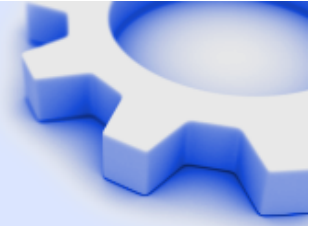


projectengine



Project Engine Server User Manual

Table of Contents

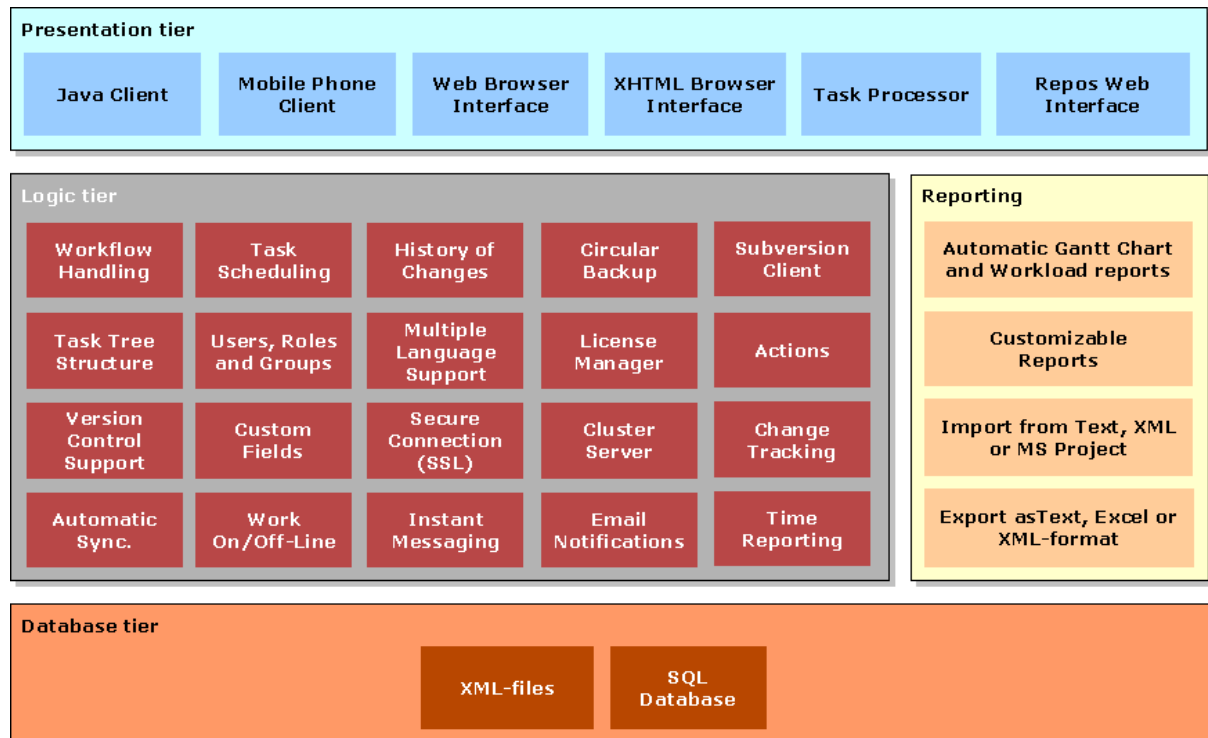
Introduction.....	4
Stateless Server.....	4
Reliability.....	5
Built for Performance.....	5
Web Services.....	5
Data Storage.....	6
Log Files.....	6
Task Summary.....	6
Configuring Reports.....	7
Forms.....	7
Inserting a Task.....	8
Display a Report.....	10
Server Properties.....	12
General Properties.....	12
Security Properties.....	13
Email Properties.....	14
Advanced Properties.....	14
Other Properties.....	16
Email Notifications.....	17
Setting up Email Notifications.....	17
Mail Server Port.....	17
Email Attachments.....	17
Email Template.....	18
Links from Email Notifications.....	18
Receiving Emails.....	19
Security.....	20
Running the Server.....	21
Software Requirements.....	21
Environment Variable.....	21
Running the Server.....	21
Starting the Web Interface.....	21
Stopping the Server.....	21
Server Admin Console.....	22
Starting the Server Admin Console.....	22
Server Properties.....	22
Stop Server.....	22
Send Test Email.....	22
Users Online.....	22
Backup.....	23
Automatic Backup.....	23
Action Log.....	23
Restoring Data from a Backup.....	24
Deploying in an Application Server.....	25
Installing the Application.....	25
Environment Variable.....	25
Launching the Web Interface.....	25
Licensing.....	26
Trial Licenses.....	26
Fixed Licenses.....	26
Floating Licenses.....	26
Upgrading the Server.....	27
Clustering Servers.....	29
Remote Server Properties.....	29

Task Processor.....	31
Assigning a Task to the Task Processor.....	31
Tags in Commands.....	31
Supported Commands.....	32
command.....	32
call.....	33
sql.....	33
get.....	34
post.....	34
load.....	35
save.....	35
wait.....	36
remove.....	36
Setting up an External Task Processor.....	37
Enabling the Internal Task Processor.....	38
Disable the Task Processor.....	38
Task Processor Error Messages.....	38
Database Connection.....	39
Database Synchronization.....	39
Database Settings.....	39
Setting up the Database Connection.....	40
Upgrading the Database.....	41
Technical Details.....	41
Error Messages.....	42

Introduction

The Project Engine Server is the heart of the system. All the intelligence and processing of data is performed here. The server works like a web-/application server, where a HTTP command, with some parameters, gives a response in XML/HTML back to the client.

The server connects any number of clients and servers together to make online collaboration possible. Access the server from the web interface, mobile client, the XHTML interface or any other client that conforms to the specifications of the Project Engine Services API.

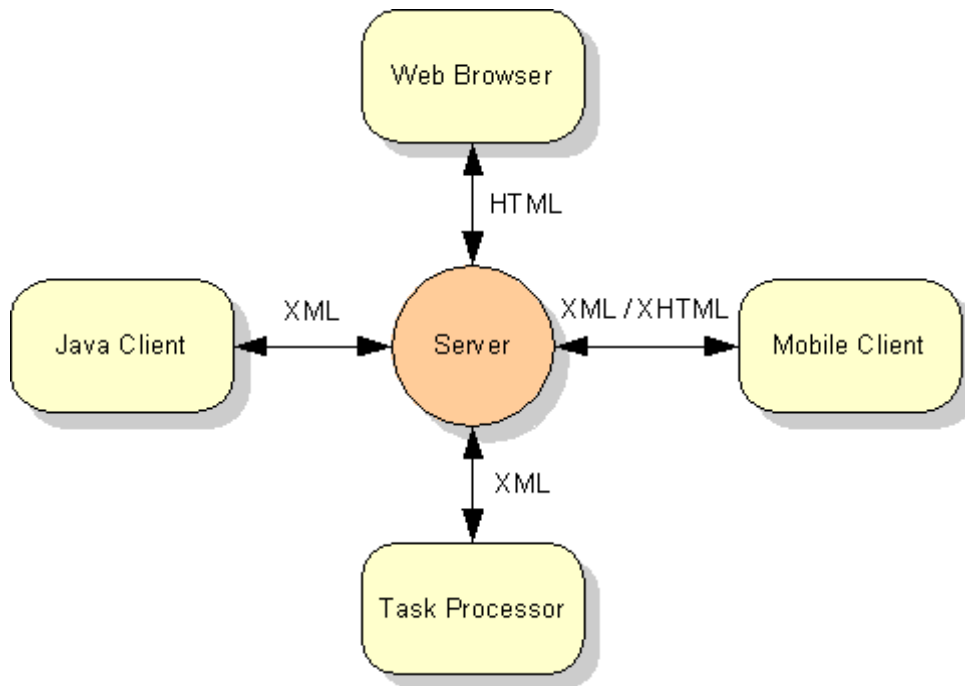


Feature overview of the three-tier system.

Stateless Server

Project Engine Server does not store any information about the status of its clients, it just processes one request after another. The server does not ever send anything to the clients except as a response to a request. The fact that the server is stateless, except from the database, also has the benefit that it becomes extremely reliable. Reliability is the most important feature since the server is designed for running around the clock without interruption.

