

The TaskJuggler Manual

Chris Schläger

Marc Rühschneck

The TaskJuggler Manual

by Chris Schläger and Marc Rührschnack

Copyright © 2001, 2002, 2003, 2004, 2005 Chris Schläger

This document describes TaskJuggler version 2.1.0

Table of Contents

About this document.....	i
1. Introduction.....	1
1.1. About TaskJuggler	1
1.2. Features and Highlights	1
1.3. TaskJuggler on the Web	2
1.4. Change Log	2
1.4.1. Version 1.0.0 (2002-03-14)	3
1.4.2. Version 1.0.1 (2002-03-15)	3
1.4.3. Version 1.1 (2002-05-27)	3
1.4.4. Version 1.2 (2002-06-17)	4
1.4.5. Version 1.3 (2002-07-30)	5
1.4.6. Version 1.4 (2002-12-18)	6
1.4.7. Version 1.4.1 (2003-02-24)	7
1.4.8. Version 1.4.2 (2003-03-10)	8
1.4.9. Version 1.9.0-unstable (2003-06-25)	9
1.4.10. Version 1.9.1-unstable (2003-07-29)	10
1.4.11. Version 1.9.2-unstable (2003-09-05)	11
1.4.12. Version 2.0.0 (2003-11-24)	11
1.4.13. Version 2.0.1 (2004-03-08)	12
1.4.14. Version 2.1.0 (2005-03-07)	12
2. Installation.....	15
2.1. Obtaining TaskJuggler	15
2.2. Requirements	15
2.2.1. The command line tool taskjuggler.....	15
2.2.2. The TaskJuggler Documentation	15
2.2.3. txx2gantt - Transforms XML files to Postscript Gantt charts.....	16
2.2.4. Contributed Stuff.....	17
2.3. Compilation and Installation.....	17
3. Usage	19
3.1. Basics	19
3.2. General Usage	19
3.3. The Command Line Options.....	19
3.4. Reporting Bugs and Sending Feedback	20
4. Your First Project	21
4.1. Starting the project.....	21
4.2. Global Attributes.....	22
4.3. Declaring Resources	23
4.4. Declaring Accounts.....	24
4.5. Specifying the Tasks	24
4.6. Specifying Milestones.....	29

4.7. Generating Reports of the scheduled Project.....	31
4.7.1. Generating HTML Task Reports	32
4.7.2. Generating HTML Resource Reports	33
4.7.3. Generating HTML Account Reports	34
4.7.4. Generating XML Reports	35
5. Language Reference.....	37
5.1. Comments	37
5.2. Attribute Classes	37
5.2.1. DATE	37
5.2.2. DATEINTERVAL	37
5.2.3. GLOBAL_ID	37
5.2.4. ID	37
5.2.5. INTEGER	38
5.2.6. LOGICALEXPRESSION	38
5.2.7. REAL	40
5.2.8. SORTINGCRITERIA	40
5.2.9. STRING	40
5.2.10. TIME.....	40
5.2.11. TIME.....	40
5.2.12. UNIT.....	40
5.2.13. WEEKDAY	41
6. Property Reference	43
6.1. The TJP File.....	43
6.2. account <id> <name> [<type>]	43
6.3. account <accountid>	44
6.4. accumulate	44
6.5. allowredefinition	45
6.6. allocate <resource>	45
6.7. alternative <resource> [, <resource> ...]	45
6.8. barlabels <mode>.....	46
6.9. booking <start> <end> <task>	47
6.10. caption <text>	48
6.11. celltext <text>	48
6.12. cellurl <url>	49
6.13. columns <columnid> [, <columnid> ...]	49
6.14. complete <percent>.....	51
6.15. copyright <text>.....	52
6.16. credit <date> <description> <amount>	52
6.17. csvaccountreport <filename>	53
6.18. csvresourcereport <filename>	53
6.19. csvtaskreport <filename>	54
6.20. currency <text>	54

6.21. currencyformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>	55
6.22. dailymax <value> <unit>	56
6.23. dailyworkinghours <hours>	56
6.24. depends <task> [, <task> ...]	57
6.25. disabled	58
6.26. duration <value> <unit>	58
6.27. efficiency <value>	59
6.28. effort <value> <unit>	60
6.29. end <date>	60
6.30. end <date>	61
6.31. endbuffer <percent>	61
6.32. endcredit <amount>	62
6.33. export <filename>	62
6.34. extend <property>	63
6.35. flags <flag> [, <flag> ...]	64
6.36. flags <flag> [, <flag> ...]	65
6.37. gapduration <value> <unit>	65
6.38. gaplength <value> <unit>	66
6.39. headline <text>	66
6.40. hideaccount <logicalexpression>	67
6.41. hidecelltext <expression>	67
6.42. hidecellurl <expression>	68
6.43. hideresource <logicalexpression>	68
6.44. hidetask <logicalexpression>	68
6.45. htmlaccountreport	69
6.46. htmlresourcereport	69
6.47. htmlstatusreport <file>	70
6.48. htmltaskreport	70
6.49. htmlweeklycalendar <file>	71
6.50. include <file>	71
6.51. journalentry <date> <text>	72
6.52. kotrusid <id>	73
6.53. label <text>	73
6.54. length <value> <unit>	74
6.55. limits	74
6.56. load <factor>	76
6.57. loadunit <unit>	76
6.58. macro <id>	77
6.59. mandatory	78
6.60. maxeffort <workingdays>	78
6.61. maxend <date>	79
6.62. maxstart <date>	79

6.63. minend <date>	80
6.64. minstart <date>	80
6.65. milestone	81
6.66. note <text>	81
6.67. monthlymax <value> <unit>	82
6.68. now <date>.....	82
6.69. numberformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>	83
6.70. persistent	83
6.71. priority <value>	84
6.72. precedes <task> [, <task> ...]	85
6.73. project <id> <name> <version> <start> <end>.....	86
6.74. projectid <id>.....	87
6.75. projectids <projectid> [, <projectid> ...].....	87
6.76. projection	88
6.77. rate <value>	88
6.78. rawhead <html>	88
6.79. rawstylesheet <stylesheet>.....	89
6.80. rawtail <html>.....	89
6.81. reference <url>.....	90
6.82. resource <id> <name>	90
6.83. responsible <resource>	91
6.84. rollupaccount <logicalexpression>	91
6.85. rollupresrouce <logicalexpression>	91
6.86. rolluptask <logicalexpression>	92
6.87. scenario <id> <name>	92
6.88. scenarios <scenarioid> [, <scenarioid> ...]	93
6.89. scheduled.....	93
6.90. scheduling <type>.....	94
6.91. select <mode>	94
6.92. sloppy <value>.....	96
6.93. shift <id> <name>.....	96
6.94. shift <shiftid> [<dateinterval>].....	97
6.95. shorttimeformat <format>.....	98
6.96. showprojectids	98
6.97. sortaccounts <criteria> [, <criteria> ...]	99
6.98. sortresources <criteria> [, <criteria> ...]	99
6.99. sorttasks <criteria> [, <criteria> ...]	100
6.100. start <date>	101
6.101. start <date>	102
6.102. startbuffer <percent>.....	102
6.103. startcredit <amount>	103
6.104. statusnote <text>	103

6.105. subtitle <text>	103
6.106. subtitleurl <url>	104
6.107. supplement <type>.....	104
6.108. task <id> <name>	105
6.109. taskattributes <attribute> [, <attribute> ...]	106
6.110. taskprefix <prefix>	106
6.111. taskroot <root>.....	107
6.112. timezone <zone>.....	107
6.113. timeformat <format>.....	108
6.114. timingresolution <value> <unit>	110
6.115. title <text>.....	110
6.116. titleurl <url>	111
6.117. vacation <name> <interval>	111
6.118. vacation <interval>	111
6.119. version <number>	112
6.120. weeklymax <value> <unit>	112
6.121. weekstartsmunday	113
6.122. weekstartssunday	113
6.123. workinghours <weekday> <interval> [, <interval> ...].....	113
6.124. xmlreport <file>	114
6.125. yearlyworkingdays <days>	114
7. The Example: Accounting Software.....	117
8. Migrating from TaskJuggler 1.x to 2.0.....	125
8.1. Achiving compatibility	125
8.1.1. Syntax changes.....	125
8.1.2. Scheduler changes.....	126
9. Questions and Answers	127
9.1. General Questions.....	127
9.2. Compiling and installation.....	127
9.3. Usage.....	127
10. Copyright.....	129
11. Trademarks	131

About this document

This document describes TaskJuggler version 2.1.0

The TaskJuggler manual consists of two parts. The first part contains general usage information and a tutorial. The tutorial is highly recommended for all first-time TaskJuggler users. It is a very good introduction to many TaskJuggler concepts and features. The second part is the language reference.

