script

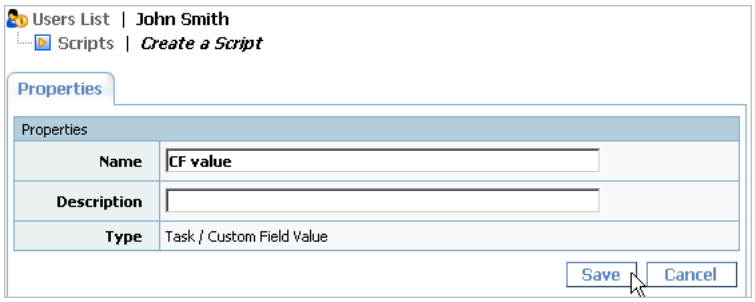
The following topic describes how to create a script.

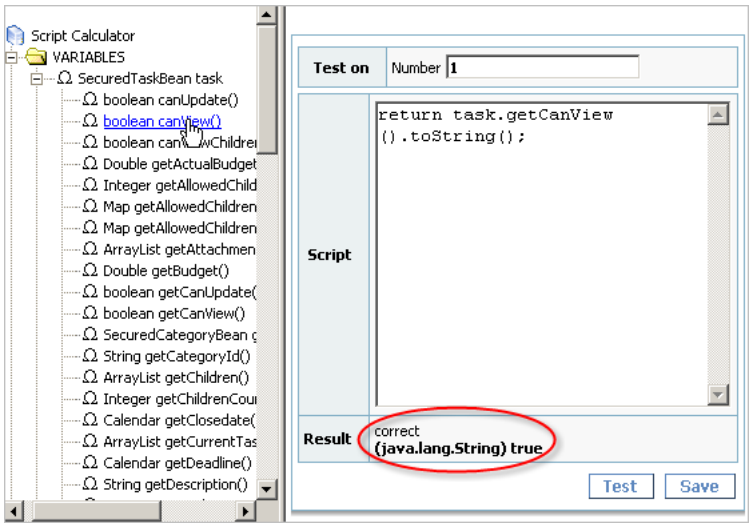
Description

To create a script:

1. Click the **Current User -> Scripts...** menu item.
2. Create a new script of the required type or select an existing one from the list.
3. Click the **Properties** tab.
4. Click the **Edit** button.
5. Type the text of the script in the popup window. Left-click a variable or a constant in the left section of the window to add it to the expression.
6. Click the **Save** button.







See Also

- Script Concepts (see page 54)

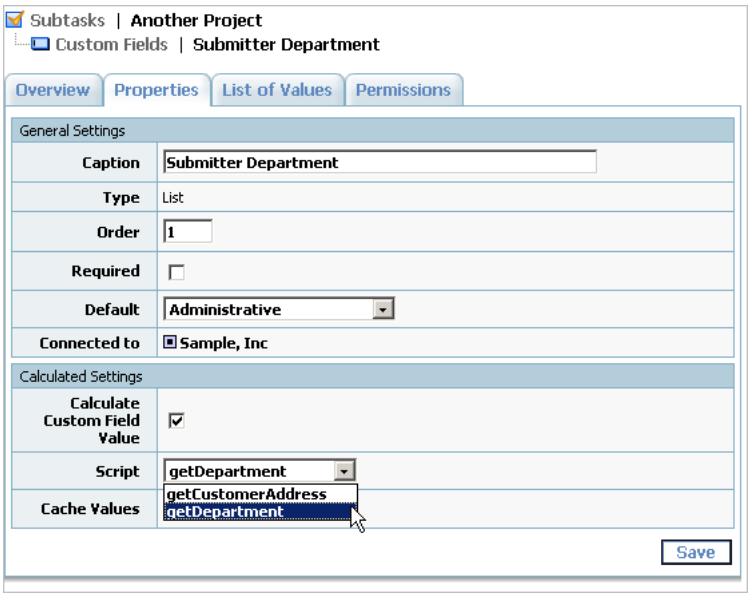
3.4.3.13 Calculating a Custom Field Value

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

Description

To calculate a custom field value:

1. Use the **Current User -> Scripts...** menu item to create **Task / Custom Field Value** or **User / Custom Field Value** script.
2. 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 / Custom Field Value** script:

```
Calendar ca = task.getSubmitdate();
ca.setTimeZone(TimeZone.getTimeZone(sc.getTimezone()));
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 / Custom Field Value** 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 time for all not-closed tasks, use the following **Task / Custom Field Value** script:

```
public double getAbudget(Object t) {
  double d = 0;
  if(t.getActualBudget() != null && t.getClosedate() == null && t.getChildrenCount() == 0)
    d = t.getActualBudget().doubleValue();
  for(Iterator it = t.getChildren().iterator(); it.hasNext();)
    d += getAbudget(it.next());
  d += t.getActualBudget().doubleValue()-d;
  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 / Custom Field Value** script:

```
String s = "";
ArrayList udfs = task.getUDFValuesList();
for(Iterator it = udfs.iterator(); it.hasNext();) {
    SecuredUDFValueBean udf = (SecuredUDFValueBean)it.next();
    s += udf.getCaption() + ":";
    Object value = udf.getValue(task);
    s +=  ((value!=null) ? value : "null") + "|";
}
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 / Custom Field Value** script:

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

Associated custom field properties:

Property	Value
Type	Integer

Example 6

To show favorite links use the following **Task / Custom Field Value** script:

```
return "<table>"+
"<tr><th>Site</th><th>URL</th></tr>"+
"<tr><td>Google</td><td><a href=\"http://www.google.com\">Click here</a></td></tr>"+
"<tr><td>MSN</td><td><a href=\"http://www.msn.com\">Click here</a></td></tr>"+
"</table>";
```

Associated custom field properties:

Property	Value
Type	Memo
HTML View	checked

Example 7

To get the logged-in user's login id, use the following **Task / Custom Field Value** script:

```
ArrayList arr = new ArrayList();
arr.add(sc.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 String 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*

Example 10

To calculate the number of months between the task submit date and the current date, use the following **Task / Custom Field Value** script. It can be used to find tasks submitted this month, last month, etc.

```
import java.util.Date;
Calendar ca = Calendar.getInstance();
ca.setTime(new Date(task.getSubmitdate().getTime().getTime()));
int submitMonth = ca.get(Calendar.MONTH);
int submitYear = ca.get(Calendar.YEAR);
```

```
ca.setTime(new Date());
int currentMonth = ca.get(Calendar.MONTH);
int currentYear = ca.get(Calendar.YEAR);
return (currentYear - submitYear)*12 + currentMonth - submitMonth;
```

Associated custom field properties:

Property	Value
Type	Integer

Example 11

To get the value of the task- or workflow-based custom field *Order Availability*:

```
String availability = AdapterManager.getInstance().getSecuredUDFAdapterManager()
     .getTaskUDFValue(sc,task.getId(),"Order Availability");
if (availability == null)
   return "Unknown";
else
   return availability;
```

To get the value of the user-based custom field *Address*:

```
String address = AdapterManager.getInstance().getSecuredUDFAdapterManager()
     .getUserUDFValue(sc,task.getSubmitter().getId(),"Address");
if (address == null)
   return "Unknown address";
else
   return address;
```

Associated custom field properties:

Property	Value
Type	String

Example 12

To show the link to attachments for tasks, use the following **Task / Custom Field Value** script:

```
if (task.hasAttachments()) {
   StringBuffer rez = new StringBuffer();
   for (Iterator it = task.getAttachments().iterator(); it.hasNext();) {
      SecuredAttachmentBean att = (SecuredAttachmentBean) it.next();
      rez.append("<a target=\"blank\" href=\"");
      rez.append(com.trackstudio.startup.Config.getInstance().getSiteURL());
      rez.append("/DownloadAction.do?method=download&session=");
      rez.append(task.getSecure().getId());
      rez.append("&id=");
      rez.append(att.getId());
      rez.append("\">");
      rez.append(att.getName());
      rez.append("</a>");
      rez.append("<br />");
   }
   return rez.toString();
} else
   return "";
```

Associated custom field properties:

Property	Value
Type	String
HTML View	checked
Cache Values	unchecked

See Also

- Custom Field Concepts (see page 54)

- Script Concepts (see page 54)
- Creating a Script (see page 103)
- Adding a Custom Field (see page 94)
- Hiding Unused Fields (see page 91)
- Custom Field Properties (see page 136)

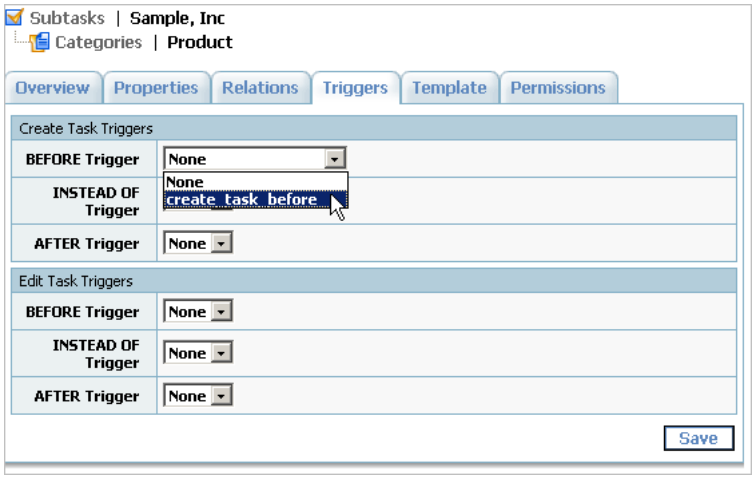
3.4.3.14 Implementing Triggers

The following topic describes how to use triggers to customize task processing.

Description

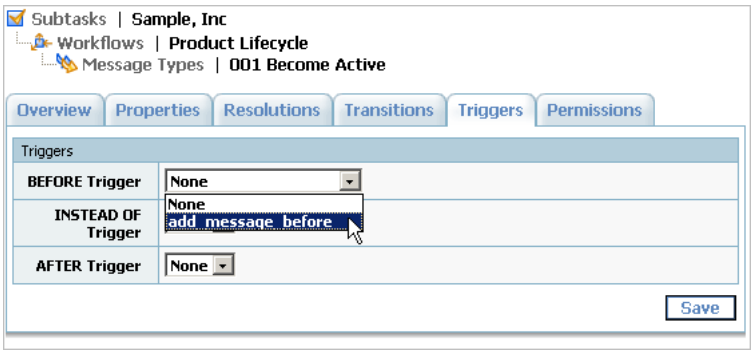
To map triggers to Create Task and Edit Task action:

1. Use the **Current User -> Scripts...** menu item to create **Trigger / Create Task / *** or **Trigger / Edit Task / *** script.
2. Click the **Current Task -> Categories...** menu item.
3. Choose the task category.
4. Click the **Triggers** tab.
5. Choose triggers for required actions.
6. Click the **Save** button.



To map triggers to Add Message action:

1. Use the **Current User -> Scripts...** menu item to create **Trigger / Add Message / *** script.
2. Click the **Current Task -> Workflows...** menu item.
3. Choose the workflow.
4. Click the **Message Types** tab.
5. Choose the message type.
6. Click the **Triggers** tab.
7. Choose triggers for required actions.
8. Click the **Save** button.