The picture below shows a number of clients connected to the server and the protocol used for transferring messages.



Example of a number of clients connected to a server.

Reliability

The most important feature of the Project Engine Server is reliability. It is designed for running around the clock without failures or maintenance. This is important since the clients have to rely on the server for performing virtually any task.

To give the administrator the ability to monitor every single request, Project Engine Server produces a log file that receives everything the server does. The administrator of the system might sometimes need to view this file if something unexpected has occurred.

Built for Performance

The server caches all data in memory for optimal performance and uses the file system or a relational database as a secondary storage. Data is written to the file system or a database before the server is shut down. Project Engine Server also periodically stores its cached data, to prevent accidental loss of data due to power failure.

Web Services

All Web Services in the Project Engine Server container are accessed through HTTP (GET or POST). Depending on the type of service the response will be XML or HTTP. A Web Service Tester is supplied with the Project Engine Server and can be used for testing all Web Services in the server container. Use for example the call "hello" to determine if the server is available. All services that return XML also have two optional stylesheet parameters to be able to modify the XML response into HTML.

A couple of example forms are provided in the "form templates" directory to provide guidance on how to call Project Engine Server directly from an HTML page. A list of services and their parameters can be found in the "Web Services.htm" document. This

document covers the basics of how to communicate with the server API from a client. All Project Engine clients use this interface.

Data Storage

Data is, by default, stored in XML format in the "server/xml" directory of the server. The server reads the data upon start-up and writes the data back in the XML files before shutting down.

It is possible to manually edit the data in the XML files but make sure the server is not running while this is done (otherwise changes will be overwritten). All elements in the XML files have an id tag. It is important to know that each element in one level must have a unique tag. Please be careful when editing the XML files by hand since you otherwise might get unsatisfying results.

The "priorities.xml" file should be altered before the start of a new project. It is possible to change the settings during a project but it is not recommended to remove any elements since some tasks might already be using them. Adding a new element as the last element should be safe though.

The following XML files are stored in the XML directory:

File	Explanation
task_tree.xml	This file contains all the tasks in a tree structure.
users.xml	Contains all users, workers and groups in a tree structure.
priorities.xml	Contains any number of task priorities. Can not be changed directly from the client.
history_list.xml	The history of changes. You might like to view this file to retrieve a task that was removed by mistake, but do not edit this file. This XML file can be completely removed at any time. Project Engine will automatically add a new empty file if it is missing.
reports.xml	Contains all custom reports.
messages.xml	Contains all messages. This XML file will contain at most 10000 messages. The server will automatically remove the oldest messages.

Log Files

All requests and replies to/from the server are logged to a file if the logging property is set to "on" in the "server.properties" file. The log file will be named "server/logs/flowlogX.txt", where X is the weekday number (1-7). One log file is saved per weekday overwriting the one-week-old file.

Turn the logging on if you suspect that something is not working correctly or if you like to study how the team works with the system. You need to restart the server after changing the logging property. The default setting is not to write to the log file.

Note that logging takes a lot of performance from the server, so use it only when necessary.

Task Summary

The template for the **Task Summary** view can be found in the "server/stylesheets" directory of the server and is called "task_summary_template.htm". This file may contain plain text or HTML. Insert values from the task by adding tags enclosed in braces, for example: {goal_descr}. The special tag {children} will be replaced with a bulleted list of child tasks.

Configuring Reports

Most reports are produced by applying a stylesheet to the XML response. Project Engine is installed with one default stylesheet per report type. The stylesheet for each report can be customized in many ways like:

- Fields to display
- Sort order
- Layout
- Color
- Font
- Formatting of dates

Use the provided XML editor or a text editor of your choice to edit the stylesheet files.

There are a number of predefined javascript functions available to format dates and text provided with each stylesheet. Use these functions to convert the raw data returned from Project Engine into local dates and character sets.

The "autogenerated.xml" file is generated by the system based on settings from the Project Engine Client and the report that initiated the operation. The "autogenerated.xml" stylesheets are used for all custom reports. The "autogenerated_template.xml" can be customized to modify the appearance of the custom reports.

Forms

Project Engine is closely integrated with the web and uses HTTP POST or GET requests to access the services provided by the server. The response will return either XML or HTML back to the client. Provided with Project Engine are a few examples of how to use standard web forms to be able to extract data from or insert data into Project Engine. The forms can be found in the "server/forms" directory.

Project Engine Web Start is a sample portal that contains some useful forms. The portal can be launched from the Start menu or from a web browser using the address:

[http\(s\)://address:port/forms/start.htm](http(s)://address:port/forms/start.htm)

Note that you need to load the "Issue Tracking" module to be able to use the forms in this example.

Inserting a Task

The screenshot shows a web browser window with the following elements:

- Browser title: Create Issue - Windows Internet Explorer
- Address bar: http://localhost:1234/fo
- Page title: Create Issue
- Page content:
 - Section: Create Issue
 - Text: Add an issue into Project Engine. Requires the Issue Tracking module.
 - Form fields:
 - Issue Type: Change Request (dropdown)
 - Title: (text input)
 - Description: (text area)
 - Buttons: Insert, Clear Information
- Browser status bar: Klar, Internet, 100%

The HTML code below can be used to create a form for adding a new task to the **Task Tree**. The form starts with defining a POST request to the **insertTaskWeb** service. The input fields are passed as parameters to the service.

Note that the name attributes should correspond to a field in Project Engine. The exceptions are: **user_id**, **password**, **stylesheet**, **session**, **error_stylesheet** or **paramX** that are parameters used by all services.

The **insertTaskWeb** service takes one parameter (**param1**) to the parent task to place the new task under. The inserted task will contain the **title** and **goal_descr** fields from the form.

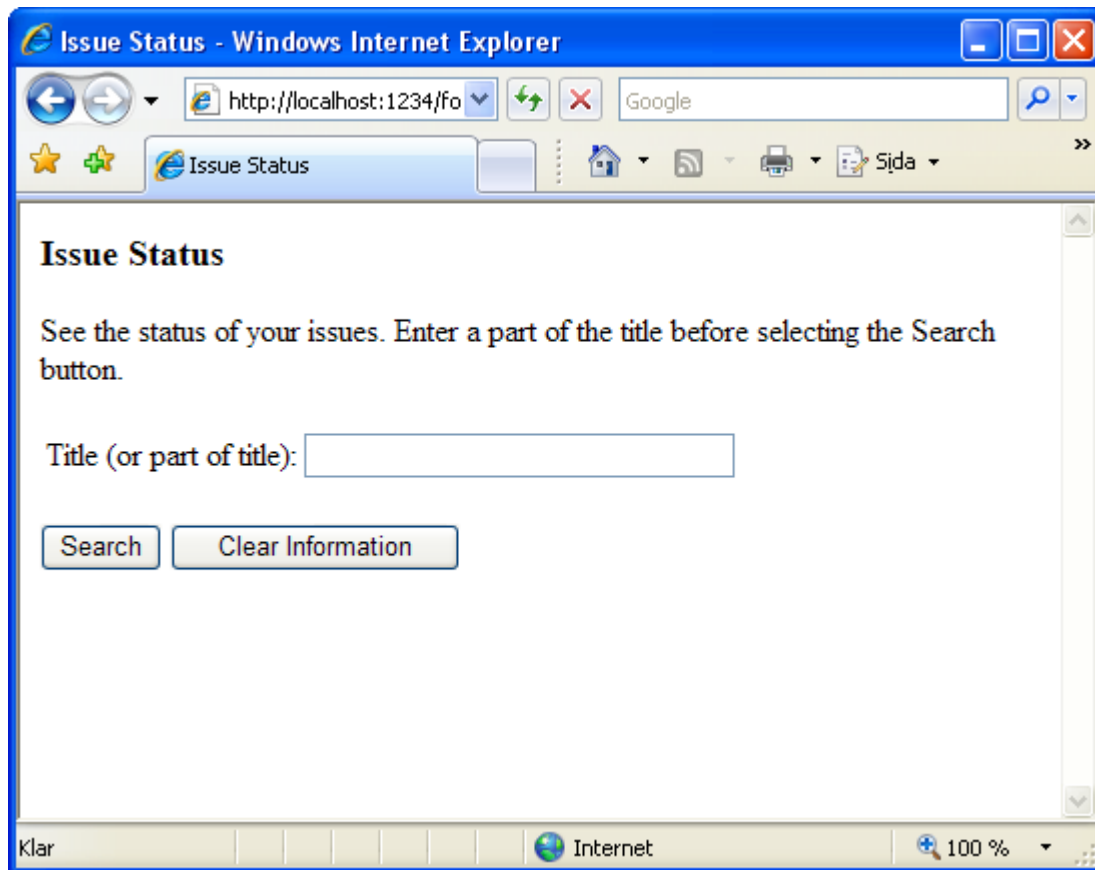
```

<html>
<head>
<title>Create Issue</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<h3>Create Issue</h3>
<p>Add an issue into Project Engine. Requires the Issue Tracking module.</p>
<FORM method="POST" action="/insertTaskWeb">
  <input type="hidden" name="user_id" value="1"/>
  <input type="hidden" name="password" value=""/>
  <input type="hidden" name="assigned_to" value=""/>
  <input type="hidden" name="stylesheet" value="forms/ok.xsl"/>
  <input type="hidden" name="error_stylesheet" value="forms/error.xsl"/>

  <table border="0">
    <tr>
      <td>Issue Type:</td>
      <td>
        <select name="param1" size="1">
          <option value="108921b645b2091">Change Request</option>
          <option value="108921c04c22189">Defect</option>
          <option value="108921bb2ba2150">Risk</option>
        </select>
      </td>
    </tr>
    <tr>
      <td>Title:</td>
      <td><input name="title" size="30" value=""/></td>
    </tr>
    <tr>
      <td>Description:</td>
      <td><textarea rows="6" name="goal_descr" cols="40"></textarea></td>
    </tr>
  </table>
  <br>
  <input type="submit" value="Insert"/>
  <input TYPE="reset" value="Clear Information"/></p>
</FORM>
</body>
</html>