Chapter 1. Introduction

1.1. About TaskJuggler

TaskJuggler is a project planing tool for Linux and UNIX system-based operating systems. Whether you want to build a skyscraper or plan your colleagues shifts for the next month - TaskJuggler is the tool for you.

Instead of clicking yourself painfully through hundreds of dialog boxes you specify your TaskJuggler project in a simple text format. You write down all your tasks with their dependencies and other properties using the graphical front-end or your favorit text editor. The information is processed by the TaskJuggler program and you will get all sorts of reports in an interactive format or as HTML or XML format. The HTML files can be viewed and printed with any web browser that supports Cascading Style Sheets. JavaScript support is recommended. The XML files are used by the Gantt- and PERT chart generators and the KDE (<http://www.kde.org>) Konqueror (<http://konqueror.kde.org>) plug-in.

TaskJuggler does not only honor the task interdependencies but also takes resource constrains and prioritization into account. Using TaskJugglers powerful filtering and reporting algorithms you can create a variety of task lists, resource usage tables, status reports, project calendars and project accounting statements.

Since TaskJuggler is not constrained by the performance requirements of real-time editing it can offer a much broader set of features not found in any of the WYSIWYG (What You See Is What You Get) project planing tools. The the project description language is easy to learn and supports you very effectively during your planing and tracking process.

1.2. Features and Highlights

- Automatic scheduling of interdependent tasks with resource conflict solver.
- Powerful project description syntax with macro support.
- Graphical front-end to enter and view your project.
- Flexible working hours and vacation handling.
- Support for shifts.
- Multiple time zone support.

- Flexible resource grouping.
- Project accounting support.
- Task may have initial, finishing and running costs.
- Resource may have running costs.
- Support for simple profit and loss analysis.
- HTML and XML report generation.
- Gantt and PERT chart generators.
- Support for multiple project scenarios.
- Project tracking support.
- Groupware support by using a revision control system such as CVS or RCS on the project description files.
- Support for central resource allocation database.
- Support for cascaded and nested projects.
- Import and export of sub-projects.
- Unicode support.

1.3. TaskJuggler on the Web

The official TaskJuggler web page can be found at <http://www.taskjuggler.org> (<http://www.taskjuggler.org>).

Since the developers are mostly busy project managers themselves, we have created a forum for users to help each other.

The TaskJuggler user mailing list is hosted by Novell (<http://www.novell.com>). You can subscribe and unsubscribe at <http://www.suse.de/en/support/maillinglists/> (<http://www.suse.de/en/support/maillinglists/>).

1.4. Change Log

1.4.1. Version 1.0.0 (2002-03-14)

- Initial stable public release.

1.4.2. Version 1.0.1 (2002-03-15)

- Fixed completely broken global vacation handling.
- Added test case for vacation handling to test suite.

1.4.3. Version 1.1 (2002-05-27)

- Added some reports to the example file, so users actually get a result of the TaskJuggler run.
- Support for later completion of task and resources added. By writing 'supplement task <ID> { ... }' an already defined task can be extended. So it's easier now to create a file which contains the vacations for all resources separate from the resource definition itself.
- Extended expression parser to work on string type values as well.
- `logicalexpression` for `hidetask`, `rolluptask` etc. can now contain functions as well. Currently there is support for 'istask', 'isresource', 'isaccount', 'issubtaskof', 'contains', 'ismilestone'.
- Moved the docs directory from TaskJuggler subdir to topdir.
- Added feature list and change-log to the documentation.
- `property_reference` is now sorted in alphabetical order.
- Added lots of missing attributes to `htmlaccountreport`.
- Added missing `export` report to documentation. Export reports can now contain the scheduled tasks as well as the resource allocations.

- New keywords `planbooking` and `actualbooking` to enter fixed bookings of resources in the resource declaration.
- Added new example project to illustrate the use of export in big projects that are split into sub projects.
- HTML comments in HTML report files are now using correct syntax.
- Partial fix for correct time zone handling.
- Support for STDIN reading and STDOUT writing added. This can be used when calling TaskJuggler from CGI scripts.

1.4.4. Version 1.2 (2002-06-17)

- Fixed sorting by ID for all HTML reports.
- Fixed bug in vacation handling. Vacations that started before the project were silently ignored.
- Added support for `taskattributes` to export report.
- XML Output changes: Basically the XML output is more simple to parse, some values were added and corrected.
- Added a first simple TaskJuggler XML-output viewer for KDE. See `ktjview/README` for installation. Configure with KDE support enabled.
- Disabled ical support by introducing the `HAVE_ICAL` switch in the code. The switch is not yet configure supported, but building with `--with-kde-support` should work now without failing on missing libical.
- Support for URLs in HTML reports added.
- Legacy HTML elements have been removed from HTML reports. TaskJuggler now creates pure HTML 4.0 code.
- Added support for insertion of raw HTML code into reports. This can be achieved with `rawhead` and `rawtail`.
- Added support for user defined style sheets in HTML reports by using the `rawstylesheet` attribute.
- Strings can now be enclosed in either single or double quotes. A single quoted string may contain double quotes and vice versa.
- Working hours can now be declared on project level. This also determines if a day is considered a working day or not.

- With `startbuffer` and `endbuffer` you can now specify that there might be some air left in a certain task.
- Remo's Gantt chart generators have been included in the `Contrib` directory.

1.4.5. Version 1.3 (2002-07-30)

- This release features some bigger cleanup changes. Some of them do break compatibility with older version of taskjuggler. While we try very hard to avoid such situations, we do prefer to have a consistent and logical language. Since the TaskJuggler user base is still comparatively small, we decided to break compatibility now rather than later. The changes are fairly minor, so they won't affect many users. Please see further down for more details.
- Added Perl/Tk tool to view Gantt charts and other project information.
- Added PERT-chart generator from Philippe Midol-Monnet.
- Added support for shifts in `shift` and `task allocate shift`.
- Fixed vim syntax highlighting. Some keywords were missing.
- Export report had syntax bug when milestones were present. Fixed.
- Fixed handling of week, month and year duration specifications.
- `now` and `timingresolution` are no longer properties. They are now optional attributes of project. They currently still work as properties as well but a warning is issued and they will be removed in the next major release.
- `dailyworkinghours` and `yearlyworkingdays` have been implemented to allow the user for better control over the conversion from working days to working hours.
- Added support for a `select` function for alternative resource allocations.
- All load values in HTML reports can now be scaled by specifying a `loadunit`.
- Improved readability of scheduler error messages.
- Added new example project to the `Examples` directory to illustrate how to create shift schedules with TaskJuggler.
- Fixed scheduler for working hours around midnight. This bug affected shifts as well as general working hours.
- Extended timezone support. TaskJuggler will now operate properly when `TZ` environment variable is set.