Create Task / BEFORE:

The following trigger complains if the **Deadline** task property has not been filled in for the task:

```
if(task.getDeadline() == null){
     throw new com.trackstudio.exception.UserMessageException("Please fill in the Deadline
task property");
}
return task;
```

The following trigger complains if the **Exists in Production** custom field value has not been filled in for the task or equals **Please Choose**:

```
HashMap udfs = task.getUdfValues();
if (udfs.get("Exists in Production")==null || udfs.get("Exists in
Production").equals("Please Choose"))
     throw new UserMessageException("No value selected for the Exists in Production
field!");
return task;
```

Create Task / INSTEAD OF:

The following trigger chooses a random handler for the task:

```
String group = task.getHandlerGroupId();
String handler= task.getHandlerUserId();

if (task.getHandlerGroupId()!=null){
  ArrayList hand = AdapterManager.getInstance().getSecuredStepAdapterManager()
     .getTaskEditHandlerList(task.getSecure(),  task.getId(), task.getCategoryId());
  int size = hand.size();

  if (size>0){
     int pointer = new Random().nextInt(size);
     handler= ((SecuredUserBean)hand.get(pointer)).getId();
     group = null;
  }
}

if (task.getParentId()!=null){
    SecuredTaskTriggerBean nb = new
SecuredTaskTriggerBean(task.getId(),task.getDescription(),
      task.getName(), task.getShortname(), task.getSubmitdate(), task.getUpdatedate(),
      task.getClosedate(), task.getActualBudget(), task.getBudget(), task.getDeadline(),
      task.getNumber(), task.getSubmitterId(), task.getHandlerId(), handler, group,
      task.getParentId(), task.getCategoryId(), task.getWorkflowId(), task.getStatusId(),
      task.getResolutionId(), task.getPriorityId(),task.getUdfValues(), task.getSecure());
    return nb.create(false);
} else return nb;
```

Create Task / AFTER:

The following script fills in the custom field *See Also* of type **Task**, with a list of tasks similar to the created one:

```
HashMap set = AdapterManager.getInstance().getSecuredTaskAdapterManager()
   .findSimilar(task.getSecure(),task.getId());
String similar="";
if (set!=null && !set.isEmpty()) {
   for (Iterator i = set.keySet().iterator(); i.hasNext();) {
```

```
      SecuredTaskBean s = (SecuredTaskBean) i.next();
      Float ratio = (Float) set.get(s);
      if (ratio.floatValue()>0.75f) {
         similar+="#"+s.getNumber()+"; " ;
      }
   }
}

if (similar.length()>0) {
   similar=similar.substring(0,similar.length()-2);
   AdapterManager.getInstance().getSecuredUDFAdapterManager().setTaskUDFValueSimple(
      task.getSecure(), task.getId(), "See Also", similar);
}
return task;
```

Edit Task/BEFORE:

The following trigger prevents users from saving a task with **Budgeted Time** less than **Actual Time**:

```
SecuredTaskBean currentState = new SecuredTaskBean(task.getId(),task.getSecure());
if (currentState.getActualBudget()!=null) {
   if ((task.getBudget()==null && currentState.getBudget()!=null)
      || (task.getBudget()!=null
           && task.getBudget().compareTo(currentState.getActualBudget())<0))
      throw new UserMessageException("Budgeted Time must be greater than Actual Time");
}
return task;
```

Edit Task/INSTEAD OF:

The following trigger creates messages of the type *log* for each deadline change using the **Current Task -> Task... -> Edit** link. Create message type *log* before using this trigger.

```
SecuredTaskBean oldOne = new SecuredTaskBean(task.getId(),task.getSecure());
Object oldDeadline=oldOne.getDeadline();
Object newDeadline = task.getDeadline();
String actionType=null;

if (oldDeadline==null && newDeadLine!=null) {
   actionType="Deadline was set.";
} else if (oldDeadline!=null && newDeadLine==null) {
   actionType="Deadline was reset.";
} else if (oldDeadline!=null && newDeadLine!=null) {
   if (newDeadline.after(oldDeadline)) {
      actionType="Deadline was extended.";
   } else if (newDeadline.before(oldDeadline)) {
      actionType="Deadline was moved up.";
   }
}

Object newOne = task.update(false);
if (actionType!=null) {
   SecuredMessageTriggerBean bean = new SecuredMessageTriggerBean(null,actionType,null,null,
      task.getDeadline(), task.getBudget(), task.getId(), task.getSecure().getUserId(),
null,
      task.getPriorityId(), task.getHandlerId(), task.getHandlerUserId(),
task.getHandlerGroupId(),
      CSVImport.findMessageTypeIdByName("log",
      task.getCategory().getName()), null, task.getSecure()).create();
}
return newOne;
```

Add Message / BEFORE:

The following trigger prevents users from closing a task that contains non-closed subtasks:

```
SecuredStatusBean ssb = AdapterManager.getInstance().getSecuredStepAdapterManager()
   .getNextStatus(message.getSecure(),
message.getTaskId(), message.getMstatusId());

if(ssb != null && ssb.isFinish()) {
   Set childredIdSet =message.getTask().getAllowedChildrenWithSubtasksMap().keySet();
   for(Iterator it = childredIdSet.iterator(); it.hasNext();) {
```

```
      String ch = (String)it.next();

      if (!AdapterManager.getInstance().getSecuredFindAdapterManager()
        .findTaskById(sc,ch).getStatus().isFinish()) {
         throw new UserMessageException("Subtask status is not final.");
      }
   }
}
return message;
```

Add Message / INSTEAD OF:

The following trigger implements the ability to vote for a task by adding a message. To use this script, create a custom field *votesCounter* of type **Integer** that will be used to count votes. Assign this trigger for a message type used to vote.

```
String udfValue = AdapterManager.getInstance().getSecuredUDFAdapterManager()
   .getTaskUDFValue(message.getSecure(), message.getTaskId(),"votesCounter");
String vote="0";
if (udfValue!=null && udfValue.length()>0)
   vote = udfValue;
int v = Integer.parseInt(vote);
v++;
AdapterManager.getInstance().getSecuredUDFAdapterManager().setTaskUDFValueSimple(
  message.getSecure(), message.getTaskId(),"votesCounter", v+"");
return message;
```

Add Message/AFTER:

The following trigger copies a task to the Knowledge Base (task #2 in this example) when user adds a message.

```
String copyTo = "2";
AdapterManager.getInstance().getSecuredTaskAdapterManager().pasteTasks(message.getSecure(),
   AdapterManager.getInstance().getSecuredTaskAdapterManager().findTaskIdByQuickGo(
     message.getSecure(), copyTo), new String[]{message.getTaskId()},"SINGLE_COPY");
return message;
```

The following trigger moves a task to the Knowledge Base (task #2 in this example) when user adds a message.

```
String moveTo= "2";
AdapterManager.getInstance().getSecuredTaskAdapterManager().pasteTasks(message.getSecure(),
     AdapterManager.getInstance().getSecuredTaskAdapterManager().findTaskIdByQuickGo(
       message.getSecure(), moveTo), new String[]{message.getTaskId()},"CUT");
return message;
```

Add Message / BEFORE:

The following trigger tracks custom field value changes in the message description:

```
if (message.getUdfValues()!=null && !message.getUdfValues().isEmpty()) {
    StringBuffer sb = new StringBuffer( 32 );
    for (Iterator it=message.getUdfValues().keySet().iterator();it.hasNext();) {
    String key = it.next().toString();
    String value = message.getUdfValues().get(key).toString();
    String oldValue = AdapterManager.getInstance().getSecuredUDFAdapterManager()
        .getTaskUDFValue(message.getSecure(),message.getTaskId(),key);
    if ((oldValue!=null && !oldValue.equals(value))
        || (oldValue==null && value!=null)) {
        if (sb.length() == 0) {
           sb.append("<b>");
        }
        sb.append("'").append(key).append(" changed from ")
          .append(oldValue).append(" to ").append(value).append("<br/>").append("\n");
    }
    }
    if (sb.length() >0) {
    sb.append("</b><br/><hr/><br/>\n").append(message.getDescription());

    return new SecuredMessageTriggerBean(message.getId(), sb.toString(),
                         message.getTime(), message.getHrs(), message.getDeadline(),
                         message.getBudget(), message.getTaskId(), message.getSubmitterId(),
                         message.getResolutionId(), message.getPriorityId(), null,
                         message.getHandlerUserId(), message.getHandlerGroupId(),
                         message.getMstatusId(), message.getUdfValues(),
message.getSecure());
```

```
    }
}
return message;
```

See Also

- Script Concepts (see page 54)
- Creating a Script (see page 103)

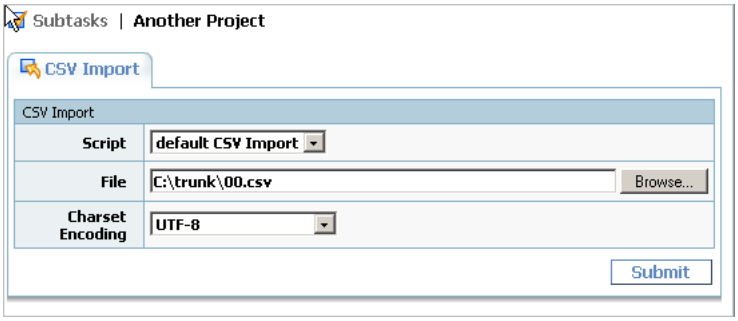
3.4.3.15 Importing Data from CSV File

The following topic describes how to import data from a CSV file.

Description

To import objects:

1. Use the **Current User -> Scripts...** menu item to create a **CSV Import** script.
2. Click the **Current Task -> CSV Import...** menu item.
3. Select the CSV Import **Script**.
4. Choose a CSV **File**.
5. Click the **Submit** button.



Example 1

CSV file example:

```
Category,Name,Status,Handler,Submit Date
Folder,“Sample, Inc”,active,Mike Clinton,10/19/04 2:35 PM
Software Bug,Can’t login,003 Closed,,10/19/04 2:50 PM
```

Use the following script to get the values of the attributes:

```
String categoryName = (String) inputMap.get("Category");
String taskName = (String) inputMap.get("Name");
String statusName = (String) inputMap.get("Status");
String handlerName = (String) inputMap.get("Handler");
String submitDateStr = (String) inputMap.get("Submit Date");
```

Use the following script to import a task from the specified file:

```
Map taskMap = new HashMap();
String locale = sc.getUser().getLocale();
DateFormatter df = new DateFormatter(sc.getUser().getTimezone(), locale);
taskMap.put(CSVImport.TASK_NAME, inputMap.get("Name"));

taskMap.put(CSVImport.TASK_CATEGORY_ID,
     CSVImport.findCategoryIdByName((String) inputMap.get("Category")));

taskMap.put(CSVImport.TASK_STATUS_ID,
     CSVImport.findStateIdByName((String) inputMap.get("Status"),
       (String) inputMap.get("Category")));

taskMap.put(CSVImport.TASK_HANDLER_USER_ID,
     CSVImport.findUserIdByName((String) inputMap.get("Handler")));
```

```
taskMap.put(CSVImport.TASK_SUBMIT_DATE, inputMap.get("Submit Date") != null ?
     df.parseToCalendar((String) inputMap.get("Submit Date")) : null);

return taskMap;
```

Example 2

The CSV file and the CSV Import script shown below demonstrate how to import custom fields of various types.