```

Display a Report



The HTML code below can be used for creating a custom report from a form. This example uses a JavaScript for submitting the form (both variants may be used). The **getTaskListWeb** service takes five parameters (**param1** to **param5**):

param1 - The task id to base the report on (Report Root Task).
param2 - Sort field (blank to disable sort).
param3 - Sort order (not user when sort field is blank)
param4 - Visible fields (semicolon separated).
param5 - Report title.

The **title** field is used in the example below to filter tasks in the report. Note that the * character is inserted before title to give the following filter:

```
title=*title
```

This filter will return all tasks that contain the entered title in the **title** field.

```

<html>
<head>
<title>Issue Status</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<script language="JavaScript">
    function executeEvent()
    {
        document.forms[0].elements["title"].value="*" + document.forms[0].elements["title"].value;
        document.forms[0].method="POST";
        document.forms[0].action="/getTaskListWeb";
        document.forms[0].submit();
    }
</script>
</head>
<body>
<h3>Issue Status</h3>
<p>See the status of your issues. Enter a part of the title before selecting the Search
button.</p>
<FORM>
    <input type="hidden" name="user_id" value="1"/>
    <input type="hidden" name="password" value=""/>
    <input type="hidden" name="param1" value="108921af91e2059"/>
    <input type="hidden" name="param2" value=""/>
    <input type="hidden" name="param3" value="0"/>
    <input type="hidden" name="param4" value="title;status;goal_descr;result_descr"/>
    <input type="hidden" name="param5" value="Issue Status"/>
    <input type="hidden" name="stylesheet" value="stylesheets/autogenerated.xsl"/>
    <input type="hidden" name="error_stylesheet" value="forms/error.xsl"/>

    <table border="0">
        <tr>
            <td>Title (or part of title):</td>
            <td><input Name="title" size="30" value=""></td>
        </tr>
    </table>
    <br>
    <input type="button" value="Search" onclick="javascript:executeEvent()" />
    <input TYPE="reset" value="Clear Information"/></p>
</FORM>
</body>
</html>

```

Server Properties

The server settings can be changed by editing the "server/settings/server.properties" file. Use a normal text editor, like Notepad, to edit the file. Make sure the server is not running when editing this file.

The "server.properties" file might look something like this after installation:

```
port_number=1234
logging=off
max_nr_history_tasks=500
save_interval=600
security_mode=0
license_key=e0f59380e0fd9283e0f59380e4f49182
keystore=testkeys
keystore_password=passphrase
create_backup=yes
language=English
mail_server_address=
send_email_interval=180
remote_connection=off
visible_gui=yes
```

Server properties can also be modified from the Server Admin Console. The server will use the new settings after being restarted.

General Properties

Property	Description
port_number	The port number used by the server. It is important that all clients use the same port. If you have more than one Project Engine Server on a single machine they must use different port numbers. The default port number is 1234.
logging	Set property to "on" if you want the server to write all requests to the "flowlogX.txt" file, otherwise to "off". The logging is set to "off" by default. on = Logging is enabled off = Logging is disabled (except for critical errors)
max_nr_history_tasks	The maximum number of history tasks that Project Engine Server will keep in memory.
save_interval	The time in seconds between a complete backup of all tasks to the XML files. Do not set this value too low since it degrades performance, neither too high since you otherwise might lose a lot of data in case of a power failure.
visible_gui	Enables or disables the user interface of the server. "Yes" is the default. The server user interface will in any case not be displayed when installed as a service. yes = Show the server user interface. no = Hide the server user interface.
license_key	The license key received from registering the product. The license key specifies the expiration date of the product and the maximum number of allowed users. The server installs with a time unlimited trial license key for

	<p>one user. Replace this trial key if you have received another key from purchasing the product or requesting a trial license.</p>
create_backup	<p>Project Engine Server can be configured to automatically backup XML data in a one-week cycle. The XML files will be stored in the backup/1 to backup/7 directory depending on the weekday.</p> <p>yes = Create a backup each save interval. no = Do not create automatic backups.</p> <p>Note that this property will have no affect when using a database instead of XML files as a storage (using the save_database property). There is no automatic backup when using a database.</p>
language	<p>Language used by the web interface. "English", "US-English" and "Dutch" are currently the only languages supported by the web interface.</p> <p>Create a custom language by creating a new language property file in the server_languages directory.</p> <p>The language property is the filename without the ".properties" extension. Remember that the language name might be case sensitive on some systems.</p>
validate_subtypes	<p>Validates subtasks created, copied or moved as a subtask to a task. An error message will be given if an invalid task type is detekted. Valid subtypes can be found in the Template tab of the template that defines the task type.</p> <p>yes = Enable validation of subtypes. no = Disable validation of subtypes (default).</p>
convert_role_to_user	<p>Set to yes to enable a feature that converts a role to a user when a user updates a task assigned to a role that he/she belongs to.</p> <p>yes = Enable role to user conversion. no = Disable role to user conversion (default).</p>
login_with_name	<p>Set to yes to force the user to enter a user name instead of selecting one from the drop-down list when logging into the system.</p> <p>yes = Login by entering the name of the user. no = Login by selecting a name from a drop-down (default).</p>

Security Properties

Property	Description
security_mode	<p>Specifies the security of the communication between the client and the server.</p> <p>Note that all clients must have the same setup as the server to be able to communicate.</p> <p>0 = No security, communication between client and server is in readable text. 1 = Use Secure Sockets Layer (SSL).</p>

Property	Description
keystore	The filename of the keystore that contains the key for SSL. Note! Only to be used for security mode 1.
keystore_password	The password for the keystore above. Note! Only to be used for security mode 1.

Email Properties

Property	Description
mail_server_address	The address to an outgoing (SMTP) mail server or blank if none is available. Set this property to blank to disable email notifications.
mail_server_user_id	User id if the outgoing mail server requires SMTP authentication. Blank if authentication is not required.
mail_server_password	Password to the outgoing mail server authentication.
mail_server_email	Email address to use when sending a mail from the mail server.
send_email_interval	Specifies the interval Project Engine Server checks for any recently delivered tasks to send an email notification for. Default is 180 seconds.
max_size_attachment	Maximum size of an attachment (in bytes) that will be included in an email notification.
incoming_mail_server_address	The address to an incoming (POP3) mail server or blank if none is available. Set this property to blank to disable import of emails to tasks.
incoming_server_user_id	User id to the incoming (POP3) mail server.
incoming_server_password	Password to the incoming (POP3) mail server.
import_root_task_id	The id of the task that the task that was created from an email is placed under.