1.4.6. Version 1.4 (2002-12-18)

- Only export references to tasks which are exported in the same report.
- Allow supplements of tasks within task definitions.
- Converted documentation to DocBook. We now have a much nicer and more structured manual. A printable version is available as well now.
- Fixed HTML code for bookedlight cells. Those were rendered without background on some browsers.
- Added support for multi-level sorting in reports. `sorttasks` and `sortresources` now take multiple criteria.
- Several bugs in the sorting direction code have been fixed. `startup`, `startdown`, `endup` and `enddown` have been replaced by `planstartup`, `planstartdown`, `planendup` and `planenddown`.
- The optional attribute `taskprefix` has been added to include. This allows other projects to be added at arbitrary points in the task tree as sub projects.
- Include statements within tasks are no longer supported. They lead to ambiguous interpretation of certain attributes.
- The optional attribute `taskroot` has been added to export. This allows to export sub tasks of a tasks to be exported as root-level tasks.
- The project file reader has been made fully Unicode aware. It is now possible to use non-ASCII characters in text strings and comments.
- Two new functions have been added for use in logical expressions. `isplanallocated` and `isactualallocated` can be used to show only resources that have been allocated to a certain project in a given time frame.
- Made week of year calculation ISO 8601:1988 and DIN 1355 compliant. This also affects the month and year correlation in weekly reports. You can use the optional project attributes `weekstartssunday` and `weekstartsmonday` to specify whether you like to start you week on Sunday or Monday.
- Support for a `flags` columns added to HTML reports.
- Sub tasks do now inherit the dependencies of their container tasks. Specifying dependencies after sub tasks is now illegal since they would be only used for checking, but not for scheduling.
- The logic checker for task attributes has been completely rewritten. Since it probably catches some more errors, you might have to fix your project now. Such cases would have resulted in wrong results anyhow. Lots of test cases have been added to the test suite to validate the checker.

- The error reporting has been drastically improved. The messages should be more precise now and errors that are triggered by other errors should be not so prominent anymore.
- A new report type has been added. `htmlweeklycalendar` can be used to generate weekly calendars.
- The format of time specifications in HTML reports is now configurable via `timeformat` and `shorttimeformat`
- The keyword `xmltaskreport` is now deprecated. It has been replaced by `xmlreport`. The rest of the syntax remains identical.
- The tool `xml2gantt.pl` has been renamed to `tjx2gantt` and moved from the Contrib directory to the main directory. The tool `xml2png` has been removed.
- Included new version 0.2.2 of TJ-Pert from Philippe.
- The load numbers on the bars of the HTML task and resource reports can now be turned on and off using the `barlabels` attribute.
- The HTML reports feature now 3 kind of index numbers. The sequence number reflects the order of declaration in the project files. The index is a logical order based on the hierarchy and other attributes. The number is the index in the generated list. What used to be the `no` column is now the `index` column.
- The sequence of properties in the project file can now be used as sorting criteria as well.

1.4.7. Version 1.4.1 (2003-02-24)

- Another redo of the loop detector. Now checking tasks not only forward, but also backwards. Insufficiently specified task boundaries are no longer detected, since they are flagged with missing start/end messages after the scheduler run.
- The dependency loop detector can now be skipped with the `--nodepcheck` command line option.
- The dependency loop detector runs now significantly faster for larger projects.
- Broken HTML table when `schedule` was used with `showactual` fixed.
- HTML reports can now show a column with the completion degree and the completion status. The rows can also be sorted by these new columns.

- The HTML and XML reports are now UTF8 encoded. This should eliminate problems with languages that require non-latin1 character sets.
- Currency values in HTML reports are now always right aligned.
- A bug in the handling of nested Resources and Shifts has been found and fixed. The bug lead to wrong load values for all nested resources. The bug was introduced between versions 1.2 and 1.3.
- If some container tasks could not be scheduled due to problems with a sub task no error message was generated. This has been fixed now.
- Fixed scheduling of container tasks, so that container tasks with only milestones get properly scheduled.
- Only export min/max start/end times when they were explicitly specified and do no longer inherit project start/end times for this purpose.
- htmlaccountreport now supports quarterly and yearly calendar columns.
- Fixed XML reports so that milestone end dates are same as start dates.

1.4.8. Version 1.4.2 (2003-03-10)

- Indentation for tree structure in HTML reports is now done with cell margins. This should no longer look bad if the label gets wrapped by the browser.
- HTML tables now use explicit head and body sections. This should repeat the table header when printing HTML reports from some browsers.
- Fixed segfault in XML report generation. Only plan values are now exported in XML report.
- Task scheduling is also set when a fixed start or end date is specified.
- Better error reporting for syntax errors in macros. The call stack with full arguments is included in the error message now.
- The cost column in HTML task or resource reports now only contains cost. Support for a revenue and profit column has been added.
- Abbreviated month name are now encoded properly in non-Latin1 languages as well.
- Milestones in HTML calendars are now visible in text browsers and printouts as well.
- New attribute reference added to task.

1.4.9. Version 1.9.0-unstable (2003-06-25)

- A new HTML report type for status report has been added. See `htmlstatusreport` for details.
- HTML reports are now a lot more flexible. New CSS elements have being used and the table elements are customizable now. See optional column attributes for details.
- Support for user-defined attributes has been added.
- Resource allocations can now be made mandatory.
- The format of numbers and currency values can now be specified with `numberformat` and `currencyformat`. The old keyword `currencydigits` has been deprecated.
- All reports have now support for daily, weekly, monthly, quarterly and yearly calendars. Task lines now contain Gantt-chart like bars.
- HTML reports got the additional columns `hierarchno` and `hierarchindex`.
- Sveral new query functions and operators for logical expressions have been added.
- Scenario specific task attributes can now be prefixed with the scenario ID followed by a colon. The attributes starting with 'plan' or 'actual' have been deprecated.
- Fixed the URLs for task and resource names in HTML reports.
- Cost, revenue and profit values as well as effort values are now indented in tree sorting mode for all HTML reports.
- Length and duration tasks with resource allocations are no longer trimmed to the first and last resource allocation.
- Fixed rounding error in effort calculation that led to the allocation of an extra time slot in some cases.
- Fixed wrong scheduling of tasks that had a length or duration specified as hours or minutes.
- 'length' based task now use the global working hours and global vacation settings as a criteria of what is a working day. The tasks now always end during working hours and not at midnight.

- `isplanallocated` and `isactualallocated` had broken time interval handling. This is fixed now.
- `workinghours` and `currency` are no longer global properties. They are now optional attributes of the project property.
- The scenario name is no longer displayed by default if more than one scenario is included in a report. A column `scenario` must be explicitly added if the scenario name should be reported for each line. The attributes `'showactual'` and `'hideplan'` have been deprecated. The `scenarios` attribute now controls which scenarios should be shown.
- Container tasks in export reports no longer have fixed start and end date if they have their sub tasks exported as well.
- Resource allocations are now inherited from parent tasks.

1.4.10. Version 1.9.1-unstable (2003-07-29)

- A new class of reports has been added. CSV reports (Comma separated values) are useful to import TaskJuggler reports into other productivity applications such as spreadsheets. The new reports are called `csvtaskreport`, `csvresourcereport` and `csvaccountreport`.
- HTML Calendars have now a navigation aid. Moving a mouse over a cell will show the date and task/resource id in the browser status bar.
- Background cells in HTML calendars are now merged. This makes taskjuggler report generation faster and reduces the size of HTML report files.
- The export report can now be a main project file as well.
- A new keyword for taskattributes of export reports has been introduced. The keyword `all` causes all supported task attributes to be exported.
- Various speed improvements.
- The broken milestone symbol in HTML calendars has been fixed.
- HTML reports now have a black grid to separate the cells. This enhances readability both on the screen and on printouts.
- The functions for Logical Expressions are now using capital letters to improve their readability. The all lowercase versions are still supported, but the recommended versions are now the ones with intermixed uppercase letters. `isTaskOfProject` was added as new query function.

- The maximum allocation of a resource for a task is no longer limited by default. `maxeffort` now defaults to 0 (unlimited) instead of 1.0 (8 hours per day). To have the same behaviour as in TaskJuggler 1.x version you need to specify `maxeffort 1.0` before any resource definition. This change was made since many users were confused when after increasing the daily working hours resources were still only allocated 8 hours per day.