CSV file:

```
Name,Category,UserUDF,URLUDF,MultiListUDF,TaskUDF,IntegerUDF,ListUDF,DateUDF,FloadUDF,String
UDF,MemoUDF
"Products","Folder","jsmith;pdagley;slaw","http://www.trackstudio.com - TrackStudio",
     "multiListValue3;multiListValue4","#2;#4;#5","54","listValue2","12/16/05 3:38 PM",
     "5.6","stringUdf value","memo UDF value"
"Customer Support","Folder","cparmenter;smanske","http://localhost:8888/TrackStudio - local
TrackStudio instance",
     "multiListValue1;multiListValue2;multiListValue4","#15;#16;#18;#22;#23","33","listValue
2","12/30/05
3:40 PM",
     "43.6","str val2","memval2"
```

CSV Import script:

```
Map taskMap = new HashMap();
String locale = sc.getUser().getLocale();
DateFormatter df = new DateFormatter(sc.getUser().getTimezone(), locale);

taskMap.put(CSVImport.TASK_NAME, inputMap.get("Name"));

taskMap.put(CSVImport.TASK_CATEGORY_ID,
   CSVImport.findCategoryIdByName((String) inputMap.get("Category")));

Map udfMap = new HashMap();

if (inputMap.get("MemoUDF") != null)
    udfMap.put("MemoUDF", inputMap.get("MemoUDF"));

if (inputMap.get("DateUDF") != null && !inputMap.get("DateUDF").equals(""))
    udfMap.put("DateUDF", sc.getUser().getDateFormatter().parse(
        df.parseToCalendar((String)inputMap.get("DateUDF"))));

if (inputMap.get("FloadUDF") != null)
    udfMap.put("FloadUDF", inputMap.get("FloadUDF"));

if (inputMap.get("IntegerUDF") != null)
    udfMap.put("IntegerUDF", inputMap.get("IntegerUDF"));

if (inputMap.get("StringUDF") != null)
    udfMap.put("StringUDF", inputMap.get("StringUDF"));

if (inputMap.get("ListUDF") != null)
    udfMap.put("ListUDF", inputMap.get("ListUDF"));

if (inputMap.get("MultiListUDF") != null)
    udfMap.put("MultiListUDF", inputMap.get("MultiListUDF"));

if (inputMap.get("TaskUDF") != null)
    udfMap.put("TaskUDF", inputMap.get("TaskUDF"));

if (inputMap.get("UserUDF") != null)
    udfMap.put("UserUDF", inputMap.get("UserUDF"));

if (inputMap.get("URLUDF") != null)
    udfMap.put("URLUDF",inputMap.get("URLUDF"));

taskMap.put(CSVImport.TASK_UDF_MAP, udfMap);

return taskMap;
```

Example 3

The following script shows how several linked objects are imported. This script imports tasks and if there is no user with the task handler's name in the system, the script creates a user with the specified name:

```
Collection list = new ArrayList();

Map taskMap = new HashMap();
taskMap.put(CSVImport.TASK_NAME, inputMap.get("Name"));
taskMap.put(CSVImport.TASK_CATEGORY_ID, CSVImport.findCategoryIdByName((String)
inputMap.get("Category")));
list.add(taskMap);

String msgHandlerName = (String) inputMap.get("Message Handler");
String msgHandlerId = null;
Map userMap = new HashMap();
if (msgHandlerName != null && msgHandlerName.length() != 0) {
   msgHandlerId = CSVImport.findUserIdByName(msgHandlerName);
   if (msgHandlerId == null) {
       userMap.put(CSVImport.OBJECT_TYPE, CSVImport.USER_TYPE);
       userMap.put(CSVImport.USER_NAME, msgHandlerName);
       userMap.put(CSVImport.USER_LOGIN, msgHandlerName);
       userMap.put(CSVImport.USER_PRSTATUS_ID,
            CSVImport.findUserStatusIdByName("administrator"));
       userMap.put(CSVImport.USER_COMPANY, "Co.");
       list.add(userMap);
   }
}

Map messageMap = new HashMap();
messageMap.put(CSVImport.OBJECT_TYPE, CSVImport.MESSAGE_TYPE);
messageMap.put(CSVImport.MESSAGE_TASK_ID, taskMap);
if(msgHandlerId != null )
    messageMap.put(CSVImport.MESSAGE_HANDLER_USER_ID, msgHandlerId);
else
    messageMap.put(CSVImport.MESSAGE_HANDLER_USER_ID, userMap);
messageMap.put(CSVImport.MESSAGE_MESSAGE_TYPE_ID,
     CSVImport.findMessageTypeIdByName(
     (String)inputMap.get("Message Type"),
     (String)inputMap.get("Category")));
list.add(messageMap);

return list;
```

See Also

- Script Concepts (see page 54)
- CSV Import Concepts (see page 58)
- Creating a Script (see page 103)

3.4.4 Searching and Analyzing Tasks

3.4.4.1 Finding a Task

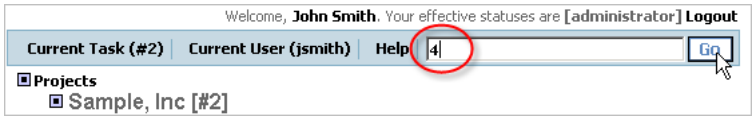
The following topic describes how to find a task.

Description

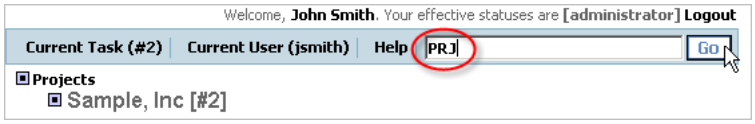
To find a task:

- By task number:
  1. Enter the task number into the input field in the right section of the **Main Menu**.

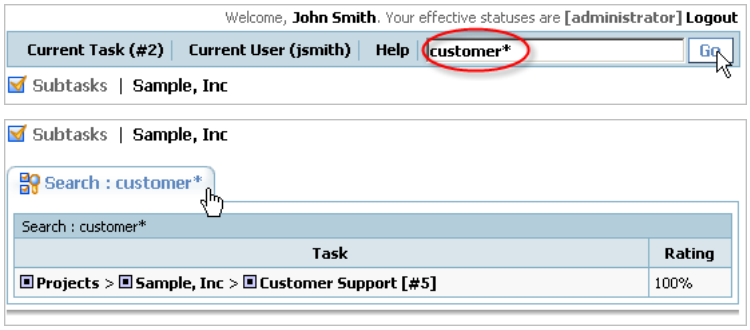
2. Click the **Go** button.



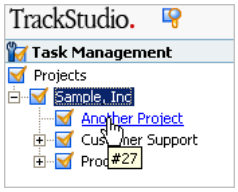
- By task alias:
  1. Enter the task alias into the input field in the right section of the **Main Menu**.
  2. Click the **Go** button.



- By content:
  1. Enter the keyword into the input field in the right section of the **Main Menu**.
  2. Click the **Go** button.



- Using the **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.



See Also

- Full Text Search Reference (see page 141)

3.4.4.2 Filtering Subtasks by Properties

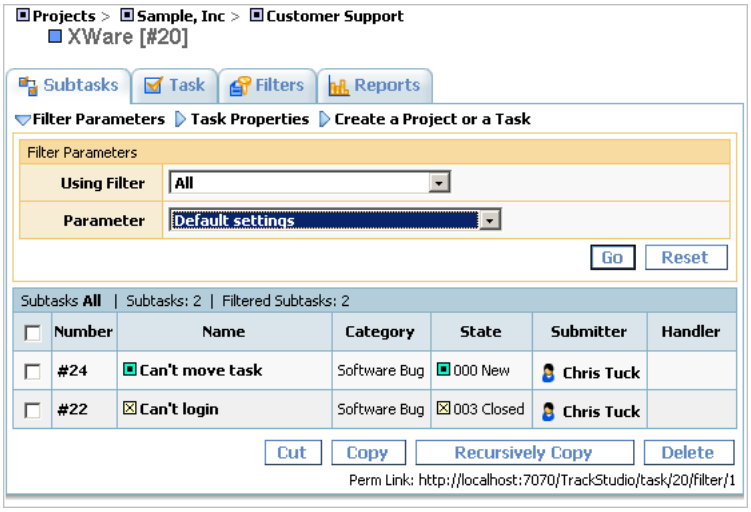
The following topic describes how to filter subtasks by properties.

Description

To filter subtasks by properties:

1. Click the **Current Task -> Filters...** menu item.
2. Choose an existing filter or create a new one.
3. Click the **Task Parameters** tab.

4. Fill in the filter conditions.
5. Click the **Save** button.
6. Click the **Current Task -> Subtasks...** menu item.
7. Use the **Filter Parameters** pulldown to switch the current filter.



Example 1

To filter all tasks for which you are the 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 134)

3.4.4.3 Filtering Tasks Using AND/OR/NOT

The following 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 **Current Task -> Subtasks...** menu item.
6. Use the **Filter Parameters** pulldown to choose your filter.

See Also

- Adding a Custom Field (see page 94)
- Calculating a Custom Field Value (see page 104)

3.4.4.4 Sorting Tasks

The following topic describes how to sort tasks by properties.

Description

To sort tasks by properties:

1. Click the **Current Task -> Filters...** menu item.
2. Choose an existing filter or create a new one.
3. Click the **Task Parameters** tab.
4. Set the order direction and the order level.
5. Click the **Save** button.
6. Click the **Current Task -> Subtasks...** menu item.
7. Use the **Filter Parameters** pulldown to choose your filter.

To temporarily change the sort order:

1. Click the **Current Task -> Subtasks...** menu item.
2. Click the column header.

3.4.4.5 Sorting Tasks by Product

The following topic describes how to sort tasks by product name.

Description

To sort tasks by product (parent project) name:

1. Click the **Current Task -> Filters...** menu item.
2. Choose an existing filter or create a new one.
3. Click the **Task Parameters** tab.
4. Set the **Relative Path** checkbox and set the sorting order for this field.
5. Click the **Save** button.
6. Click the **Current Task -> Subtasks...** menu item.
7. Use the **Filter Parameters** pulldown to choose your filter.

See Also

- Task Filter Properties (see page 138)

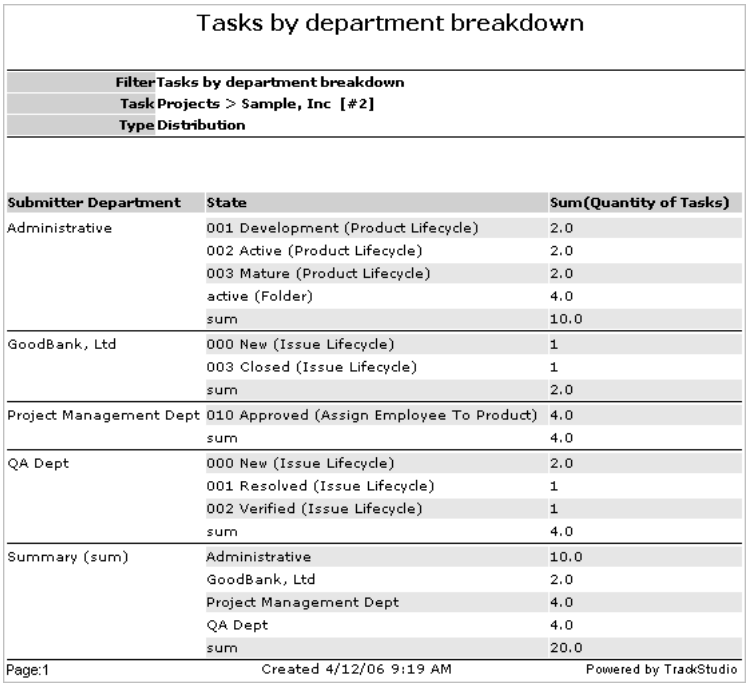
3.4.4.6 Analyzing Tasks Distribution

The following topic describes how to use distribution reports to analyze task distribution.

Description

To analyze task distribution:

1. Click the **Current Task -> Reports...** menu item.
2. Create a new **Distribution** report.
3. Enter the report parameters.
4. Click the **Generate Report** button.