Test the current email settings by pressing the Send Test Email button. Enter an email address to send to and press the **Ok** button. You will get an error message if an email could not be sent with the current settings.

Advanced Properties

Property	Description
remote_connection	Enables or disables the connection to a remote server, see Clustering Servers. Only set this property to "on" if you intend to connect several servers to form a cluster. on = Server is connected to a remote server (see clustering) off = Not connected to a remote server (default)
load_database	Load data from database or XML-files. yes = Load data from database. no = Load data from XML-files (default).
save_database	Save data to database or XML-files. yes = Save data to database. no = Save data to XML-files (default).
direct_save	Saves data immediately after an updating command.

	<p>Note that direct save requires a lot of server performance. Use only in combination with save_database=yes since the database synchronization will only save changes not the entire data structure.</p> <p>yes = Save data immediately after an update. Data will also be save periodically. no = Wait until periodic save specified by the save_interval property (default).</p>
repository_url	<p>An URL to a web based file repository or blank if none available.</p> <p>This property is needed by the web interface to translate relative (file) links into a full URL to files in a web enabled version control system such as Subversion. File links (relative to the link root path in the Java client) will not be available in the web interface if this URL is blank.</p>
task_processor_user	<p>The Task Processor User that processes tasks automatically. Set this field to blank to disable the internal Task Processor.</p>

Other Properties

The properties in the table below are not visible from the **Server Properties** dialog and can only be modified from the "server.properties" file.

attachment_directory	Path to the attachments directory (default "attachments").
xml_directory	Path to the xml directory (default "xml").
backup_directory	Path to the backup directory (default "backup").
deleted_directory	Path to the deleted directory (default "deleted"). The deleted directory contains XML-files of deleted tasks. Use the Import from XML-file item in the Task Tree Menu to restore the tasks.
license_interval	Time in seconds between validating the license key against the currently connected users and updating the no users online. This value can range from 60 to 600 seconds.
task_processor_interval	The time in seconds between fetching the Your Task List for the Task Processor User. The default interval is 60 seconds.

Email Notifications

Project Engine Server can be set up to send an email notification when a task has been delivered to a user or to notify the user that assigned a task about changes.

The user must have an email specified in the user properties dialog and have email notifications enabled. An email will only be sent when a user assigns a task to another user, or when a change is made by anyone else than the person who assigned the task.

Most of the email notification settings can be reached from the **Server Properties** dialog but the **mail_server_port** property can only be configured in the "server.properties" file.

Setting up Email Notifications

Project Engine does not include a mail server. If you do not have a mail server installed we recommend Mail Enable (<http://www.mailenable.com>) that is free of charge and easy to set up.

The mail server (SMTP) that will be used must be specified in the **mail_server_address** property, and the email address to use as the sender should be specified in the **mail_server_email** property. Use a blank mail server address to disable mail notifications. The **send_email_interval** property specifies the interval, in seconds, that Project Engine Server checks for any recently delivered tasks.

Mail Server Port

The default port for SMTP is 25. Set the port using the **mail_server_port** property if your email server uses another port.

Email Attachments

Attachments to a task will be included in the email notification if the user has enabled attachments in the manage user dialog and the attachment does not exceed the size specified in the **max_size_attachment** property.

Email Template

The "email_template.htm" file in the "server/stylesheets" directory contains the template for email notifications and looks like this by default:

```
<html>
<body>
Assigned by: {assigned_by_name}<br>
Assigned to: {assigned_to_name}<br>
Task status: {status}<br>
Total work remaining: {total_work_left}<br>
<br>
{goal_descr}
<hr>
<br><a href='http://127.0.0.1:{port_number}/serviceEditTaskForm?
session={session}&param1={id}'>Open Task</a>
|<a href='http://127.0.0.1:{port_number}/startTask?
session={session}&param1={id}'>Start Task</a>
|<a href='http://127.0.0.1:{port_number}/completeTask?
session={session}&param1={id}'>Complete Task</a>
<br><br>Do not reply to this email, it has been auto generated by Project
Engine.
</body>
</html>
```

Email notifications can be configured by editing the email template. This file may contain plain text or HTML. A HTML file must start with "<html>". Insert values from the task by adding tags enclosed in braces, for example: {goal_descr}.

Links from Email Notifications

Email notifications may include links back to the Project Engine Server. Links can be used for all operations supported by the server, for example to open a task in the web interface, start a task, or complete a task.

The links back to the server may contain the tags **port_number** and **session**. The **port_number** is the port used by the Project Engine server and **session** is used instead of passing **user_id** and **password** (that is also an option) to the server.

The "email_template.htm" file contains three links by default:

- Open Task
- Start Task
- Complete Task

The server address "127.0.0.1" may only be used on the same machine as the server. Replace this address with the address used by the server. Also include the domain name if you are using a reverse proxy (see Domains).

Receiving Emails

The Project Engine server can be configured to convert received emails into tasks. The title of the email will be the **Title** of the task. The text content of the email will be the **Goal Description** of the task. Any attachments will be added to the task.

The new task will be inserted below the **Import Root Task**. The Import Root Task is by default the root of the **Task Tree**.

The task will be assigned by the user that the email was sent from. The task will be assigned by the same user as the **Import Root Task** if the email address can not be found in any of the user settings (**Email** field in the **User Dialog**).

The new task will be assigned to the user specified by the template that defines the default task below the **Import Root Task**. An imported task is by default inserted below the "My Tasks" task. A "My Task" type is by default assigned to the same user that created the task.

Note that the email is removed after the task has been successfully inserted into the **Task Tree**.

The **Incoming Mail Server** should point to the POP3 server to use for receiving emails. Enter the user id and password in the **Incoming Mail User Id** and **Incoming Mail Password** fields.

The **send_email_interval** property specifies the interval, in seconds, that Project Engine Server checks for any received emails.

Security

Project Engine has two security levels (0-1). Both the client and the server must be set to the same level to be able to communicate. The table below shows the details of the two modes:

Security Mode	Description
0 - No security	This mode sends data unencrypted in a readable text format between the client and server. This option provides the fastest communication and is recommended when all clients are behind a firewall. This is the default setting.
1 - SSL	<p>Secure Sockets Layer. This option provides the safest protection of the data communicated between the client and server but is also the slowest. SSL requires a certificate to be installed on the client and a corresponding key at the server.</p> <p>Project Engine Server installs with a sample keystore - "testkeys" and the client installs with a certificate file - "cacerts". To set up an SSL connection you need to change the security_mode property to "1" (SSL) on both the client and server.</p> <p>Read more about how to create and install a certificate on java.sun.com.</p>

Running the Server

Software Requirements

Since Project Engine is implemented in Java you need to make sure you have a Java Runtime Environment (JRE) installed. We recommend using Java Runtime Environment 1.4 or newer. Download Java from: <http://www.java.com>

Environment Variable

The environment variable "PROJECT_ENGINE_DATA" should point to the "ProjectEngine/server" folder that contains settings and data. This environment variable will be set automatically by the startup scripts.

Running the Server

Run the "startServerWindows.bat" file in Windows or "startServerLinux" file in Linux/Unix to start the server.