1.4.11. Version 1.9.2-unstable (2003-09-05)

- Support for new XML format has been added. The old format is still supported. TJ can read both old and new format XML files but will use the new XML format for output.
- The property `projectids` has been added. It is used in export reports to declare all the project IDs that are used in the report.
- Resource booking periods can now overlap with off-duty hours, vacation or other task assignments. This is controlled by the `sloppy` attribute.
- Effort based tasks now correctly recognize if the effort was partially specified with booking attributes. The effort is no longer allocated on top of the bookings.
- You can now reference environment variables by writing `$(VAR)` as a means to pass runtime values to taskjuggler.
- Several inconsistencies and off-by-one errors with respect to task end times have been fixed.
- TaskJuggler can now create 'make' compatible dependency information.
- The number of errors after which TaskJuggler stops processing is now configurable via a command line option.

1.4.12. Version 2.0.0 (2003-11-24)

- Fixed completion coloring in HTML reports.
- Fixed segfault in certain cases of inherited resource allocations.
- Macro names in macro calls can now be prefixed by a questionmark to suppress warnings if the macro is undefined.

- Microsoft and MacOS text files are now read in correctly.
- Report cells can be left empty and URLs can be omitted controlled by a logical expression. This is controlled by `hidecelltext` and `hidecellurl`.
- New functions `isATask`, `isAResource` and `isAnAccount` can now be used in logical expressions.
- XML version 2 files are now compressed with `zlib`.

1.4.13. Version 2.0.1 (2004-03-08)

- Fixed handling of resource allocations with multiple shift intervals.
- Fixed double-quoting of HTML special characters such as umlauts.
- Added query function `isDutyOf()` to select tasks where a certain resource has been assigned to.
- The contents of XML reports can now be limited with the usual filter mechanisms. Support for `hideresource`, `hidetask`, `rollupresource` and `rolluptask` has been added. Also scenario filtering was implemented for XML reports.
- Weekly, monthly, quarterly and yearly HTML reports now have resource vacations as well. If the vacation fills the complete report cell term, the cell has a yellow background.
- Fixes for building TaskJuggler on FreeBSD added.
- `maxeffort` and `load` have been replaced by the far more flexible concept of limits.

1.4.14. Version 2.1.0 (2005-03-07)

- TaskJuggler now has a nice face. Beside the commandline application `taskjuggler`, you can now use `TaskJuggler` or `ktjview2` as a graphical user interface to enter and schedule your projects.
- New optimizer that achieves much better resource selection resulting in shorter overall project times.

- Passive resources like meeting rooms, machines and the like, that do not contribute to the effort of a task can now be modelled by setting their efficiency to 0.0.
- Added critical path analyser. Each task is rated and the rating can be listed in the HTML and CSV report.
- New task state added. When a task is not finished by the planned end date, it now marked as `late`.
- Task dependency specifications (depends or precedes can now have optional gap specification. It is possible to specify the gap in calendar time (gapduration) or working time (gaplength).
- The speed of report generation has been significantly improved. This is especially true for reports that make use of filter functions.
- Added `status` and `statusNote` to XML reports.
- Added some missing properties to the documentation. Mainly the sorting criterias were missing.
- Fixed a memory leak during XML report generation.
- Fixed scheduling of nested task that had an external dependency and an inherited start/end date.
- Limits of resource allocations with multiple alternatives are now correctly handled. The limits were applied to each individual resource instead of to the whole allocation.
- The task priority is now always properly respected. Due to a bug in the scheduling algorithm a heavy mixture of ALAP and ASAP tasks with various levels of priorities, ALAP tasks were treated more favorable then they should have been treated. This fix can drastically reduce the scheduling speed when you have a heavy mixture of ALAP and ASAP tasks with varying priorities.
- The error checking and reporting of logical expressions has been drastically improved.
- The reports are now generated relative to their definition file and no longer relative to the current working directory where you started the program.

Chapter 2. Installation

2.1. Obtaining TaskJuggler

TaskJuggler may be obtained from the following web site:

<http://www.taskjuggler.org> (<http://www.taskjuggler.org>)

2.2. Requirements

TaskJuggler uses many additional tools. We have avoided to re-invent the wheel again wherever possible, at the same time we tried to keep the dependency list reasonably small. Nevertheless can it be challenging to the unexperienced user to build and install TaskJuggler from source code. Most users are probably better served by using binary packages from their distributor. Some distributions even have TaskJuggler included like the SuSE (<http://www.suse.de/en>) distribution.

This version of TaskJuggler was built and tested on SuSE Linux 9.1, 9.2 and 9.3 on various hardware platforms.

2.2.1. The command line tool taskjuggler

To build and run TaskJuggler, you need:

- qt3.2 — The Qt C++ class library (<ftp://ftp.trolltech.com/qt/source>) version 3.3 or higher
- GNU Compiler Collection — We are currently using GCC 3.2 to develop TaskJuggler. Since we try to write the code platform independent and ANSI standard compliant it should work with other compilers as well.

These requirements are enough to build and use the command line program that translates project descriptions into HTML or XML reports. If you want to build this documentation or use the tools to process the XML files you need to take care of additional dependencies. If any of the following dependencies are not matched, the configure script will issue a warning but will not fail.

2.2.2. The TaskJuggler Documentation

- DocBook4 — The DocBook (<http://www.oasis-open.org/docbook/>) system with additional stylesheets and utilities.
- DocBook-Toys — The DocBook-Toys (<http://www.suse.de/~ke/docbook-toys>) from Karl Eichwalder.
- OpenJade — The OpenJade (<http://openjade.sourceforge.net/>) system.
- JadeTeX — If you want to build the Postscript version of the documentation you need the JadeTeX (<ftp://ftp.dante.de/tex-archive/macros/jadetex>) macro package for teTeX (<http://www.tug.org/teTeX>).
- XSLT Processor — like xsltproc from the libxslt (<http://xmlsoft.org/XSLT/>) package.
- dvips — This is usually part of your TeX system like te_latex (<http://www.tug.org/teTeX>).

2.2.3. tjsx2gantt - Transforms XML files to Postscript Gantt charts

tjsx2gantt is installed by default and generates Postscript Gantt charts from the XML files that TaskJuggler generates. tjsx2gantt is written in Perl.

- Perl 5.6 — The Perl (<http://www.perl.com/pub/a/language/info/software.html#stable>) interpreter and the following Perl modules.
- XML::Parser — Available from CPAN (<http://www.cpan.org/modules/by-module/XML/XML-Parser-2.31.tar.gz>)
- PostScript::Simple — Available from CPAN (<http://www.cpan.org/modules/by-module/PostScript/PostScript-Simple-0.04.tar.gz>)
- Date::Calc — Available from CPAN (<http://www.cpan.org/modules/by-module/Date/Date-Calc-5.3.tar.gz>)
- Class::MethodMaker — Available from CPAN (<http://www.cpan.org/modules/by-module/Class/Class-MethodMaker-1.06.tar.gz>)
- Data::Dumper — Available from CPAN (<http://www.cpan.org/modules/by-module/Data/Data-Dumper-2.101.tar.gz>). This

module is only needed for debugging.

- Poster — The Poster (<http://www.geocities.com/SiliconValley/5682/poster.html>) utility cuts big Postscript pictures into printable chunks. It's handy but optional as well.

2.2.4. Contributed Stuff

Several people have contributed useful tools for TaskJuggler users. These tools can be found in the Contrib directory of the source code package. Please refer to the included README files for installation information.

2.3. Compilation and Installation

Before compiling TaskJuggler you need to set the `QTDIR` environment variable to the base directory of your Qt version. This is normally `/usr/lib/qt3`.

To compile and install TaskJuggler on your system, type the following in the base directory of the TaskJuggler distribution:

```
% ./configure
% make
% make install
```

Since TaskJuggler uses `autoconf`, you should have no trouble compiling it. TaskJuggler has some weak or non-mandatory dependencies. If any of them are not satisfied, a warning is issued. In case a mandatory dependency is not found, an error is displayed and the configuration program stops. If you have problems, please report them to the TaskJuggler mailing list at `taskjuggler@suse.com` (<mailto:taskjuggler@suse.com>) in English. Maybe someone there can help you out.