Tasks by department breakdown

Filter	Tasks by department breakdown
Task	Projects > Sample, Inc [#2]
Type	Distribution

Submitter Department	State	Sum(Quantity of Tasks)
Administrative	001 Development (Product Lifecycle)	2.0
	002 Active (Product Lifecycle)	2.0
	003 Mature (Product Lifecycle)	2.0
	active (Folder)	4.0
	sum	10.0
GoodBank, Ltd	000 New (Issue Lifecycle)	1
	003 Closed (Issue Lifecycle)	1
	sum	2.0
Project Management Dept	010 Approved (Assign Employee To Product)	4.0
	sum	4.0
QA Dept	000 New (Issue Lifecycle)	2.0
	001 Resolved (Issue Lifecycle)	1
	002 Verified (Issue Lifecycle)	1
	sum	4.0
Summary (sum)	Administrative	10.0
	GoodBank, Ltd	2.0
	Project Management Dept	4.0
	QA Dept	4.0
	sum	20.0

Page:1 Created 4/12/06 9:19 AM Powered by TrackStudio

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 Time*
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 Quantity*
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 / Custom Field Value** script

```
Calendar ca = task.getSubmitdate();
ca.setTimeZone(TimeZone.getTimeZone(sc.getTimezone()));
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* of type **List**, 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 Quantity*
Function	*Sum*

See Also

- Script Concepts (see page 54)
- Generating a Report (see page 72)
- Calculating a Custom Field Value (see page 104)

3.4.5 Managing Categories and Workflows

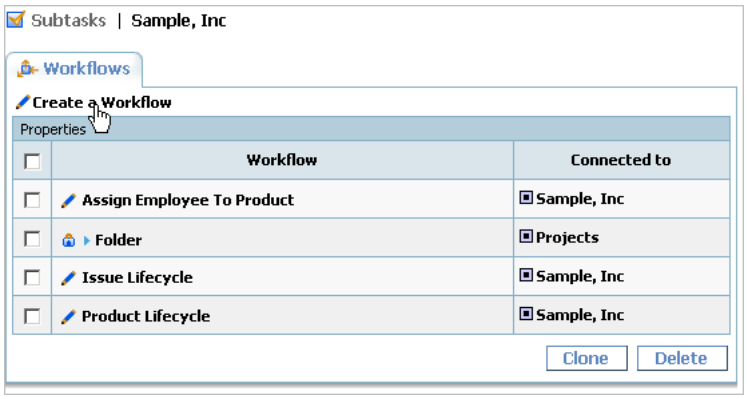
3.4.5.1 Creating a Workflow

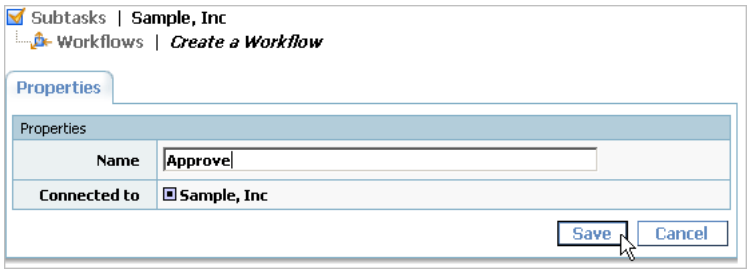
The following topic describes how to create a workflow.

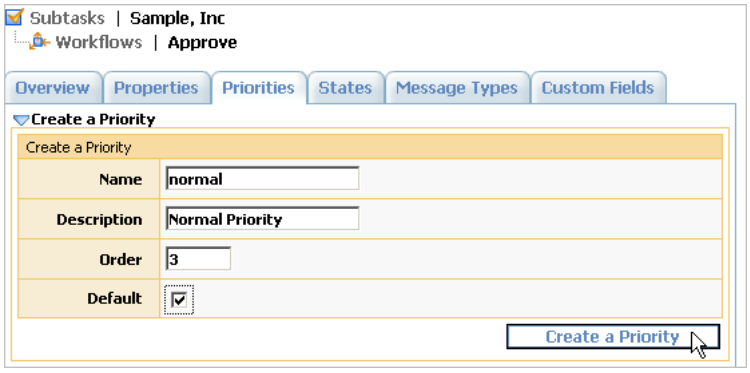
Description

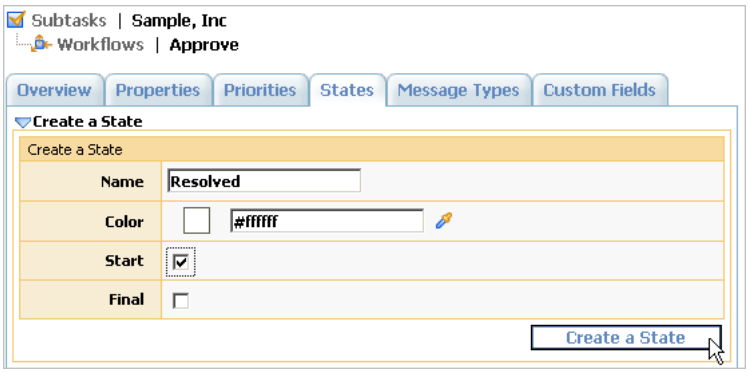
To create a workflow:

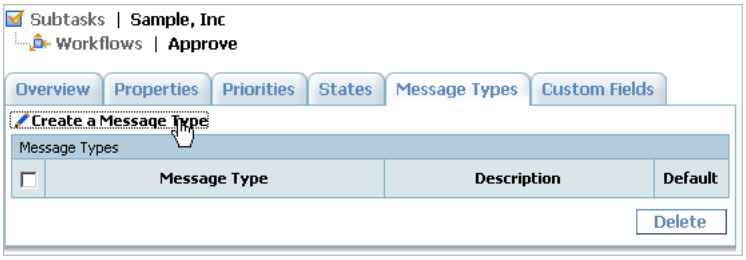
1. Click the **Current Task - > Workflows...** menu item.
2. Click the **Create a Workflow** link.
3. Enter the name for a new workflow into the input field.
4. Click the **Save** button.
5. Click the **Priorities** tab to create priorities for a task.
6. Click the **States** tab to create states for a task.
7. Click the **Message Types** tab to create required message types to move a task from one state to another.
    1. **Optional:** Use the **Resolutions** tab to set available resolutions for each message type.
    2. Use the **Transitions** tab to specify transitions between task states.
    3. Use the **Permissions** tab to 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.
8. Create workflow-based custom fields if required.

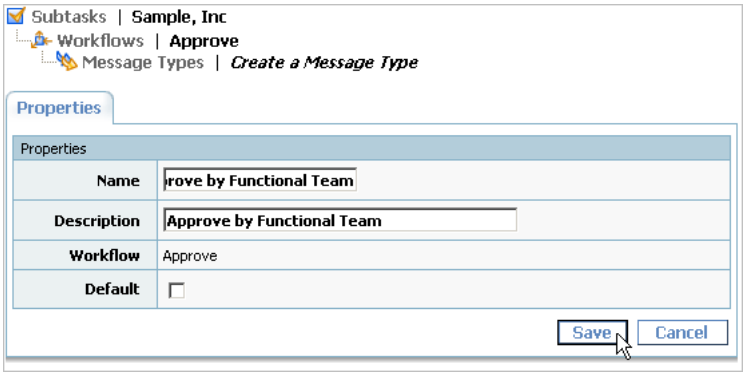


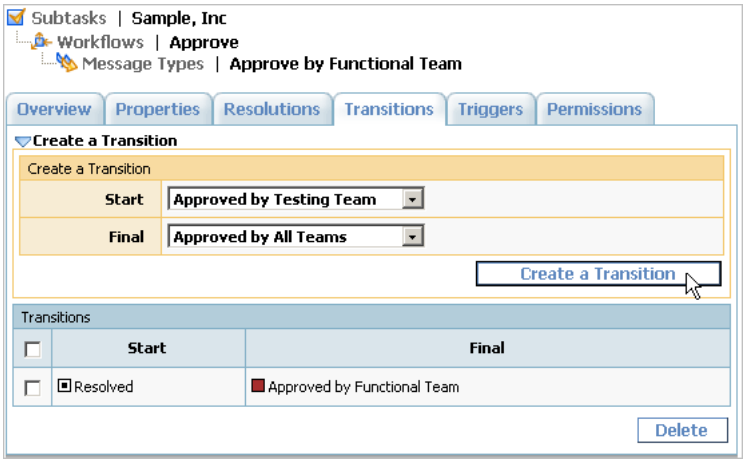


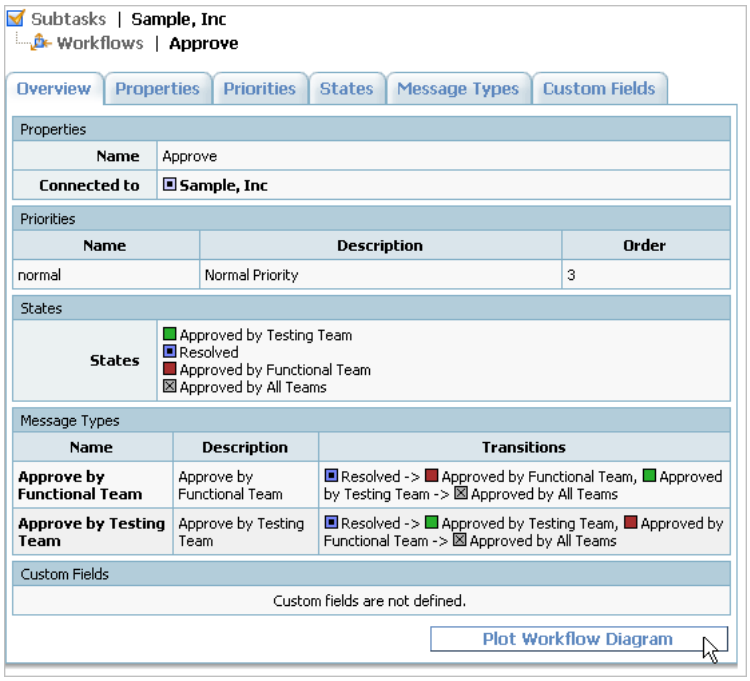


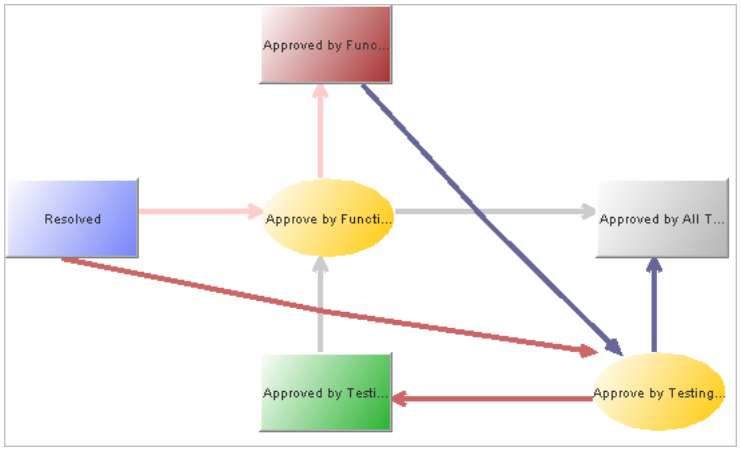

Subtasks | Sample, Inc
Workflows | Approve
Overview
Properties
Priorities
States
Message Types
Custom Fields
Create a State
Create a State
Name
Resolved
Color
#ffffff
Start
Final
Create a State


Subtasks | Sample, Inc
Workflows | Approve
Overview
Properties
Priorities
States
Message Types
Custom Fields
Create a Message Type
Message Types
Message Type
Description
Default
Delete


Subtasks | Sample, Inc
Workflows | Approve
Message Types | Create a Message Type
Properties
Properties
Name
rove by Functional Team
Description
Approve by Functional Team
Workflow
Approve
Default
Save
Cancel


Subtasks | Sample, Inc
Workflows | Approve
Message Types | Approve by Functional Team
Overview
Properties
Resolutions
Transitions
Triggers
Permissions
Create a Transition
Create a Transition
Start
Approved by Testing Team
Final
Approved by All Teams
Create a Transition
Transitions
Start
Final
Resolved
Approved by Functional Team
Delete





Example

This examples describes how to configure parallel approval of the task by several approvers. For example, this lets a developer submit code, and have a functional team perform unit testing while testers check the code simultaneously. The deliverable can go to the next level only if it gets two approvals.

To configure a workflow that implements a parallel approval process:

1. Use the **Current User -> Statuses...** menu item to create new user statuses for functional and testing teams.
2. Use the **Current Task -> Workflows...** menu item to create a new workflow.
3. Create the following states in your workflow:

State	Description
Resolved	Start state. Nobody has approved the bug yet.
Approved by Functional Team	Deliverable was tested.
Approved by Testing Team	Deliverable was checked.
Approved by All Teams	Deliverable was both tested and checked.

4. Create the following message types in your workflow, define transitions and permissions.

Message type	Transitions	Permissions
Approve by Functional Team	*Resolved -> Approved by Functional Team* *Approved by Testing Team -> Approved by All Teams*	Members of *Functional Team* only.
Approve by Testing Team	*Resolved -> Approved by Testing Team* *Approved by Functional Team -> Approved by All Teams*	Members of *Testing Team* only.

See Also

- Workflow Concepts (see page 51)
- User Status Concepts (see page 53)
- Creating a Category (see page 124)

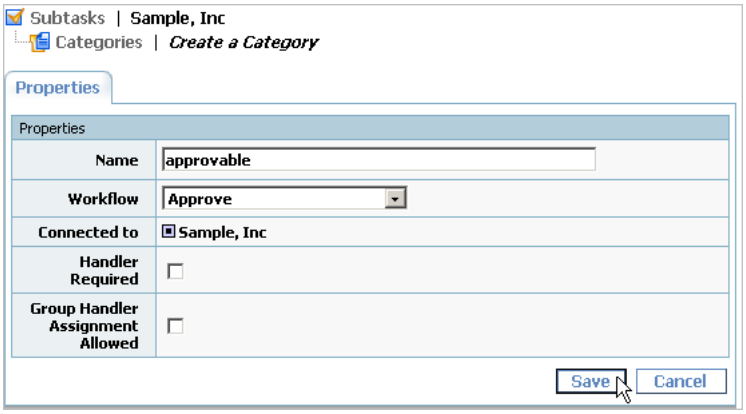
3.4.5.2 Creating a Category

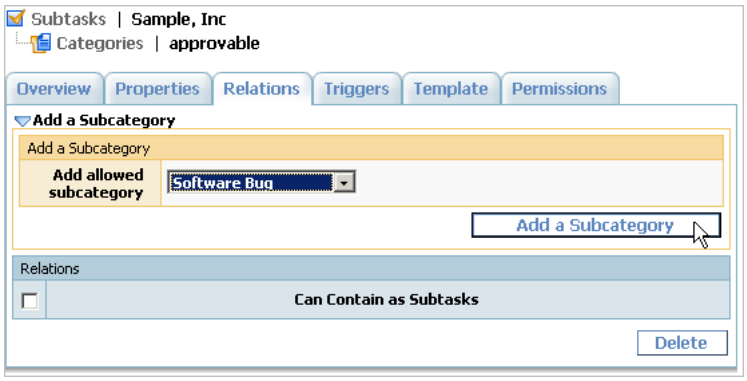
The following topic describes how to create a task category.

Description

To create a task category:

1. Click the **Current Task -> Categories...** menu item.
2. Click the **Create a Category** link.
3. Specify the name of the category.
4. Select the workflow to be used by tasks of this category.
5. Click the **Save** button.
6. **Optional:** Use the **Relations** tab to add possible subcategories for the tasks within the created category.
7. Use the **Permissions** tab to set user permissions for the tasks within the created category.
8. Use the **Relations** tab of the parent categories (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 52)
- User Status Concepts (see page 53)
- Defining Category Dependency (see page 125)

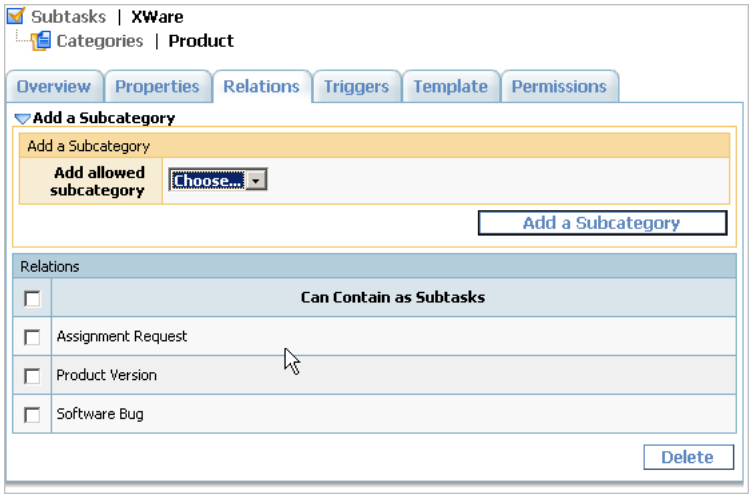
3.4.5.3 Defining Category Dependency

The following topic describes how to define category dependency.

Description

To define category dependency:

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



See Also

- Category Concepts (see page 52)

3.4.5.4 Setting Workflow for Task Category

The following topic describes how to set workflow for a task category.

Description

To set workflow for task category:

1. Click the **Current Task -> Categories...** menu item.
2. Select a category from the list.
3. Click the **Properties** tab.
4. Choose the workflow for your category.
5. 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 142)
- Workflow Concepts (see page 51)

3.4.6 Managing E-mail Notification and Submission

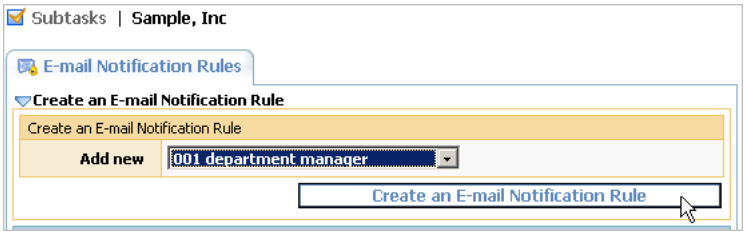
3.4.6.1 Receiving E-mail Notification when Tasks Change

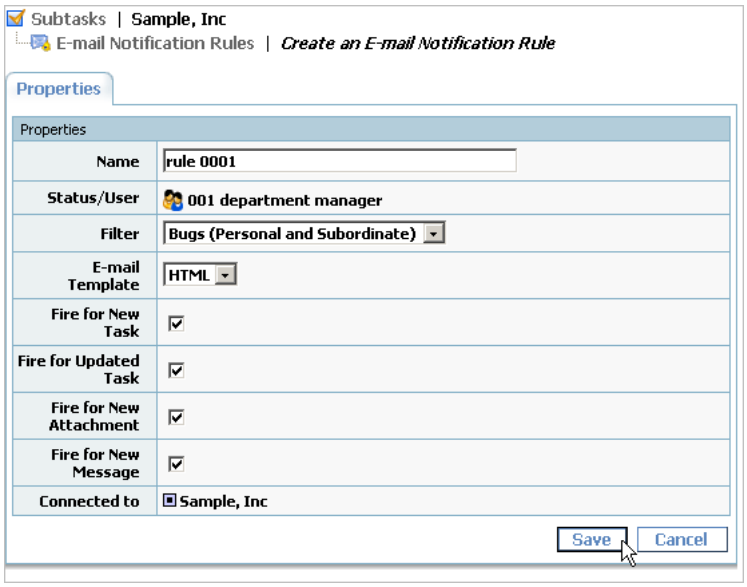
The following topic describes how to create an e-mail notification rule.

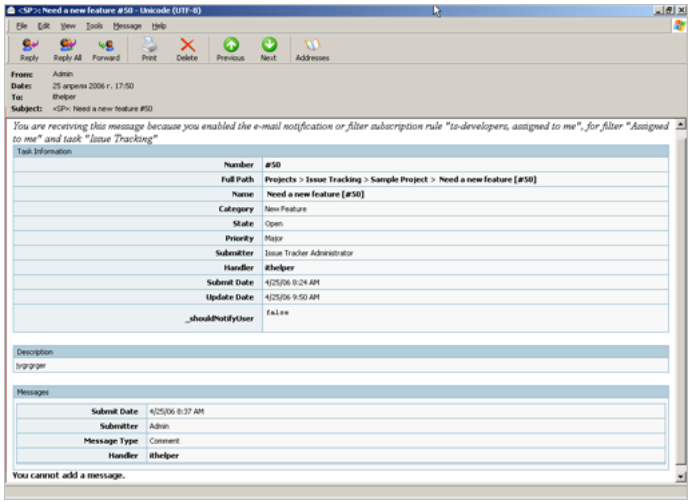
Description

To receive e-mail notification when a task changes:

1. Click the **Current Task -> E-mail Notification Rules...** menu item.
2. Expand the **Create an E-mail Notification Rule** pulldown.
3. Choose the e-mail notification recipient.
4. Click the **Create an E-mail Notification Rule** button.
5. Fill in e-mail notification rule setting**.**
6. 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 e-mail 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, use the following filter:

Section	Column	Condition
Message Parameters	**Submitter**	*is not Current User*

Example 2

To get e-mail notifications only if you are the handle of the task, use the filter:

Section	Column	Condition
Task Parameters	**Handler**	*is Current User*

Example 3

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

Section	Column	Condition
Task Parameters	**Priority**	*High*
Message Parameters	**Submitter**	*John Smith*

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 Parameters	**Handler**	*is Current User*
Message Parameters	**Submitter**	*is not Current User*

Example 5

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

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

java.util.Date oldDate = task.getSubmitdate().getTime();
long diff = 0;

for(Iterator it = messages.iterator();it.hasNext();) {
   java.util.Date newDate = ((Calendar)it.next()).getTime();
   diff = newDate.getTime() - oldDate.getTime();
   oldDate = newDate;
}
return diff/1000;
```