Starting the Web Interface

Use a web browser to access the web client. The web interface will be located on port 9090 by default: <http://localhost:9090/ProjectEngineWebClient>

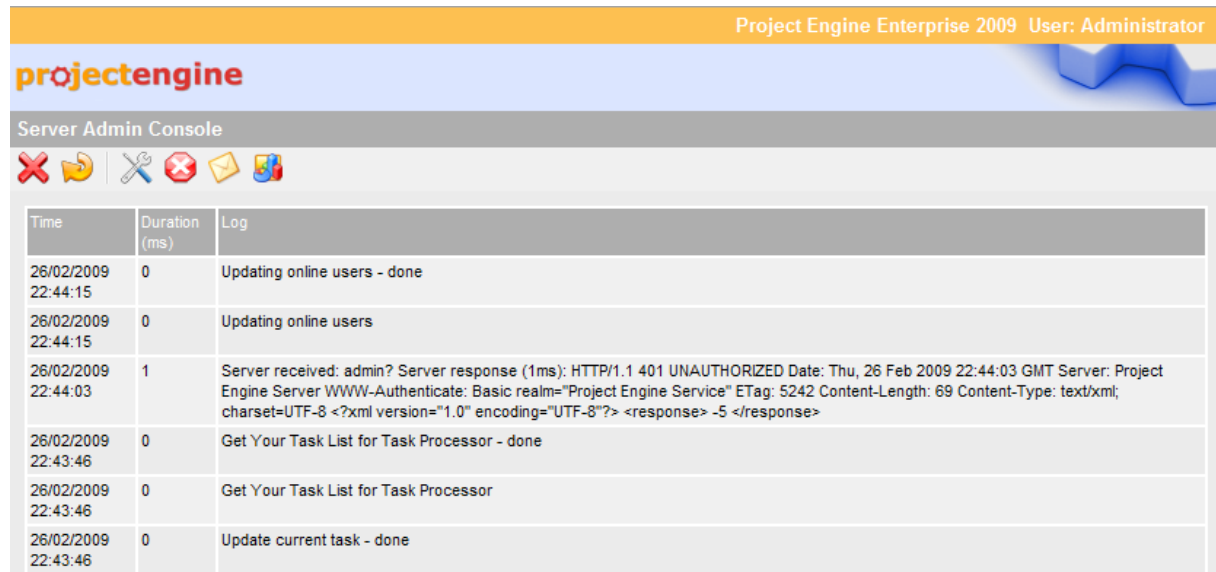
Use the login id "admin" with no password to login the first time.

Stopping the Server

Press the **Stop Server** button in the Server Admin Console to stop the server. The server may also be stopped by terminating the process. All data will be saved to the XML files or the database before the server is stopped.

Server Admin Console

The Server Admin Console can be used for adjusting server settings or monitoring the application.



Time	Duration (ms)	Log
26/02/2009 22:44:15	0	Updating online users - done
26/02/2009 22:44:15	0	Updating online users
26/02/2009 22:44:03	1	Server received: admin? Server response (1ms): HTTP/1.1 401 UNAUTHORIZED Date: Thu, 26 Feb 2009 22:44:03 GMT Server: Project Engine Server WWW-Authenticate: Basic realm="Project Engine Service" ETag: 5242 Content-Length: 69 Content-Type: text/xml; charset=UTF-8 <?xml version="1.0" encoding="UTF-8"?> <response> -5 </response>
26/02/2009 22:43:46	0	Get Your Task List for Task Processor - done
26/02/2009 22:43:46	0	Get Your Task List for Task Processor
26/02/2009 22:43:46	0	Update current task - done

The list shows the latest activities performed by the server. Press the **Update** button to update the list. Press the **Cancel** button to return to the Project Engine Web Interface.

Starting the Server Admin Console

The Server Admin Console can be accessed from the following link:

[http\(s\)://address:port/admin](http(s)://address:port/admin)

Server Properties

Click the **Server Properties** button to open the **Server Properties** page. This page contains the same settings as in the Server Properties chapter.

Stop Server

Press the **Stop Server** button to stop the server. You will need to confirm this on the next page.

Send Test Email

Press the **Test Email** button to test the email settings. Enter an email address on the next page to send a test email to.

Project Engine will use the most recent settings from the **Server Properties** page, not the actual settings, when sending the email. That makes it possible to try several email settings without restarting the server.

Users Online

Press the **Users Online** button to display a list of users that are currently working online.

Backup

It is strongly recommended to create backups of the data periodically. Once a week might be suitable for a small project but larger projects might prefer a daily backup.

The data to backup can be found in the "server/xml" directory. The server does not have to be shut down before performing this action.

Attachments can be found in the "server/attachments" directory. If you are using attachments you should consider to backup the files in this directory also.

Automatic Backup

Project Engine Server can be configured to automatically backup XML data in a one-week cycle.

Set the **Create Backup (create_backup)** property to "yes" (default) to activate the automatic backup cycle. Project Engine will save the XML data to the backup folder under the current day (1-7) at each **Save Interval**.

Note that backups to the same folder will be made many times per day.

Action Log

To make it virtually impossible to loose data the server logs all activities that may alter the state of the XML files or database. All activities are logged to the "logs/actionlog.txt" file. The file contains all actions performed by the server since last successful save to the XML files or database.

If **create_backup** if set to "yes" the server will also log activities to the "actionlog.txt" file in the current backup directory. The "actionlog.txt" file in the current backup directory will be cleared of actions when a new backup has been stored successfully. The file contains the activities that have not yet been backed up to the XML files. The "actionlog.txt" file uses the same format as the "synclog.txt" file and Project Engine modules (.pem files).

Restoring Data from a Backup

The backup data has exactly the same format as the data in the "server/xml" directory. To restore data from a backup, follow the steps below:

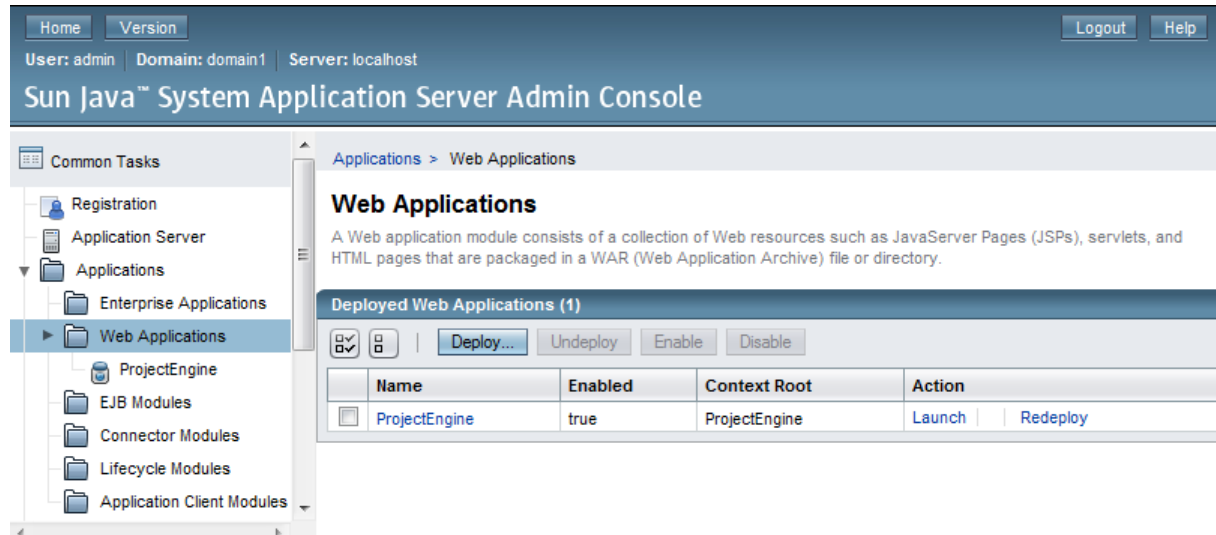
1. Stop the server and wait for the server to shut down.
2. Backup the "server/xml" directory of the server (optional).
3. Replace all XML files in the "xml" directory with the XML files from the backup.

Data can also be restored from one of the "server/backup" directories 1-7. To restore data from a backup, follow the steps below:

1. Stop the server and wait for the server to shut down.
2. Backup the "server/xml" directory of the server (optional).
3. Replace all XML files in the "server/xml" directory with the XML files from one of the backup directories "server/Backup/1" to "server/Backup/7".
4. Start the server.
5. If the "server/logs/actionlog.txt" file is not empty (actions not part of the backup): Rename the "actionlog.txt" file to "actionlog.pem" and use the **Import Module** command from the **File** menu of the client to import the commands from the "actionlog.pem" file. The client should be online when performing the import.

Deploying in an Application Server

Project Engine Server can be deployed as a servlet in any Java application server or servlet container. The Jetty servlet container is used by default. Follow the instructions below to use another container.