In case you want to build TaskJuggler without the graphical front-ends, you can disable them during the configuration process:

```
% ./configure --with-kde-support=no --prefix=/usr/local
% make
% make install
```


Chapter 3. Usage

3.1. Basics

TaskJuggler uses one or more text files to describe a project. The main project should be placed in a file with the `.tjp` extension. This main project may contain several sub-projects. Such sub-projects should be placed in files with the `.tji` extension. These sub-projects are included in the main project during compile time.

When TaskJuggler is used with other tools the project description or the reports may also be in XML format. The recommended extension for such files is `.tjx`.

3.2. General Usage

To use the graphical front-end simply start it from the menu. Depending on your Linux Distribution you will find it under Office/Project Management or a similar entry. To start it from a shell type

```
% TaskJuggler
```

or

```
% ktjview2
```

The commandline version of TaskJuggler works like a compiler. You provide the source files, it computes the contents and creates the output files.

Let's say you have a project file called `AcSo.tjp`. It contains the tasks of your project and their dependencies. To schedule the project and create report files you have to ask TaskJuggler to process it.

```
% taskjuggler AcSo.tjp
```

TaskJuggler will try to schedule all tasks with the specified conditions and generate the reports that were requested with the `htmltaskreport`, `htmlresourcereport` or other report properties in the input file.

3.3. The Command Line Options

<code>--help</code>	Print all command line options with a short description.
<code>--version</code>	Print the version and copyright info.
<code>-v</code>	Same as ' <code>--version</code> '.
<code>-s</code>	Stop TaskJuggler run after syntax check. This option is for testing and debugging.
<code>-M</code>	Output fragment of a Makefile that lists the dependencies of the passed TaskJuggler.
<code>--makefile <file></code>	Similar to ' <code>-M</code> ' but write the dependency information into the specified file.
<code>--maxerrors N</code>	Specifies the number of errors after which TaskJuggler stops checking the input.
<code>--nodepcheck</code>	Do not check for dependency loops. The loop detector uses an algorithm that never fails.
<code>--debug N</code>	Print debug output, N must be between 0 and 4, the higher N the more output is printed.
<code>--dbmode N</code>	Limit debug output to certain modules of the code. N is a bit mask. Each set bit enables a module.
<code>--updatedb</code>	Update the Kotrus database with the new resource usage information.

Calling TaskJuggler with `.` as input filename will cause reading from stdin. To redirect output to stdout you can specify `--` as filename for reports. This feature can for example be used to generate dynamic HTML pages from CGI scripts.

3.4. Reporting Bugs and Sending Feedback

All official releases of TaskJuggler are meant to be stable releases unless explicitly noted differently. But our test suite is still very small and some features cannot be tested automatically very well. So it's very likely that your current version of TaskJuggler contains some bugs. If you find a bug please follow this procedure:

- Read this manual to make sure that it is actually a bug and not a feature.
- Check the TaskJuggler web page (<http://www.taskjuggler.org>). Possibly the bug was already found and a patch or workaround exists.
- Try to create a test project that is as small as possible but still reproduces the bug.
- Send the test project and a detailed description of the problem to the developer mailing list at taskjuggler-devel@suse.com (<mailto:taskjuggler-devel@suse.com>).

Chapter 4. Your First Project

We have mentioned already that TaskJuggler uses plain text files that describe the project to schedule it. As you will see now, the syntax of the file is easy to understand and very intuitive. This chapter will walk you step by step through your first project. You create the project plan for a made-up accounting software project. Refer to Chapter 7 for a full listing of the example. This project demonstrates some of the basic features of TaskJuggler for more advanced features please refer to Chapter 5.

4.1. Starting the project

To start a TaskJuggler project file you need to use the project property.

```
project acso "Accounting Software" "1.0" 2002-01-16 2002-04-26 {  
  now 2002-03-04  
  timeformat "%Y-%m-%d"  
  currency "EUR"  
  scenario plan "Plan" {  
    scenario delayed  
    "Delayed"  
  }  
}
```

It tells TaskJuggler the default project ID, a short name for your project, a version number and a start and end date. The start and end dates don't need to be exact, but must enclose all tasks. It specifies the time interval the TaskJuggler scheduler will use to fit the tasks in.

All TaskJuggler properties have a certain number of fixed attributes and a set of optional attributes. Optional attributes are always enclosed in curly braces. In this example we use the optional attributes `now` to set the current day for the scheduler to another value than to the moment your invoke TaskJuggler. We pick a day during the above specified project period. So we always get the same results of a TaskJuggler run, no matter when we process our first project. The currency attribute specifies the unit of currency values.

Since each country seems to have it's own way of specifying dates, the format is configurable. Use the `timeformat` attribute to specify the default format. This is the format used for reports. It's not the format that you use in TaskJuggler project files. This format is fixed and must be

Year-Month-Day-Hours:Minutes:Seconds-Timezone. From Hours onwards, the values are optional and default to 0.

In this tutorial we would like to compare two scenarios of the project. The first scenario is the one that we have planned. The second scenario is how it really happened. The two scenarios have the same task structure, but the start and end dates of the task may vary. In reality we assume that the project got delayed, so we call the second scenario "Delayed". The scenario property is used to specify the scenarios. The delayed scenario is nested into the plan scenario. This tells TaskJuggler to use all values from the plan scenario also for the delayed scenario unless the delay scenario has it's own values. We'll see further down, how to specify values for a scenario.

4.2. Global Attributes

Besides finding suitable start and end dates of our project, we also like to do a simple profit and loss analysis. So we have to specify the default daily costs of an employee. This can be changed for certain employees later, but it illustrates an important concept of TaskJuggler - inheritance of attributes. In order to reduce the size of the TaskJuggler project file to a still readable minimum, properties inherit many optional attributes from their enclosing scope. We'll see further down, what this actually means. Here we are at top-level scope, so this is the default for all following properties.

```
rate 310.0
```

The rate attribute can be used to specify the daily costs of resources.

Macros are another TaskJuggler feature to keep project files small. Macros are text patterns that can be defined once and inserted many times further down the project. A macro always has a name and the text pattern is enclosed by square brackets.

```
macro allocate_developers [  
    allocate dev1  
    allocate dev2 { load 0.5 }  
    allocate dev3  
]
```

To use the macro you simply have to write `${allocate_developers}` and TaskJuggler will replace the term `${allocate_developers}` with the pattern.

We will use the macro further down in the example and then explain the meaning of the pattern.

4.3. Declaring Resources

A TaskJuggler feature that you will probably make heavy use of is `flags`. Once declared you can attach them to many properties. When you generate reports of the TaskJuggler results, you can use the flags to filter out information and limit the report to exactly those details that you want to have included.

```

flags team

resource dev "Developers" {
    resource dev1 "Paul Smith" { rate 330.0 }
    resource dev2 "Sebastien Bono"
    resource dev3 "Klaus Mueller" { vacation 2002-02-01 - 2002-02-05 }

    flags team
}
resource misc "The Others" {
    resource test "Peter Murphy" { maxeffort 0.8 rate 240.0 }
    resource doc "Dim Sung" { rate 280.0 }

    flags team
}

```

This snippet of the example shows the `resource` property. Resources always have an ID and a Name. IDs may only consist of ASCII characters, numbers and the underline character. All global TaskJuggler properties have IDs. They need to be unique within their property class. The ID is needed so that we can reference the property again later without having the need to write the potentially much longer name. Names are strings and as such enclosed in double quotes. Strings may contain any character, even non-ASCII characters. As you can see, resource properties can be nested. `dev` is a virtual resource, a team, that consists of three other resources.

`dev1`, alias Paul Smith costs more than the normal employee. So the declaration of `dev1` overwrites the inherited default rate with a new value of 330.0. The default value has been inherited from the enclosing scope, resource `dev`. Which in turn has inherited it from the global scope.

The declaration of the resource Klaus Müller uses another optional attribute. With `vacation` you can specify a certain time interval where the resource is not available.