See Also

- Configuring E-mail Notification (see page 30)
- E-mail Notification Concepts (see page 56)
- E-mail Notification Rule Properties (see page 143)

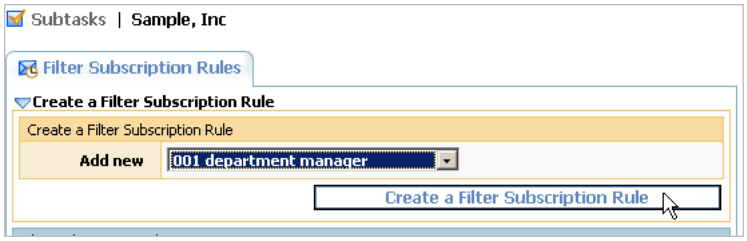
3.4.6.2 Periodically Receiving Lists of Tasks by E-mail

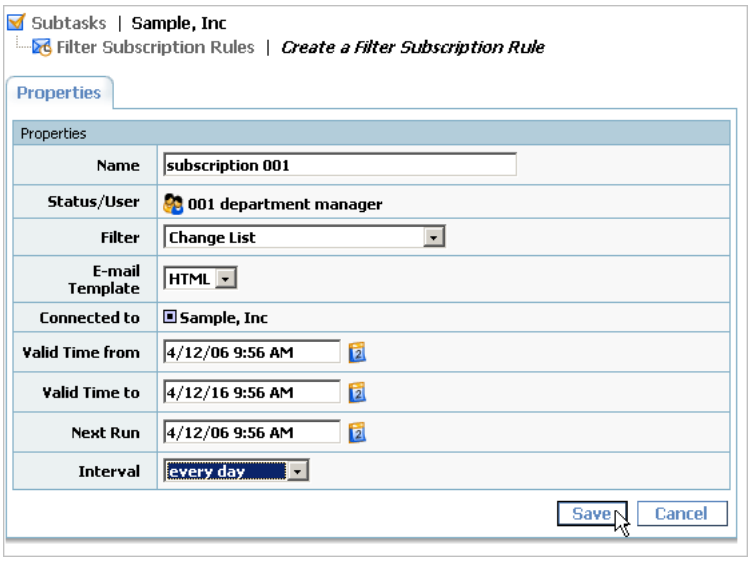
The following topic describes how to create a filter subscription rule.