Installing the Application

Follow the steps below to install Project Engine in an application server:

1. Install the required files by unzipping the "ProjectEngine2010.zip" file. Note that you should not unzip the files below the "Program" folder in Windows Vista or newer.
2. Start the application server and open the Admin Console.
3. Select to install Web Applications (.war files). Deploy the "ProjectEngineServer.war" and "ProjectEngineWebClient.war" files from the "ProjectEngine/webapps" folder.

Environment Variable

The environment variable "PROJECT_ENGINE_DATA" should point to the "ProjectEngine/server" folder that contains settings and data.

Launching the Web Interface

The Project Engine web interface can be found at the address:

`http://address:port/ContextRoot`

Assuming we are launching from the server machine with "ProjectEngineWebClient" as the context root and an application server using port 8080 we would get the address:

`http://localhost:8080/ProjectEngineWebClient`

Licensing

Project Engine Server supports **trial, fixed and floating** licenses. The license key is a hexadecimal string that contains information about the customer, max number of users and expiration dates. A purchased license key is unique to a customer.

Trial Licenses

A trial license expires on a specific date. A trial license allows up to a certain number of users to work against the server at the same time.

A user license will be released when a user has been inactive for 5 minutes, by default.

Note that a user is considered to be active just from automatic updates from the client.

Fixed Licenses

A fixed license limits the number of users you are able to add to the system. Users are managed in the "Manage Users" dialog of the client ("users.xml" file on the server).

The fixed license do not have an expiration date but contains a **purchase date**. The license key will work with any versions of Project Engine Server but features introduced after the purchase date might be unavailable. You need to purchase an upgrade to take advantage of the new features.

Fixed licenses are supported for backward compatibility only and can not be purchased from the Project Engine website.

Floating Licenses

A floating or trial license allows up to a certain number of users to work against the server at the same time.

A user license will be released when a user that has been inactive for 5 minutes.

Note that a user is considered to be active just from automatic updates from the client.

The floating license do not have an expiration date but contains a **purchase date**. The license key will work with any versions of Project Engine Server but features introduced after the purchase date might be unavailable. You need to purchase an upgrade to take advantage of the new features.

Upgrading the Server

Follow the steps below to upgrade to a newer version of the Project Engine Server:

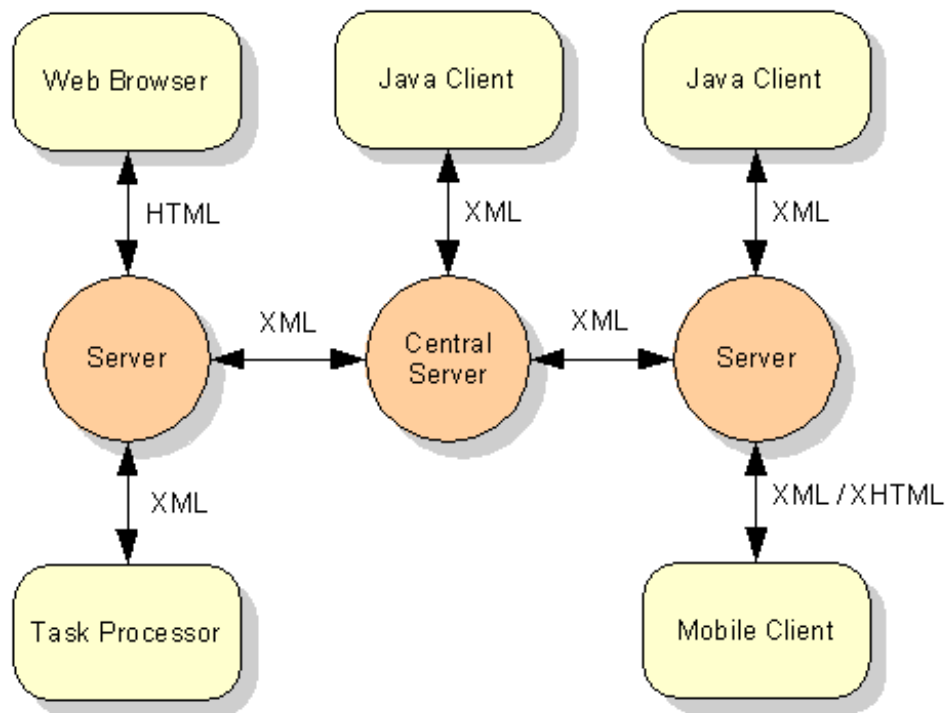
1. Make sure the server is not running.
2. Make a backup of the data from the current installation of Project Engine Server. Copy all files below the "ProjectEngine/server" folder to a new location.
3. Uninstall the old version of Project Engine.
4. Install the required files by unzipping the "ProjectEngine2010.zip" file. Note that you should not unzip the files below the "Program" folder in Windows Vista or newer.
5. When installation is completed copy the old XML files from the "server/xml" directory in the old Project Engine Server installation and replace the XML files in the new installation. Note that you should keep the copy of the old files since they will be automatically converted to the new format.
6. Copy the "server/settings" folder from the old version of Project Engine Server to the new.
7. Copy the content in the "server/attachments" directory from the old version of Project Engine Server to the new.

Performance

A server connected to many clients may run into performance problems resulting in slow responses to client requests. There are numerous ways of optimizing performance; some of them are listed below:

1. Increase the maximum amount of memory used by the JVM on the client and server side. The system will be very slow when running out of memory. Add the `-Xmx512m` option to the shortcut in the Start menu (default in Project Engine 2007) or the Java command that launches the application to specify a 512MB memory pool.
2. Turn off logging in the server properties. Important, drains a lot of performance!
3. Uncheck the **Update after save** option in the **Client Properties** dialog and use the **Update** button to refresh reports.
4. Set a time for **Hide completed tasks after...** in **User Dialog Preferences**.
5. Set **create_backup** to off in server properties and perform backups manually.
6. Make the project team use the web interface when possible since it is less demanding.
7. Increase the save interval of the server (**save_interval** property).
8. Increase the update interval value of each client. The higher the less frequent updates with the server.
9. Reduce the number of history tasks stored (the **max_nr_history_tasks** property). This has very little effect when using a database connection (only saves memory).
10. Close non-used views in the client. Each open view uses calls to the server to remain updated.
11. Use non-encrypted communication if possible.
12. Split the project load on several servers, see Clustering Servers. Local servers may use unencrypted communication with the clients while servers communicate over encrypted connections.
13. Setup the server to use a database to store the data in instead of the XML files. The history list will be less demanding and also unlimited in size. Only changes are written to the database at each save interval not the entire task tree as when using XML files.
14. Archive task periodically. Use a custom report to list old tasks. Export all tasks in the custom report using the **Save as XML** button. Mark all tasks in the report and use the right-click menu to remove all tasks. All tasks may be imported using the **Import from XML-file** option in the **Task Tree** menu.

Clustering Servers



Example of a configuration of two local servers connected to a central server.

Several Project Engine Servers can be connected in a cluster. Each server can access exactly one other server. This makes it possible to work connected to a local server while all tasks are shared between local servers using one central server.

The main benefits of a clustered solution is that the clients will have a fast local connection to the server while still being able to access all tasks.

Information is synchronized between servers at a predefined interval specified by the **update_interval** property.

Remote Server Properties

The "remote_server.properties" file contains the information required for connecting to a remote server. The properties for the remote server can also be changed from the server user interface by selecting the **Remote Server Properties** button.

Note that the **remote_connection** property (Advanced tab in the **Server Settings** dialog) in the "server.properties" file needs to be set to "on" to enable the remote connection.

Property	Description
server_address	The Internet address to where the remote Project Engine Server is located.
port_number	The port number used by the remote server.
security_mode	Specifies the security of the communication between the server and the remote server. 0 = No security, communication is in readable text.

	1 = Use Secure Sockets Layer (SSL).
update_interval	The time in seconds between synchronization of servers. Do not set this value too low since it degrades performance! Should be less frequent than the update rate between servers and clients. Set to 0 to deactivate synchronization.
proxy_server_address	The address of the proxy server. Provide this address when accessing the Internet through a proxy and therefore need to tunnel through using SSL. Remove (or set to blank) this property if not behind a proxy.
proxy_port_number	The port number of the proxy server. Only used when tunneling through a proxy server.
user_id	User used when synchronizing data. Select a user with access to all tasks, for example the Administrator. Click the Update Users button to retrieve a list of users from the remote server. Adjust the settings and click the Update Users button again if the list is empty.
password	Password for the user above. A password is required for retrieving data from the remote server.