Here you need to understand how TaskJuggler handles time intervals. Internally TaskJuggler uses the number of seconds after January 1st, 1970 to store any date. So all dates are actually stored with an accuracy of 1 second. `2002-02-01` specifies midnight February 1st, 2002. Again following the TaskJuggler concept of needing as little info as needed and extending the rest with sensible defaults, TaskJuggler adds the time `0:00:00` if nothing else has been specified. So the vacation ends on midnight February 5th, 2002. Well, almost. Every time you specify a time interval, the end date is not included in the interval. But the second before the date that you have specified is. So Klaus M'illers vacation ends `23:59:59` on February 4th, 2002.

Peter Murphy only works 6.5 hours a day (actually 6.4 hours). So we use the `maxeffort` attribute to limit his daily working hours. We could also define exact working hours using the `shift` property, but we ignore this for now.

Note that we have attached the flag `team` after the declaration of the sub-resources to the team resources. This way, they flags don't get inherited by the sub-resources. If we would have declared the flags before the sub-resources, then they would have the flags attached as well.

4.4. Declaring Accounts

The use of our resources will create costs. For a profit and loss analysis, we need to balance the costs against the customer payments. So that we don't get lost with all the amounts, we declare 3 accounts to credit the amounts to. We create one account for the development costs, one for the documentation costs and one for the customer payments.

```
account dev "Development" cost
account doc "Dokumentation" cost
account rev "Payments" revenue
```

The `account` property has 3 fixed attributes, an ID, a name and a type. The type can either be `cost` or `revenue`. For the analysis TaskJuggler subtracts the total amount of all cost accounts from the total amount from all revenue accounts.

Accounts can also be nested. Nested accounts may not have a type specified. They inherit the type of the top-level account.

4.5. Specifying the Tasks

Let's focus on the real work now. The project should solve a problem - the creation

of an accounting software. Since the job is quite complicated we break it down into several sub tasks. We need to do a specification, develop the software, test the software and write a manual. In TaskJuggler syntax this would look like that:

```
task AcSo "Accounting Software" {
    task spec "Specification"
    task software "Software Development"
    task test "Software testing"
    task deliveries "Milestones"
}
```

Similar to resources, tasks are declared by using the task keyword followed by an ID and a name string. All TaskJuggler properties have their own namespaces. This means, that it is quite OK to have a resource and a task with the same ID. Tasks may have optional attributes which can be tasks again, so tasks can be nested. In contrast to all other TaskJuggler properties, task IDs inherit the ID of the enclosing task as a prefix to the ID. The full ID of the `spec` task is `AcSo.spec`.

To track important milestones of the project, we also added a task called Milestones. This task, like most of the other task will get some sub tasks later on. We consider the specification task simple enough that we don't have to break it into further sub tasks. So let's add some more details to it.

```
task spec "Specification" {
    effort 20d
    ${allocate_developers}
    depends !deliveries.start
}
```

The effort to complete the task is specified with 20 man days. Alternatively we could have used the length attribute or the duration attribute. `length` specifies the duration of the task in working days while `duration` specifies the duration in calendar days. Contrary to effort these two don't have to have a specification of the involved resources. Since `effort` specifies the duration in man days, we need to say who should be allocated to the task. The task won't finish before the resources could be allocated long enough to reach the specified effort. Tasks with `length` or `duration` criteria and allocated resources, will last exactly as long as requested. Resources will be allocated only if available.

Here we use the above mentioned macro `allocate_developers`. The expression

```
${allocate_developers}
```

is simply expanded to

```
allocate dev1
allocate dev2 { load 0.5 }
allocate dev3
```

If you need to allocate the same bunch of people to several task, the macro saves you some writing. You could have written the allocate attributes directly instead of using the macro. Since the allocation of multiple resources to a task is a very common place for macro usage, we found it a good idea to use it in this example as well.

One more interesting thing to note is the fact that we like the resource `dev2` only to work 50% of the day on this task, so we use the optional attribute `load` to specify this.

For TaskJuggler to schedule a task it needs to know either the start and end criteria of a task, or one of them and a duration specification. The start and end criteria can either be fixed dates or relative dates. Relative dates are specification of the type "task B starts after task A has finished". Or in other words, task B depends on task A. In this example the `spec` task depends on a sub tasks of the `deliveries` tasks. We haven't specified it yet, but it has the local ID `start`.

To specify the dependency between the two task we use the `depends` attribute. The attribute must be followed by one or more task IDs. If more than one ID is specified, each ID has to be separated with a comma from the previous one. Task IDs can be either absolute IDs or relative IDs. An absolute ID of a task is the ID of this task prepended by the IDs of all enclosing tasks. The task IDs are separated by a dot from each other. The absolute ID of the specification task would be `AcSo.spec`.

Relative IDs always start with one or more exclamation marks. Each exclamation mark moves the scope to the next enclosing task. So `!deliveries.start` is expanded to `AcSo.deliveries.start` since `AcSo` is the enclosing task of `deliveries`. Relative task IDs are a little bit confusing at first, but have a real advantage over absolute IDs. Sooner or later you want to move tasks around in your project and then it's a lot less likely that you have to fix dependency specifications of relative IDs.

The software development task is still too complex to specify it directly. So we split it into sub tasks again.

```
task software "Software Development" {
  priority 1000
  task database "Database coupling"
  task gui "Graphical User Interface"
```

```
task backend "Back-End Functions"
}
```

We use the `priority` attribute to mark the importance of the tasks. 500 is the default priority of top-level tasks. Setting the priority to 1000 marks the task as most important task, since the possible range is 1 (not important at all) to 1000 (ultimately important). `priority` is an attribute that is inherited to sub tasks if specified before the sub tasks declaration. So all sub tasks of `software` have a priority of 1000 as well unless they have their own priority definition.

```
task database "Database coupling" {
    effort 20d
    depends !!spec
    allocate dev1, dev2
}
```

The work on the database coupling should not start before the specification has been finished. So we use again the `depends` attribute to let TaskJuggler know about this. This time we use two exclamation marks for the relative ID. The first one puts us in the scope of the enclosing `software` task. The second one is to get into the `AcSo` scope that contains the `spec` tasks. This time we allocate resources directly without using a macro.

```
task gui "Graphical User Interface" {
    effort 35d
    delayed:effort 40d
    depends !database, !backend
    allocate dev2, dev3
}
```

TaskJuggler can schedule your project for two different scenarios. We have called the first scenario "plan" scenario and the second "delayed" scenario. Many of the reports allow you to put the values of both scenarios side by side to each other, so you can compare the two scenarios. All scenario specific values that are not explicitly stated for the delayed scenario are taken from the plan scenario. So the user only has to specify the values that differ in the delayed scenario. The two scenarios must have the same task structure and the same dependencies. But the start and end dates of tasks as well as the duration may vary. In the example we have planned the work on the graphical user interface to be 35 man days. It turned

out that we actually needed 40 man days. By prefixing the start effort attribute with `delayed:` the effort value for the delayed scenario can be specified.

```
task backend "Back-End Functions" {
    effort 30d
    complete 95
    depends !database, !!spec
    allocate dev1
    allocate dev2
}
```

By default TaskJuggler assumes that all tasks are on schedule. Sometimes you want to generate reports, that show how much of a task has actually been completed. TaskJuggler uses the current date for this unless you have specified another date using the `now` attribute. If a task is ahead of schedule or late that can be specified using the `complete` attribute. This specifies how many percent of the task have been completed up to the current date. In our case the back-end implementation is slightly ahead of schedule as we will see from the report.

```
task test "Software testing" {

    task alpha "Alpha Test" {
        effort 1w
        depends !!software
        allocate test, dev2
    }

    task beta "Beta Test" {
        effort 4w
        depends !alpha
        allocate test, dev1
    }
}
```

The software testing task has been split up into an alpha and a beta test task. The interesting thing here is, that efforts can not only be specified as man days, but also man weeks, man hours, etc. Per default TaskJuggler assumes a man week is 40 man hours or 5 man days. These values can be changed using the `dailyworkinghours` attribute.