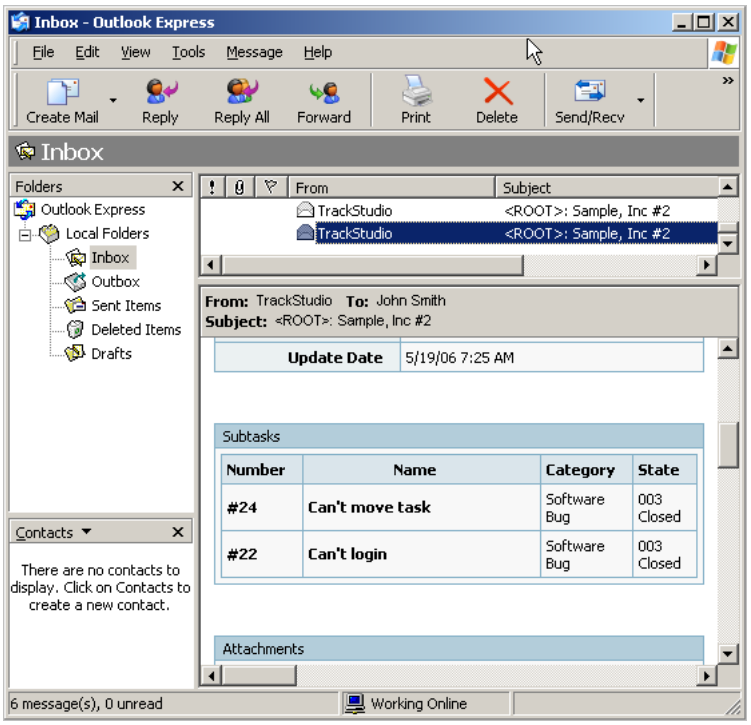
Description

To periodically receive a list of tasks by e-mail:

1. Choose the project or bug for which you want to enable the filter subscription.
2. Click the **Current Task -> Filter Subscription Rules...** menu item.
3. Expand the **Create a Filter Subscription Rule** pulldown.
4. Choose the e-mail notification recipient
5. Click the **Create a Filter Subscription Rule** button.
6. Fill in rule properties.
7. Click the **Save** button.







Notes

You will not receive notification if the filtered list is empty.

See Also

- Configuring E-mail Notification (see page 30)
- E-mail Notification Concepts (see page 56)
- Filter Subscription Rule Properties (see page 143)

3.4.6.3 Sending Alert for Overdue Tasks

The following 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 a filter subscription rule based on this filter.

See Also

- E-mail Notification Concepts (see page 56)
- Filtering Subtasks by Properties (see page 116)
- Periodically Receiving Lists of Tasks by E-mail (see page 128)

3.4.6.4 Adding a Task by E-mail

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

Before You Begin

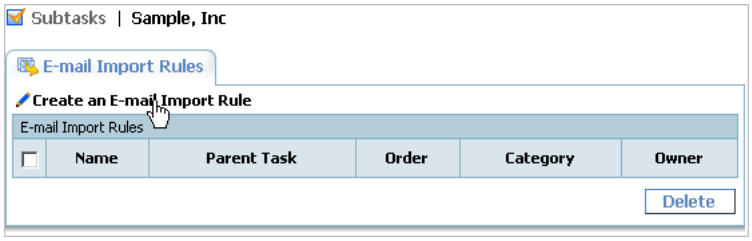
To create an e-mail import rule:

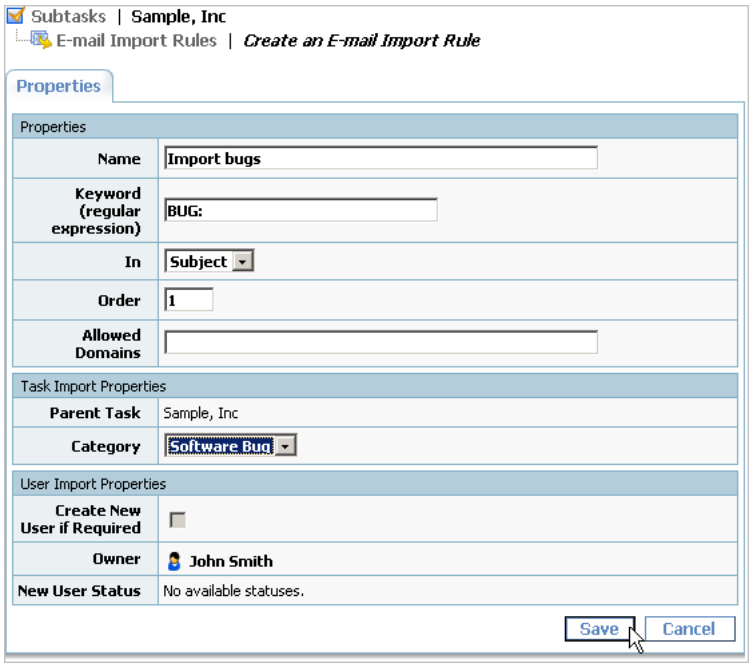
1. Choose the parent project for your tasks.
2. Click the **Current Task -> E-mail Import Rules...** menu item.
3. Click the **Create E-mail Import Rule** link.
4. **Optional:** To create a new user accounts when receiving e-mail from an unknown users, set the **Create New User if Required** checkbox and select the status for a new user. Use the **Current Task -> Access Control Rules...** menu item to add user statuses that can be used in e-mail import rules.
5. Fill in other e-mail import rule properties.
6. Click the **Save** button.

Description

To add a task by e-mail:

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





Conditions

- The user should be registered both in TrackStudio and in their e-mail client (Outlook, Outlook Express, etc) with the same name or e-mail, otherwise the **Create New User if Required** checkbox should be marked.
- 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 **trackstudio.mail.properties**)

See Also

- Configuring E-mail Submission (see page 31)
- E-mail Submission Concepts (see page 57)
- Granting Users Access to a Project (see page 83)
- E-mail Import Rule Properties (see page 144)
- Regular Expressions Overview

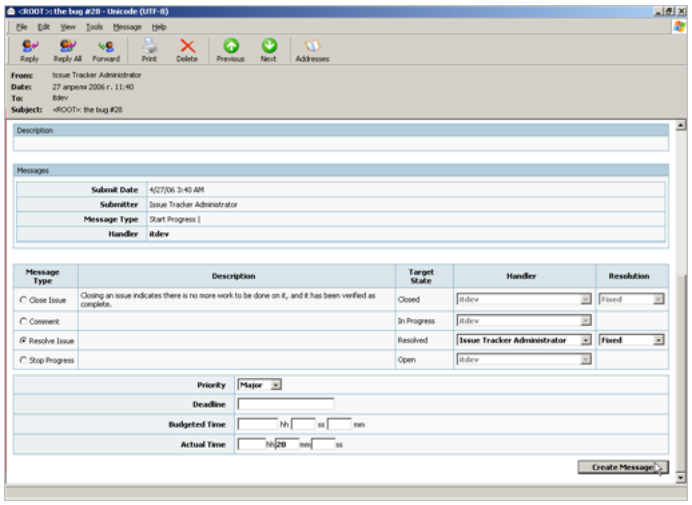
3.4.6.5 Adding a Message by E-mail

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

Description

To add a message using the 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 at the bottom of the 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.



To add a message of the default message type 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 and attach files.
4. **Optional:** To change the task handler, enter the name and the e-mail address of a new handler in the CC field.
5. Send an e-mail notification to the e-mail defined in TrackStudio e-mail settings (**mail.store.user property** in **trackstudio.mail.properties**).

Conditions

- The user should be registered both in TrackStudio and e-mail client properties with same name or e-mail.
- The 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 **trackstudio.mail.properties**).

See Also

- Configuring E-mail Submission (see page 31)
- E-mail Submission Concepts (see page 57)

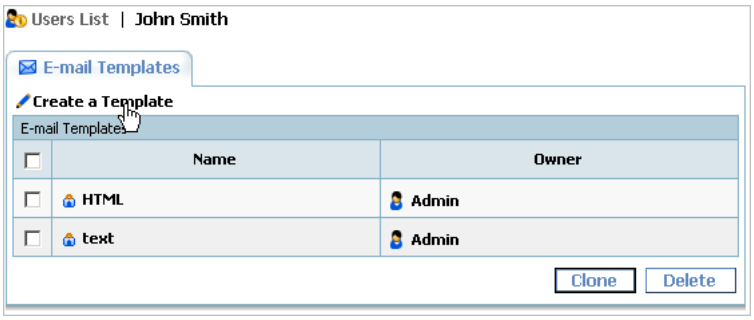
3.4.6.6 Customizing E-mail Notification Templates

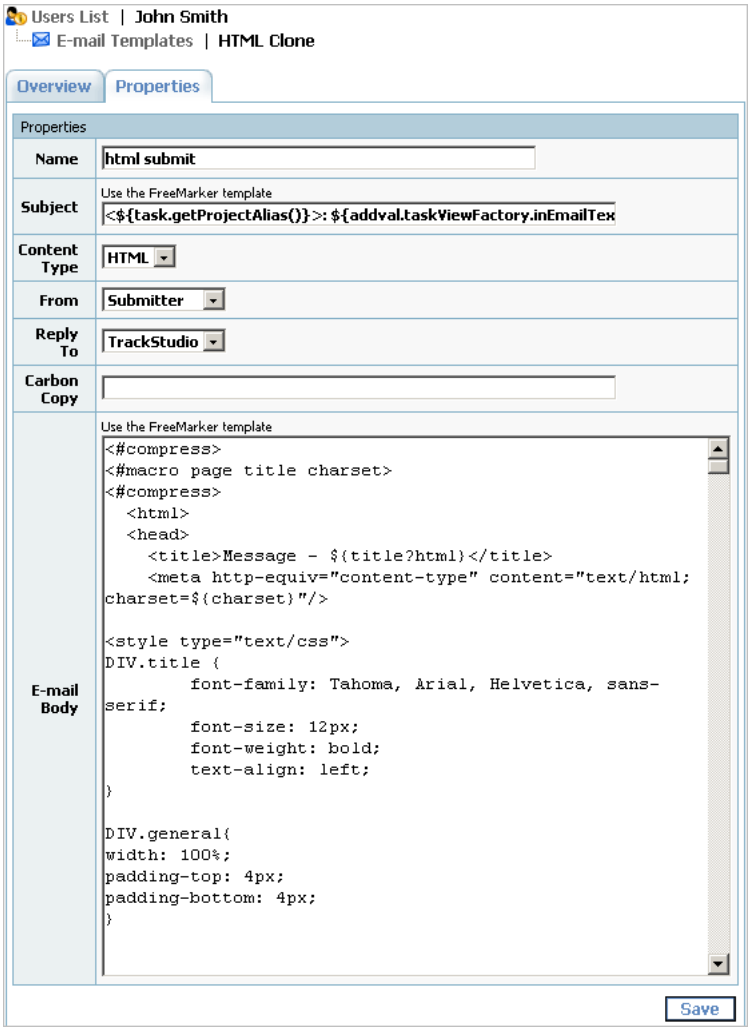
The following topic describes how to customize e-mail notification templates.