Task Processor

The Task Processor processes tasks automatically. The Task Processor simply completes the tasks in its task list (**To-Do List**). The only difference between a task assigned to a user and a task assigned to the Task Processor (also a user in the system) is that the **Goal Description** contains a command instead of a text description.

Any number of Task Processors can be connected to a server, and they can be installed at any location in the world. The Enterprise edition contains a Task Processor User that is enabled by default. The built-in Task Processor User will try to execute it's task list once every minute.

The Task Processor can be used to create advanced workflows that consists of both manual and automatic tasks. The Task Processor can communicate with external systems and databases in a distributed environment.

Note that the build-in Task Processor User in the Enterprise edition does not occupy a license while connecting external Task Processors will use one user license each.

Assigning a Task to the Task Processor

Tasks assigned to the Task Processor User will be processed by the Task Processor when delivered to the Task Processor Users **To-Do List**. The Task Processor will try to execute the command in the **Goal Description** field.

The status of the task will be set to **In Progress** before processing the command. The status will be set to **Completed** after being processed ok and the **Result Description** field will contain the result of the operation.

The **Result Description** will contain an error message if the operation fails and the status of the task will be **In Progress**. Clear the **Result Description** and set the status to **Not Started** or **In Progress** to allow the Task Processor to try again.

Tags in Commands

Commands may contain any number of tags. Inserting tags can be very useful, for example to include the **Result Description** and the **Title** in an automatically generated email. Such a command could look like:

```
call sendEmail bill@company.com {parent.title} "Result: {parent.goal_descr}"
```

Note that tags inclosed in braces ({ }) will be replaced by a value when the task is delivered.

Use the "escape" command if a tag that might contain quotes is enclosed in quotes. For example if parent.goal_descr in the example above might contain quotes.

```
call sendEmail bill@company.com {parent.title} "Result: {{escape parent.goal_descr}}"
```

Supported Commands

The following commands are supported by the Task Processor:

Name	Parameters
command	command_parameter
call	parameter1 parameter1 ...
sql	driver_class connection_string sql_command
get	address
post	address request
load	file_path
save	file_path text
wait	milliseconds/file_path
remove	file_path

All commands are described in detail below.

command

Executes an operating system command on the Task Processor machine. The first and only parameter is the command. Enclose the command string within quotation marks (") to allow spaces between command parameters.

Note that this command can be used for executing other applications to extend the functionality in Project Engine.

Parameters

command_parameter	The operating system command to execute.
-------------------	--

Examples

```
command "explorer http://www.projectengine.nu"  
command "c:\windows\system32\ipconfig.exe"  
command "java -jar C:\ProjectEngineAdmin.jar createTrialLicense pro"
```

Return Value

An error message or the output from the operating system command is stored in the **Result Description** field of the task.

call

Calls any of the services in the Project Engine Server. The first parameter is the name of the service followed by the parameters. See the Web Services documentation for more information on supported services and their parameters.

Parameters

parameter1	The first parameter to the service.
parameter2	The second parameter to the service.
...	...

Example

```
call sendEmail bill@company.com "A Title" "Hello Bill"
```

Return Value

An error message or the result of the call is stored in the **Result Description** field of the task.

sql

Executes an SQL command in a database. For example an Update, Insert, Delete or Select statement. The **sql** command takes three parameters: **driver_class**, **connection_string** and **sql_command**.

The result is stored in the **Result Description** field of the task. If the SQL command contains a select statement the result will be a table of data stored in the **Result Description** field. The data table has the same format as a CSV file (Comma Separated Text File).

Parameters

driver_class	The driver_class is the driver class to use, the driver should be provided in the "database_driver.jar" file.
connection_string	The connection_string is the string that specifies the location and login information for the database.
sql_command	The sql_command parameter contains the SQL command to execute.

Example

```
executeSql "com.mysql.jdbc.Driver" "jdbc:mysql://  
localhost/pe?user=root&password=xxxxxx" "select id from reports"
```

Return Value

An error code or the fetched result from a Select statement is stored in the **Result Description** field of the task.

get

Calls a HTTP server using a GET command. The get command takes only one parameter. The parameter should be the same as the web address in a browser window. The returned HTML will be stored in the **Result Description** field of the task.

Parameters

address	Web address (URL). The parameter should be the same as the web address in a browser window.
---------	---

Example

```
get "http://www.projectengine.se"
```

Return Value

An error message or the returned HTML is stored in the **Result Description** field of the task.

post

Calls a HTTP server using a POST command. The post command takes two parameter. The first parameter is the web address to the service and the second is the request content. The returned HTML will be stored in the **Result Description** field of the task.

Parameters

address	Web address (URL). The parameter should be the same as the web address in a browser window.
request	The POST request content. The content may contain request parameters or other content such as a SOAP request to call a Web Service.

Example

```
get "http://www.projectengine.se"
```

Return Value

An error message or the returned HTML is stored in the **Result Description** field of the task.

load

Loads a text file in UTF-8 format to the **Result Description** field.

Parameters

file_path	The relative (to the "TaskProcessor.jar" file) or full path to a text file to load into the Result Description field.
-----------	--

Example

```
load "c:\my file.txt"
```

Return Value

An error message or the loaded text is stored in the **Result Description** field of the task.

save

Saves a text to a file in UTF-8 format.

Parameters

file_path	The relative (to the "TaskProcessor.jar" file) or full path to a text file to create. The file will be overwritten if it exists.
text	The text to save.

Example

```
save "c:\my file.txt" "hello"
```

Return Value

The result of the call is stored in the **Result Description** field of the task.

wait

Wait a number of milliseconds or for the existence of a file.

Parameters

milliseconds/file_path	<p>The wait command can be used in two different ways:</p> <p>A number: The number of milliseconds to halt the Task Processor. Note that this will prevent the execution of other tasks for the actual Task Processor.</p> <p>A file or directory path: Wait for a file or directory to exist. The task will not be completed until the specified file or directory exists. This can be used before a load command to ensure that the file exists before loading it into the Result Description of a task.</p>
------------------------	--

Example

```
wait 1000
wait "c:\new file.txt"
```

Return Value

The result of the call is stored in the **Result Description** field of the task.

remove

Removes a file or directory. Useful for removing a file that has been loaded.

Parameters

file_path	The relative (to the "TaskProcessor.jar" file) or full path to a text file to remove.
-----------	---

Example

```
remove "c:\my file.txt"
```

Return Value

The result of the call is stored in the **Result Description** field of the task.

Setting up an External Task Processor

The external Task Processor uses the **Remote Server Properties** dialog to set up the connection to the server. This is exactly the same settings as when connecting several servers to form a cluster.

The **User** and **Password** fields needs to be set to an existing user on the server - the Task Processor User. The **Update Interval** is the time in seconds between fetching the **Your Task List** for the Task Processor User from the connected server.

Enabling the Internal Task Processor

Select a Task Processor User from the drop-down menu in the **Advanced** tab of the **Server Properties** dialog to enable the task processor built into the Project Engine Server. The internal Task Processor has a default update interval of 60 seconds.

The Task Processor User in the Enterprise edition should be enabled directly after installation. Simply assign a task with a command to the Task Processor and wait, up to one minute, for the task to complete.

Disable the Task Processor

Select the blank option in the **Task Processor User** drop-down to disable the build-in Task Processor

Task Processor Error Messages

All error messages begins with an error code followed by a description. A task has failed if the **Result Description** field begins with a negative number.

Error	Description
0	Processed ok or no command specified. This is not an error.
-100	Unknown command
-102	No command to execute
-103	No service specified
-104	Failed to connect to database
-105	SQL Exception
-106	Too few parameters
-107	Malformed URL
-108	Failed to remove file or directory
-109	Failed to load file
-110	Failed to save file

Database Connection

Project Engine Enterprise can use a database as a repository for all dynamic data instead of the XML files used by default. Project Engine can use any modern relational SQL database such as MySQL, Microsoft SQL Server or Oracle.