Let's go back to the outermost task again. At the beginning of the example we stated that we want to credit all development work to one account with ID `dev` and

all documentation work to the account `doc`. To achieve this, we use the attribute `account` to credit all tasks to the `dev` account.

```
task AcSo "Accounting Software" {  
  
    account dev  
  
    task software "Software Development" {
```

Since we specify the attribute for the top-level task before we declare any sub tasks, this attribute will be inherited by all sub tasks and their sub tasks and so on. Since the only exception is the writing of the manual, we need to change the account for this task again since it is also a sub task of `AcSo`.

```
    task manual "Manual" {  
        effort 10w  
        depends !deliveries.start  
        allocate doc, dev3  
        account doc  
    }
```

4.6. Specifying Milestones

All task that have been discussed so far, had a certain duration. We did not always specify the duration explicitly, but we expect them to last for a certain period of time. Sometimes you just want to capture a certain moment in your project plan. These moments are usually called milestones since they have some level of importance for the progress of the project.

TaskJuggler has support for milestones as well. They are handled as special types of tasks. By using the optional attribute `milestone` for a task, this task is declared a milestone. Milestones have no duration, so it's illegal to specify any duration criteria, or a non identical start and end date.

```
task deliveries "Milestones" {  
  
    account rev  
  
    task start "Project start" {  
        milestone  
        start 2002-01-16
```

Chapter 4. Your First Project

```
        delayed:start 2002-01-20
        startcredit 33000.0
    }

    task prev "Technology Preview" {
        milestone
        depends !!software.backend
        startcredit 13000.0
    }

    task beta "Beta version" {
        milestone
        depends !!test.alpha
        startcredit 13000.0
    }

    task done "Ship Product to customer" {
        milestone
        # maxend 2002-04-17
        depends !!test.beta, !!manual
        startcredit 14000.0
    }
}
```

We have put all important milestones of the project as sub tasks of the `deliveries` task. This way they show up nicely grouped in the reports. All milestone have either a dependency or a fixed start date. For the first milestone we have used the attribute `start` to set a fixed start date. All other tasks have direct or indirect dependencies on this task. Moving back the start date will slip the whole project. This has actually happened, so we use the `delayed:` prefix again to specify the start date for the delayed scenario.

Every milestone is linked to a customer payment. By using the `startcredit` attribute we can credit the specified amount to the account associated with this task. Since we have assigned the `rev` account to the enclosing task, all milestones will use this account as well.

Did you notice the line in the task `done` that starts with a hash? This line is commented out. If `TaskJuggler` finds a hash it ignores the rest of the line. This way you can include comments in your project. The `maxend` attribute specifies that the task should end no later than the specified date. This information is not used for scheduling but only for checking the schedule afterwards. Since the task will end later than the specified date, commenting out the line would trigger a warning.

Now the project has been completely specified. Stopping here would result in a valid TaskJuggler file that could be processed and scheduled. But no reports would be generated to visualize the results.

4.7. Generating Reports of the scheduled Project

TaskJuggler offers a number of report types. Probably the most popular ones are the HTML reports. You can advise TaskJuggler to generate one or more HTML pages that contain lists of your tasks, resources or accounts.

Before we start with the reports, we present you another macro. We like to add a navigation bar to each HTML page that holds a number of buttons. Each button changes the page to another report. This way we can create a navigation bar that holds links to all reports. Since we have created a macro, we can add the navigation bar to all reports without much hassle. The navigation bar is constructed with raw HTML tags. If you are not familiar with HTML this will look very strange to you. Don't worry, this is just a cool feature we would like to demonstrate. You can use TaskJuggler to it's full extend without having to learn HTML code.

The HTML code is injected into the reports using the rawhead attribute. This will put the HTML code close to the top of the HTML page right after the body started. As you can see here, string parameters of attributes can be enclosed in single quotes as well. This is handy, if the string itself needs to contain double quotes.

```
macro navbar [
rawhead
  '<p><center>
  <table border="2" cellpadding="10">
  <tr>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="Tasks-Overview.html">Tasks Overview</td>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="Tasks-Details.html">Tasks Details</td>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="Staff-Overview.html">Staff Overview</td>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="Staff-Details.html">Staff Details</td>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="Accounting.html">Accounting</td>
    <td class="default" style="font-size:120%" rowspan="2">
      <a href="acso.eps">GANTT Chart (Postscript)</td>
  </tr>
  </table>
```

```
</center></p><br>'
]
```

4.7.1. Generating HTML Task Reports

As the first report, we would like to have a general overview of all tasks with their computed start and end dates. To visualize the dates a bit more, we also include a Gantt chart like bar graph. The property `htmltaskreport` defines exactly this, a list of all tasks in a table. The columns are flexible and can be specified with the `column` attribute. For this report we like to see the number, the name, the start and end date, a weekly calendar and the total effort in the table.

```
htmltaskreport "Tasks-Overview.html" {
  ${navbar}
  columns hierarchindex, name, duration, effort { title "Work"},
    start, end, weekly
  timeformat "%a %Y-%m-%d"
  barlabels empty
  headline "Accounting Software Project"
  caption "This table presents a management-level overview of the
    project. The values are days or man-days."
}
```

Since we don't like the default column title for the effort column we change it to "Work". Dates should be displayed as Sun 2003-09-29, so we use the attribute `timeformat`. For the overview page we don't like to have load values in the calendar. We just want to have the Gantt chart like bars that visualize the duration of the tasks. Since the load values are reported by default, we use the attribute `barlabels` to suppress them. With the `headline` attribute we can specify a headline for the report. To have a little more info included as well, we use the `caption` attribute. Both of these attributes are followed by the string to be included into the report.

Now we like to generate a report that contains a lot more details about the task. The weekly calendar is replaced by a daily calendar. The weekly calendar had a column for each week. The daily calendar features a column for each day. The column includes the load for the task for the week or day and a colored background in case the task is active that day or week.

```
htmltaskreport "Tasks-Details.html" {
  ${navbar}
  columns no, name, start, end, scenario, daily
```

```

start 2002-03-01
end 2002-04-01
scenarios plan, delayed
headline "Accounting Software Project - March 2002"
caption "This table shows the load of each day for all the tasks.
        Additionally the resources used for each task are listed. Since the
        project start was delayed, the delayed schedule differs significantly
        from the original plan."
hideresource 0
}

```

In order to limit the report to a reasonable size, we limit the daily calendar to the interval 2002-03-01 - 2002-04-01 with the `start` and `end` attributes. We also like to list all assigned resources right after each task. Normally resources are hidden in task reports but they can be enabled by using the `hideresource` attribute. The attribute is followed by a logical expression that specifies what resources to hide. The expression is evaluated for each resource and if the result is true (not 0) than the resource is hidden. Since we want to show all resources we put a 0 in, so it's false for all resources.

To add even more information to this report, we also turn on the reporting of values of the delayed scenario by using the `scenarios` attribute. This causes TaskJuggler to split the lines of the report into one for each scenario where appropriate and report the values underneath each other.

4.7.2. Generating HTML Resource Reports

The previous report listed the resources per task. Now we want to generate a report the lists all resources. It's again a report with a weekly calendar. This time we use the attribute `loadunit` to report the load in hours instead of days.

```

htmlresourcereport "Staff-Overview.html" {
    ${navbar}
    columns no, name, scenario, weekly, effort
    scenarios plan, delayed
    loadunit hours
    headline "Weekly working hours for the Accounting Software Project"
}

```

Now a similar report but with much more details. We want to include the tasks again, this time each resource should be followed by the tasks the resource is assigned to. In `htmltaskreports` resources are hidden by default while in

htmlresourcereports tasks are hidden by default. To include tasks the attribute `hidetask` needs to be used. It is followed by a logical expression just like `hideresource`.

```
htmlresourcereport "Staff-Details.html" {
    ${navbar}
    columns name, daily, effort
    start 2002-03-01
    end 2002-04-01
    hidetask 0
    hideresource team
    sortresources nameup
    loadunit hours
    headline "Daily working hours for the Accounting Software Project -
    March 2002"
}
```

When specifying the resources we have grouped the resources into two teams by creating two pseudo resources that had the real employees as sub resources. We have attached the flag `team` to those pseudo resources. We now use this flag as logical expression for `hideresource`. So all resources that have this flag will be hidden in the report. For better readability we sort the resource list by name in ascending order. The attribute `sortresources` is taking care of this.