Description

To customize e-mail notification templates:

1. Click the **Current User -> E-mail Templates...** menu item.
2. Choose an existing e-mail template or create a new one.
3. Click **Properties** 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 56)
- E-mail Notification Rule Properties (see page 143)

3.5 Reference

3.5.1 Task Properties

Task properties:

Task Property	Description	Examples
Number	Unique numeric task number, which is automatically assigned, and not re-used even after a task is deleted.	*#23*
Full Path	Shows the full task path in the tasks hierarchy.	*Projects > Sample, Inc [#2]*
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 (i.e. the workflow) depend on the 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 or the user status who has been 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 dates of any of its subtasks.	*2005-01-01*
Close Date	Date/time when the task was closed (if the task is closed). Set for a task when a task's workflow is transitioned into a state designated as **Final**.	*2005/01/01*

Deadline	Date when the current task must be finished.	*2005/01/01*
Budgeted Time	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 Time	Elapsed task processing time (working time, not calendar time). The actual time for the project is the sum of actual times of current task and all its subtasks.	*3 hh 10 mm 05 ss*
Description	Use the task description, if its name is not informative enough.	

3.5.2 User Properties

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 (the user's 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*
Expiry 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 user who should no longer be able to use the system, for whatever reason (retirement from the company, and so on). Users marked inactive cannot login, and cannot be assigned tasks.	*Yes* *No*
Show Context Help	Use this flag to hide or show the help messages.	*Yes* *No*
HTML Editor	Allows you to enable or disable usage of HTML editor in the task description and message body fields. You cannot disable the HTML editor if you have edited the description of an existing task, or if you use description templates.	*Yes* *No*
Navigation Tree	Allows you to specify which users and tasks are to be shown in the **Navigation Tree**, or to hide the tree entirely.	*All tasks and users* *Projects and managers only* *None*

3.5.3 Message Properties

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	From the associated workflow, describes the action performed within a task.	*resolve, close*
Handler	User or user group who should now process this task.	*John Smith*
Resolution	New task "resolution" related to the current workflow state.	*fixed, not a bug*
Priority	New task priority.	*low, normal, high*
Deadline	Date when the current task must be finished.	*01/01/2005*
Budgeted Time	Estimated time (in hours) allocated for the task. This is working time, not calendar time.	*20 hh 30 mm*
Actual Time	Time spent on processing the task.	*10 hh 20 mm*
Description	Message comment.	

3.5.4 Custom Field Properties

General custom field properties:

Property	Description	Examples
Caption	Custom field name.	*platform* *release* *version* *customer name*
Type	Custom field type. Please refer to the table below, for more information about available custom field types.	*String* *Memo*
Order	An integer value which indicates the sorting position of the custom field on the user page.	*10* *20* *30*
Required	Indicates whether a field value is required or not. The users cannot save a task if they do not fill in all required properties.	*Yes* *No*
HTML View	Set this flag to allow your users use HTML formatting. This property is available for **String** and **Memo** custom fields only.	*Yes* *No*
Default	Default field value.	*100*

Properties, used for calculated custom fields:

Property	Description	Examples
Calculate Custom Field Value	Set to mark field as calculated.	*Yes* *No*
Script	The script used to calculate custom field value.	*getCustomerAddress*

Cache Values	Set this flag to improve calculated custom field performance. Mark if script uses only properties of the current task to calculate its value.	*Yes* *No*

Properties, used for String custom fields to fill in the lookup dropdown list with the possible values:

Property	Description	Examples
Show Lookup Dropdown	Mark to use the **Lookup Script** to fill in the dropdown list with the possible values of a custom field.	*Yes* *No*
Lookup Script	Specify the script that returns the list of possible values of a custom field.	*getDaysOfWeek*
Lookup Only	Mark to indicate that custom field cannot take any value except of returned by **Lookup Script**.	*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

3.5.5 Assigned Status Properties

Assigned status properties for tasks:

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's 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.

Assigned status properties for users:

Property	Description
User	The user who has this status assigned to him.

Assigned Status	User status for the user.
Override	Mark to override inherited statuses (both user's own status and assigned statuses for the parent users).
Connected to	The user that has this status assigned to it.
Owner	The user who assigned the status.

3.5.6 Task Filter Properties

General task 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 reports are visible only to the filter owner, while non-private filters are visible to all users who have access rights to the current task or its subtasks. If you have no access rights for a given 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 cannot modify or delete filters created by other owners, but you can use them for task filtering and email notification.	*John Smith*
Tasks per Page	Number of tasks per page.	*20*
Deep Search	Use the **Deep Search** checkbox to recursively filter tasks through the entire hierarchy beginning with the current task.	*Off*
Bulk Processing Tool	Use the **Bulk Processing Tool** to add messages to a number of tasks at once. To add a message using the **Bulk Processing Tool** you should specify non-empty **Message Description**.	*On*
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 on tasks that pass the filter.	`(software OR TrackStudio) AND bugs`

Task filter parameters:

Setting	Description	Example
Filter	Mark this checkbox to filter by task field.	*On*
Hide	Mark this checkbox to to hide the value from the 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 parameters:

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 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.	First 5
Filter Messages	Use to filter messages in tasks. If it is off, only tasks will be filtered.	Last 1

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 the number of messages that 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) Subsequently, 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.

3.5.7 User Filter Properties

General user 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 cannot modify or delete foreign filters, but you can use them for user filtering.	*John Smith*
Users per Page	Number 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 parameters:

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*

3.5.8 Report Properties

Report properties:

Property	Description	Example
Name	The report name.	*Tasks by department breakdown*
Type	The report layout type.	*Distribution*
Filter	The filter that is used to determine which tasks and messages will be included into the report.	*Tasks by department breakdown*
Filter Settings	Use this property to change specified filter settings.	
Private	Private reports are visible only to the report owner, while non-private reports are visible to all users who have access rights to the current task or its subtasks. If you have no access rights for a given task, you can create only private reports for it.	*Yes* *No*
Owner	The user who created the report.	*John Smith*
Connected to	Parent task for report. Report will be available for this task and its subtasks.	*Sample, Inc*
Group by, Subgroup by	These properties are available for **Distribution** reports only. Use them to specify which fields will be used to group the tasks.	*Category*
Value	The numeric field used for calculating the summary value. The summary value will be calculated both for each group and report. This property is available for **Distribution** report only.	*State*
Function	The function that will be applied to the **Value** field for calculating the summary value. This property is available for **Distribution** report only.	*Sum* *Avg* *Min* *Max*
Hours Format	The format of time displaying. This property is available for **UserWorkload** report only.	*hh:mm:ss*
Add Page Breaks	Mark to generate multi-page report.	

3.5.9 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
```


3.5.10 Category Properties

Category properties:

Column	Description
Name	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.
Handler Required	Mark to force users to select task handler when creating a task.
Group Handler Assignment Allowed	Mark to allow users to assign tasks to a user status.

3.5.11 E-mail Notification Rule Properties

E-mail notification rule properties:

Property	Description
Name	The name of the e-mail notification rule.
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 (*Handler="Current User"*).
Connected to	E-mail notification is activated for this task. If a user subscribes to e-mail notifications for a given project, notifications will be automatically activated for all subtasks of this project, and for both existing and new tasks.
Status/User	The recipient of e-mail 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.
Fire for New Task	Check to send an e-mail notification when new tasks are created.
Fire for Updated Task	Check to send an e-mail notification when existing tasks are updated.
Fire for New Attachment	Check to send an e-mail notification when an attachment is uploaded.
Fire for New Message	Check to send an e-mail notification when a new message is added.

3.5.12 Filter Subscription Rule Properties

Filter subscription rule properties:

Property	Description
Name	The name of the filter subscription rule.
Connected to	The task for which the user will periodically receive by e-mail, a filtered list of subtasks.
Status/User	User or user group which will receive subscription e-mail. You can subscribe yourself or your subordinate users.
E-mail Template	The template that will be used for filter subscription e-mail. A user can receive several e-mail notification messages that differ in their templates.
Valid Time	You can specify start (**From**) and stop (**To**) date of subscription. You will receive e-mail only when the current date is between the two.
Next Run	The next time the filter should be executed. Generally, you should not modify this field.
Interval	Select interval between filter subscription messages - from 30 minutes to 1 month.

3.5.13 E-mail Import Rule Properties

E-mail import rule properties:

Property	Description
Name	The name of the e-mail import rule.
Keyword (regular expression)	Specify the **Keyword** either in the header, subject or in the body of the e-mail. 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 content. The keyword check is case-insensitive.
In	Specify the field in which to search for the keyword. The **Subject, Body,** and **Header** options are supported.
Order	The order in which e-mail import rules will be checked.
Allowed Domains	Accept e-mail from specified domains only. Use **;** as a delimiter.

Task import rule properties:

Property	Value
Parent Task	The parent task of the created tasks.
Category	The category of the created tasks.

User import rule properties:

Property	Value
Create New User if Required	Check this option only if you are going to allow the creation of new users via e-mail.
Owner	The manager of the created users.
New User Status	Status of the created users. To choose **New User Status**, allow users of that status access the **Parent Task** using the **Current Task -> Access Control Rules...** menu item and ensure that users of this status can create tasks of the specified **Category**.

3.5.14 Self-registration Rule Properties

Self-registration rule properties:

Property	Description
Name	The name of the self-registration rule.
URL for Registration	Give this URL to your users or customers to register in TrackStudio using this self-registration rule.
Status	The status of users created via this rule.
Children 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.

Task	The project to which access for the created user will be granted, or the project where new user projects will be created.
Create a New Project for Each User	Check this option if self-registered users should be able only to view their own tasks and projects. In this case, a new user project is created and the new user is granted access to it.
Category	If **Create a New Project for Each User** is selected, define the **Category** that specifies the category of the new user project.

3.5.15 CSV Import Script Reference

The following topic describes objects used in **CSV Import** scripts.

Description

To create a task, put the following data into the resulting **Map**:

Key	Value Type	Default Value	Required	Description
OBJECT_TYPE	String	CSVImport.TASK_TYPE	No	Object type ID
TASK_NAME	String	"Not specified"	No	Task name
TASK_SHORTNAME	String	null	No	Task alias
TASK_DESCRIPTION	String	""	No	Task description
TASK_BUDGET	String or Double	null	No	Task budgeted time
TASK_DEADLINE	Calendar	null	No	Task deadline
TASK_PRIORITY_ID	String	Default priority	No	Task priority ID
TASK_HANDLER_USER_ID	String/Map	null	No	Task handler (user) ID/Map
TASK_HANDLER_GROUP_ID	String	null	No	Task handler (user status) ID
TASK_CATEGORY_ID	String	Must be defined	Yes	Task category ID
TASK_SUBMITTER_ID	String/Map	Logged user	No	Task submitter ID/Map
TASK_SUBMIT_DATE	Calendar	Current date	No	Task submit date
TASK_UPDATE_DATE	Calendar	Current date	No	Task update date
TASK_CLOSE_DATE	Calendar	null	No	Task close date
TASK_RESOLUTION_ID	String	null	No	Task resolution ID
TASK_STATUS_ID	String	Start state	No	Task state ID
TASK_UDF_MAP	Map	null	No	Custom field ID/value pairs
TASK_PARENT_TASK	String	Current task	No	Parent task number
TASK_PARENT_TASK_ID	String	Current task	No	Parent task ID

To create a user, put the following data into the resulting **Map**:

Key	Value Type	Default	Required	Description
OBJECT_TYPE	String	Must be **CSVImport.USER_TYPE**	Yes	Object type ID

USER_LOGIN	String	"Not specified" + postfix	No	User login
USER_PASSWORD	String	None	No	User password. No empty password allowed.
USER_NAME	String	"Not specified"	No	User name
USER_PHONE	String	null	No	User phone
USER_EMAIL	String	null	No	User e-mail
USER_PRSTATUS_ID	String	Must be defined	Yes	User status ID
USER_TIMEZONE	String	Logged user time zone	No	User time zone code
USER_LOCALE	String	Logged user locale	No	User locale code
USER_COMPANY	String	null	No	User company
USER_EMAIL_TYPE_ID	String	1 - **HTML** template ID	No	User default e-mail template ID
USER_DEFAULT_PROJECT_ID	String	null	No	User default project ID
USER_EXPIRE_DATE	Calendar	null	No	User expire date
USER_IS_ACTIVE	Boolean	true	No	User is active
USER_SHOW_HELP	Boolean	true	No	Show inline help
USER_HTML_EDITOR	Boolean	true	No	Always use the HTML editor to edit task/message description
USER_SHOW_TREE_MODE	Integer	2- **All tasks and users**	No	Navigation tree show mode: 0 - **None** 1 - **Projects and managers** 2 - **All tasks and users**
USER_UDF_MAP	Map	null	No	Custom field ID/value pairs
USER_PARENT_USER	String	Logged user	No	Parent user login
USER_PARENT_USER_ID	String	Logged user	No	Parent user ID

To create a message, put the following data into the resulting **Map**:

Key	Value Type	Default	Required	Description
OBJECT_TYPE	String	Must be **CSVImport.MESSAGE_TYPE**	Yes	Object type ID
MESSAGE_TASK_ID	String	Must be defined	Yes	Task ID
MESSAGE_MESSAGE_TYPE_ID	String	Must be defined	Yes	Message type ID
MESSAGE_DESCRIPTION	String	""	No	Message description
MESSAGE_HOURS	Double	null	No	Message actual time
MESSAGE_HANDLER_USER_ID	String/Map	null	No	Message handler (user) ID/Map
MESSAGE_HANDLER_GROUP_ID	String	null	No	Message handler (group) ID/Map
MESSAGE_RESOLUTION_ID	String	null	No	Message resolution ID
MESSAGE_SUBMIT_DATE	Calendar	Current date	No	Message submit date
MESSAGE_SUBMITTER_ID	String/Map	Logged user	No	Message submitter

MESSAGE_UDF_MAP	Map	null	No	Custom field ID/value pairs

To search for the object ID by its name, use the following methods of the **CSVImport** class:

Method Name	Arguments	Returns
findCategoryIdByName	Category name	Category ID
findPriorityIdByName	Priority name, Category name	Priority ID
findResolutionIdByName	Resolution name, Message type name	Resolution ID
findTaskResolutionIdByName	Resolution name, Category name	Resolution ID
findStateIdByName	State name, Category name	State ID
findTaskIdByName	Task name	Task ID
findTaskIdByNumber	Task number	Task ID
findUserIdByLogin	User login	User ID
findUserIdByName	User name	User ID
findMessageTypeIdByName	Message type name, Category name	Message type ID
findUserStatusIdByName	User status name	User status ID
findUDFIdByName	Custom field caption	Custom field ID
findUDFListIdByValue	Value of the custom field list	Value ID

4 Developer's Guide

4.1 Localizing User Interface

The following topic 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 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 (**Current User -> 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_MENU_ACL=Access Control...
MSG_MENU_CATEGORIES=Categories...
MSG_MENU_COPY_RECURSIVELY_TASK=Recursively Copy Task {0}
MSG_MENU_COPY_TASK=Copy Task {0}
```