Database Synchronization

Instead of saving data directly all data is cached in memory, just like when operating against XML files, and synchronizes all tasks and units at each save interval.

Note that data is synchronized, not overwritten, any data changed by another application will be loaded to the memory cache. The **updated_date** field is used to determine if the database entry is newer than the cached information. The history list works slightly different and stores all changes in the history table directly.

Database Settings

There are three properties files that needs to be configured for the database connection. The "server.properties" file contains the **load_database** and **save_database** properties that tells the server to load and/or save data to the database instead of the XML-files. The **load_database** and **save_database** properties can be configured from the Server Properties dialog.

Property	Description
load_database	Load data from database or XML-files. yes = Load data from database. no = Load data from XML-files (default).
save_database	Save data to database or XML-files. yes = Save data to database. no = Save data to XML-files (default).

The "database.properties" file contains the connection string and the database driver name. An example for MySQL is provided by default but the user name and password must be changed.

Property	Description
driver_class	The classpath to the JDBC database driver. For example: com.mysql.jdbc.Driver The database driver Java archive (JAR) should be installed in the same directory as the ProjectEngineServer.jar file and have the name: "database_driver.jar". A driver for MySQL is installed with Project Engine by default. Replace this file with a driver for Oracle, SQL Server or another database if needed.
connection_string	The connection string for the database used by Project Engine. The example below is for a MySQL database. For example: jdbc:mysql://localhost/pe?user=root&password=xxx

The "database_mapping.properties" file contains the mapping from the fields used in the XML files to the actual columns used by the corresponding table. Also the table name is included in the mapping. Some example rows of the mapping file is shown below. This file is provided with each installation of Project Engine and can in most cases be used as is.

Property	Description
task.table_name=task_tree	task.table_name defines the name of the table to map against a task, in this case the table "task_tree".
task.parent_id=parent_id	The name of the parent_id column in the task_tree table of the database.
task.id=id	The id field of a task is mapped to the id column in the task_tree table of the database.

Setting up the Database Connection

Perform the following steps in order to setup a database connection:

1. Create a new database named "pe" (or something else).
2. Run the "pe_mysql.sql", "pe_oracle.sql" or "pe_sqlserver.sql" script provided with Project Engine that is located in the "sql" directory of the server installation. The ".sql" script creates all tables and columns used by Project Engine.
3. Update the "database.properties" file and make sure that it contains the correct path to the database, the database name, user id and password. Make sure the driver classpath points to an installed Java database driver. A driver for MySQL is provided with Project Engine (no need to change the classpath).
4. Make sure the server is not running.
5. Set the properties **load_database=no** and **save_database=yes** in the "server.properties" file. This setting will load the data from the XML files to the memory cache and synchronize with the database at each save interval (and shutdown).
6. Start the server. Data will be fetched from the XML files.
7. Stop the server. Data will be stored in the database.
8. Set the properties **load_database=yes** and **save_database=yes** in the "server.properties" file.
9. Start the server. The server will from now on load and save data from/to the database.

Upgrading the Database

The database might need to be updated with new columns and/or tables when a new release of Project Engine is installed. This operation is quite simple since Project Engine server has the ability to temporarily fetch data from a database and storing data to the XML files. The server may also load data from the XML files and store data in the database.

Perform the following steps in order to upgrade a database:

1. Make sure the server is not running.
2. Set the properties **load_database=yes** and **save_database=no** in the "server.properties" file. This setting will store all data from the database in XML files (with the updated format).
3. Start the server.
4. Stop the server. The information from the database should now be stored in the XML files on the server.
5. Set the properties **load_database=no** and **save_database=yes** in the "server.properties" file. This setting will load the data from the XML files to the memory cache and synchronize with the database at each save interval (and shutdown).
6. Start the server. Data will be fetched from the XML files.
7. Stop the server. Data will be stored in the database.
8. Set the properties **load_database=yes** and **save_database=yes** in the "server.properties" file.
9. Start the server. The server will from now on load and save data from/to the database.

Technical Details

All XML files except the "priorities.xml" file are stored in the database when the **save_database** property is set to "yes". All fields are stored as text and in UTF-8 URL encoded format for optimal compatibility with any database. All strings are truncated to at most 4000 characters since that is the maximum size of a varchar2 in Oracle. MySQL does not have a size limit on the TEXT field type. The SQL Server script uses varchar that has a limit of 8000 characters.

The **task_tree** and **users** tables are synchronized with the database while the **reports** and **messages** tables are saved to the database. History items are added to the **history_list** table directly when a change is made to a task.

Error Messages

The client normally displays the translated text of an error returned from the server but in some cases, with the web interface, the error can not be translated.

The list below shows the English translation of the errors that may be returned from the server (or client):

Error	Description
-1	Failed to call service, the server may be unavailable. Try to work Offline.
-2	Service failed, unknown error.
-3	Unknown or unavailable service called. The client might not be the same version as the server or the service might be prevented by your license key. Time to update the server or license.
-4	To few parameters to service. The client might not be the same version as the server.
-5	Incorrect password. Try again.
-6	Not allowed to update this task. The task is not assigned by or to you.
-7	Not allowed to change the assigned to field. Can only be changed by the user that assigned the task, or by an Administrator.
-8	Not allowed to view this task.
-9	Not allowed to move a task out of its parent. Try to move using drag and drop or copy/move to.
-10	Not allowed to view the root task.
-11	Could not create new task.
-13	Not allowed to delete this task. Can only be deleted by the user that assigned the task or an Administrator.
-14	Task not found. A task might no longer be visible for a view or report. Try removing a few reports.
-15	Parent task not found.
-16	Invalid date format. One or more date fields was not correctly formatted.
-17	Unit not found. A user, role or group was not found.
-18	Not allowed to activate this task. Can only be activated by the user that assigned the task, or by an Administrator.
-19	Operation can only be performed by the creator of the unit.
-20	Not allowed to add more users. The license key for the server does not allow you to add more users.
-21	Failed to add attachment. An attachment could not be loaded. A probable cause is lack of memory. Try to increase the memory for the Java VM.
-22	Failed to retrieve attachment. An attachment could not be fetched from the server.
-23	This operation is not allowed for a restricted user.
-24	This operation is only allowed for administrators.
-25	Password has been cleared.
-26	Could not move task. The task can not be moved to that location.
-27	Failed to parse MS Project XML.
-28	User is not assigned to attachment. There is already an attachment with the same name in the system and that attachment is assigned to another user.
-29	Too many users in the system. The server license does not allow any more users at the moment, try to login later (there is a 5 minute timeout for users in the system).
-30	Not allowed to paste a reference under a reference.
-31	Not allowed to view this media.
-32	Invalid time format. One or more time fields were not correctly formatted.
-33	This operation is only allowed for the user that assigned the task or an

	Administrator.
-34	No report with that name found.
-35	Service call failed, unknown reason. An unexpected failure in one of the services. The client might not be the same version as the server.
-36	Message not found.
-37	Failed to send email.
-38	Could not move unit. The unit can not be moved to that location.
-39	Invalid integer format. One or more integer custom fields were not correctly formatted.
-40	Invalid float format. One or more float custom fields were not correctly formatted.
-41	Empty mandatory field. One or more mandatory custom fields were blank.
-42	This task has been changed since it was opened. Cancel your changes and reopen the task.
-43	Must specify a from or to date.
-44	Not a valid subtype. The task may not have a subtask with that task type.
-45	This operation is only allowed for assigned by, assigned to or an Administrator user.
-46	Unknown MIME type.
-47	Trying to define a new task type that already exists.
-48	A task must have a task type.
-49	Task is not valid.
-50	Invalid transition. Contains one or more invalid transitions of custom fields.
-51	Status is not a visible field and may not be updated manually.
-52	The root task may only be updated by the main Administrator (with user id 1).
-53	Supply a message to send.
-54	Failed to save to XML files or database.
-55	Item already in list
-56	Not a valid predecessor
-57	Report must have a name
-58	Insert failed, unit already exists.
-59	Invalid action, no action performed.
-60	Not able to complete task since it has uncompleted subtasks. Complete or remove the uncompleted subtasks first.