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 letters 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 does not allow for many international characters. It is 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 is 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 do not 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 a 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.

4.2 Extending TrackStudio Functionality

The following 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 the 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).
- E-mail integration.

4.2.1 Implementing Adapters

The following topic describes adapter development.

Description

An adapter is a class which implements the `com.trackstudio.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, which implements the `com.trackstudio.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:

```
adapter.email com.trackstudio.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.store com.trackstudio.app.adapter.store.BaseStoreAdapter;
              com.trackstudio.app.adapter.store.MailImportMessageStoreAdapter;
              com.trackstudio.app.adapter.store.MailImportTaskStoreAdapter;
              com.trackstudio.app.adapter.store.CleanStoreAdapter;
              com.trackstudio.app.adapter.store.PostProcessingStoreAdapter
```

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

TrackStudio assumes each type of adapter as consists of three components: the adapter interface, the proper adapter, and AdapterManager establishing the pipeline.

Let's look the implementation 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 **result** of a returning type, which must come last in the list of parameters. For example,

```
boolean authorizeImpl(String userId,
                      String password,
                      boolean result)
              throws GranException;
```

- Persistent objects (with rare exception) are passed either by their string identifier or in the collections `java.util.Collection`, `java.util.LinkedList`, etc. In the case of an ID passing of the object, to continue working 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 that you use transactions in this case.

Interface example:

```
package com.trackstudio.app.adapter;

import com.trackstudio.exception.GranException;

public interface AuthAdapter extends Adapter {
    boolean authorizeImpl(String userId, String password, boolean result)
              throws GranException;
    void changePasswordImpl(String userId, String password)
              throws GranException;
}
```

2. An adapter has the following structure:

```
package com.trackstudio.app.adapter.auth;

public class SimpleAuthAdapter implements AuthAdapter {

    public boolean init() {
        return true;
    }

    public String getDescription() {
        return "Basic Database Authentication Adapter";
    }

    public boolean authorizeImpl(String userId, String password, boolean result)
         throws GranException {
        ...
    }

    public void changePasswordImpl(String userId, String password)
         throws GranException {
        ...
    }
}
```

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. `XXXAdapterManager` controls the lists of adapters supporting the defined interface and is responsible for the correct passing of the parameters.

The system enhancement is carried out through classes implementing the existing interfaces (for example, `com.trackstudio.app.adapter.AuthAdapter`). 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 registration of new adapters in the system. To call a method (e.g. for exporting), you must execute the following:

```
AdapterManager.getInstance().getExportAdapterManager()
```

```
                                .export(taskid, userid);
```

4.3 Building TrackStudio from Source

The following 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 3.2.x

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

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

To build the Standalone distribution package for UNIX, execute the following command:

```
> ant "-Dinstall4j.home=C:\Program Files\install4j" sa sa-unix
```

To build the WAR distribution package, execute the following command:

```
> ant wardist
```

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

See Also

- Install4j

4.4 Integrating with Third-Party Systems

The following topic 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 security.

The following services are available using SOAP API:

- JavaAttachment (Java-only)
- Acl
- Attachment
- Category
- EmailType
- Export

- Filter
- Find
- Index
- MailImport
- Message
- Prstatus
- Registration
- Report
- SCM
- 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 a proxy to convert parameter types. For example, Java collections are not supported in .NET so a Java collection is converted to 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 follows:

```
    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;
    }
```

4.4.1 Using Java SOAP

The following topic 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
```

4.4.2 Using SOAP from Web Browser

The following topic describes how to use TrackStudio SOAP API from an Web 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
```

When you do this, 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 by ID

- 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>
```

4.4.3 Using .NET SOAP

The following topic 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 must:

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

A

Adding a Bug 61
Adding a Custom Field 94
Adding a Message by E-mail 131
Adding a Task by E-mail 130
Adding Comments to a Task 64
Adding your Web Site to the TrackStudio Server 40
Allowing Group Handler Assignment 100
Analyzing Tasks Distribution 119
Assigned Status Properties 137
Assigning a Task 62
Attaching a File to a Task 66

B

Backing Up and Restoring the Database 24
Building TrackStudio from Source 151
Bulk Updating Tasks 74

C

Calculating a Custom Field Value 104
Category Concepts 52
Category Properties 142
Changing a Password 88
Changing the Task State 65
Changing the TrackStudio URL 40
Choosing Localized Interface 101
Closing a Bug 77
Concepts 48
Configuring E-mail Notification 30
Configuring E-mail Submission 31
Configuring MS Active Directory Authentication 11
Configuring NTLM Authentication 13
Configuring TrackStudio Cluster 26
Configuring X11 Server 15
Creating a Category 124
Creating a Customer Account 82
Creating a Database for Performance Testing 23
Creating a Project 59
Creating a Script 103
Creating a User Account 81
Creating a Workflow 120
Creating and Managing the Database 18
CSV Import Concepts 58
CSV Import Script Reference 145
Custom Field Concepts 54
Custom Field Properties 136
Customers Creating Accounts for Themselves 85
Customizing E-mail Notification Templates 132
Customizing Task and User Processing 91

D

Defining Category Dependency 125
Defining Task Priorities 102
Defining Which Users Can Create, Edit or Delete Tasks 101
Defining Who Can View and Edit Custom Fields 96
Deleting a Task 78
Deleting a User Account 90
Developer's Guide 148

E

Editing Account Properties 83
Editing Task Properties 61
E-mail Import Rule Properties 144
E-mail Notification Concepts 56
E-mail Notification Rule Properties 143
E-mail Submission Concepts 57
E-mail Template Concepts 56
Establishing a User Group Account 78
Extending TrackStudio Functionality 149

F

Filter Concepts 53
Filter Subscription Rule Properties 143
Filtering Subtasks by Properties 116
Filtering Tasks Using AND/OR/NOT 117
Finding a Task 115
Finding a User Account by Login 87
Finding a User Account by Properties 87
Full Text Search Reference 141

G

Generating a Report 72
Granting Users Access to a Project 83
Granting Users Access to Other Users 84

H

Hiding Menu Items, Tabs or Buttons 93
Hiding Unused Fields 91

I

Implementation Guide 42
Implementing Adapters 149
Implementing Triggers 109
Importing and Exporting the Database 24
Importing Data from CSV File 113
Initializing a DB2 Database 18
Initializing a Firebird Database 22
Initializing a MySQL Database 22
Initializing a PostgreSQL Database 19
Initializing an HSQLDB Database 19
Initializing an MS SQL Server Database 20
Initializing an ORACLE Database 20
Installation Guide 3
Installing an SSL Certificate 16
Installing Eclipse Plug-in 28
Installing IDEA Plug-in 27
Installing JBuilder Plug-in 27
Installing TrackStudio SA for UNIX 14
Installing TrackStudio SA for Windows 7
Installing TrackStudio WAR for UNIX 15
Installing TrackStudio WAR for Windows 7
Integrating E-mail Client and TrackStudio 30
Integrating IDE and TrackStudio 27
Integrating IIS and TrackStudio 9
Integrating SCM and TrackStudio 28
Integrating Serence KlipFolio and TrackStudio 34
Integrating with Third-Party Systems 151

L

Linking Tasks 69
Linking Users to a Task 71
Localizing User Interface 148
Locking a User Account 89

M

Managing Categories and Workflows 120
Managing E-mail Notification and Submission 126
Managing Projects and Bugs 59
Managing User Accounts 78
Message Concepts 52
Message Properties 136
Moving a User Account 89
Moving Tasks 67

N

New Features 1

P

Periodically Receiving Lists of Tasks by E-mail 128

R

Receiving E-mail Notification when Tasks Change 126
Reference 134
Report Concepts 53
Report Properties 140
Restricting Handler List 98
Restricting Who Can Close Task 99
Running as Windows Service 8

S

Sample Project Overview 43
Script Concepts 54
Searching and Analyzing Tasks 115
Self-registration Concepts 56
Self-registration Rule Properties 144
Sending Alert for Overdue Tasks 129
Server-specific Notes 17
Setting Default Handler for a Project 99
Setting Workflow for Task Category 126
Sorting Tasks 118
Sorting Tasks by Product 118

Step-by-Step Guides 59

T

Task Concepts 50
Task Filter Properties 138
Task Properties 134
TrackStudio Configuration Properties 35
TrackStudio Updates and Upgrades 3

U

UNIX Installation 14
Upgrading from Version 3.0.x 6
Upgrading from Version 3.1.x 5
Upgrading from Version 3.2.x 4
Upgrading from Version 3.5.x 3
User Concepts 51
User Filter Properties 139
User Interface Concepts 49
User Properties 135
User Status Concepts 53
User's Guide 42
Using .NET SOAP 155
Using Java SOAP 152
Using JES for E-mail Integration 33
Using SOAP from Web Browser 153

W

Windows Installation 7
Workflow Concepts 51