4.7.3. Generating HTML Account Reports

To conclude the HTML reports a report that shows how poorly the project is calculated is generated. The company won't get rich with this project. Due to the slip, it actually needs a loan from the bank to pay the salaries.

```
htmlaccountreport "Accounting.html" {
    ${navbar}
    columns no, name, total, monthly
    headline "P&L for the Accounting Software Project"
    caption "The table shows the profit and loss analysis as well as the
    cashflow situation of the Accounting Software Project."
    accumulate
    scenarios plan, delayed
}
```

The `htmlaccountreport` property produces similar reports as the above ones, but it lists accounts instead of tasks or resources. The `total` column shows the value of

the account at the end of the reported time interval. The accumulate attribute puts the calendar in accumulation mode. The monthly columns list the value of the account at the end of the month. Normally the amount that has been added or subtracted from the account would be listed.

4.7.4. Generating XML Reports

Finally we generate an XML report that contains all info about the scheduled project. This report will be used by `tjx2gantt` to create a nice GANTT chart of our project. The file can also be read by tools like `tjGUI` or the KDE Konqueror plug-in. Since the Konqueror plug-in already uses the new, version 2 XML format, you have to comment out the version attribute.

```
xmlreport "AccountingSoftware.tjx" {  
  #version 2  
}
```


Chapter 5. Language Reference

5.1. Comments

There are two ways to annotate a project file with comments. All text after a '#' will be ignored. Comments that span multiple rows must be started with '/*' and ended with '*/'.

5.2. Attribute Classes

5.2.1. DATE

A DATE is an ISO-compliant date in the format `YYYY-MM-DD[-hh:mm[:ss]][-TIMEZONE]`. Hour, minutes, seconds and the TIMEZONE are optional. If not specified, the values are set to 0. The local timezone or the default timezone is used if no other is specified. If the timezone is not known taskjuggler will fall back to UTC (GMT). The value of TIMEZONE can either be a timezone name or since this can be ambiguous, the offset to GMT as `+HHMM` or `-HHMM`. See the source code (`taskjuggler/Utility.cpp`) for details.

5.2.2. DATEINTERVAL

A date interval consists of a start and end DATE. The end date is optional. If it is missing a 24 hour interval is assumed. The start and end date must be separated by a dash character.

5.2.3. GLOBAL_ID

A GLOBAL_ID may have the same characters as ID, but additionally may contain '.' and '!'. '!' may only be used at the beginning and is used in relative IDs. A '!' means one level up.

5.2.4. ID

A string that may consist of the characters A-Z, a-z, 0-9, and _. It may not start with

a number.

5.2.5. INTEGER

A number that is an integer.

5.2.6. LOGICALEXPRESSION

This is a logical expression consisting of logical operations, such as '&' for and, '|' for or, '~' for not, '>' for greater than, '<' for less than, '=' for equal, '>=' for greater than or equal and '<=' for less than or equal to operate on INTEGER values or symbols. As symbols flag names and certain functions are supported. The expression is evaluated from left to right. '~' has a higher precedence than other operators. Use braces to avoid ambiguous operations. If flagFoo, flagFooBar, and flagBar are declared flags, the following example is a correct expression:

```
(flagFoo | flagFooBar) & ~flagBar
```

The following functions can be used in logical expressions:

isChildOf(ID)

true if the property has ID as sub.

isParentOf(ID)

true if the property has ID as enclosing property.

isLeaf(ID)

true if the property has no sub properties.

endsAfter(ID, DATE)

true if the task ends in scenario ID after the specified date

endsBefore(ID, DATE)

true if the task ends in scenario ID before the specified date

isAnAccount()

true if the property is an account

isAccount(ID)

true if the account has the listed ID

isAllocated(ID, DATE, DATE)

true if the resource has been allocated in the specified time interval in the scenario with the specified ID.

isAllocatedToProject(PRJID SCENARIOID, DATE, DATE)

true if the resource has been allocated to the specified project in the specified time interval in the scenario with the specified ID.

isMilestone()

true if the task is a milestone.

isAResource()

true if the property is a resource ID.

isResource(ID)

true if the resource has the listed ID.

isATask()

true if the property is a task.

isTask(ID)

true if the task has the listed ID.

isTaskStatus(ID, STATUS)

true if the task has in scenario ID the specified status. STATUS can be any of notstarted, inprogresslate, inprogress, ontime, inprogressearly, late, finished.

startsAfter(ID, DATE)

true if the task starts in scenario ID after the specified date

startsBefore(ID, DATE)

true if the task starts in scenario ID before the specified date

isTaskOfProject(ID)

true if the task is part of the project with the specified ID.

`isDutyOf(RESOURCE_ID, SCENARIO_ID)`

true if the resource with the specified ID is assigned to the task in the specified scenario.

`treeLevel()`

Nesting level of the property.

5.2.7. REAL

A real number (e.g., 3.14).

5.2.8. SORTINGCRITERIA

See attribute description for allowed values.

5.2.9. STRING

A string may contain any characters and is enclosed in single quotes or double quotes. A single quoted string may contain double quote characters and vice versa. A string may include line breaks.

5.2.10. TIME

A time in the format HH:MM.

5.2.11. TIME

A time interval consists of a start and end TIME. The start and end time must be separated by a dash character.

5.2.12. UNIT

May be min for minutes, h for hours, d for days, w for weeks, m for months, y for years. Week, month and year specifications are only approximated values and are handled slightly different for length, effort and duration intervals. For length and

effort only working days are counted. The number of working days per week, month or year is determined by the setting of `yearlyworkingdays`. The number of working hours or minutes per working day is determined by the setting of `dailyworkinghours`.

5.2.13. WEEKDAY

May be

`mon` for Monday
`tue` for Tuesday
`wed` for Wednesday
`thu` for Thursday
`fri` for Friday
`sat` for Saturday
`sun` for Sunday

Optional attributes of a property must be enclosed by `{ }`.

Chapter 6. Property Reference

6.1. The TJP File

The TJP File			
Description	All TaskJuggler project files should start with the project property and must contain at least one task definition. To visualize the results of the scheduling process, at least one of the reports should be specified.		
Optional Attributes	account, copyright, csvaccountreport, csvresourcereport, csvtaskreport, export, flags, htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar, include, macro, maxeffort, limits, priority, projectid, projectids, project, rate, resource, shift, supplement, task, vacation, xmlreport		
Context			
Inheritable	No	Scenario Spec.	No

6.2. account <id> <name> [<type>]

account <id> <name> [<type>]			
Description	Declares an account. Accounts can be used to calculate costs of tasks or the whole project. Account declaration may be nested, but only the top level accounts may have a type attribute specified. An account that has sub-accounts may not have a credit or a kotrusid. sub-accounts inherit this type.		
Attributes	Name	Type	Description
	id	ID	Each account must have a unique ID.
	name	STRING	

account <id> <name> [<type>]			
	type	ID	The type may be cost or revenue.
Optional Attributes	account, credit, kotrusid		
Context	The TJP File, account,		
Inheritable	No	Scenario Spec.	No
See also	csvaccountreport, htmlaccountreport		

6.3. account <accountid>

account <accountid>			
Description	All amounts associated with the task will be credited to the specified account.		
Attributes	Name	Type	Description
	accountid	ID	
Context	task,		
Inheritable	Yes	Scenario Spec.	No
See also	account		

6.4. accumulate

accumulate			
Description	If this attribute is specified the values in the calendar columns are accumulated over the reported interval.		
Context	csvaccountreport, csvresourcereport, csvtaskreport, htmlaccountreport, htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No

6.5. allowredefinition

allowredefinition			
Description	If this attribute is specified, redefinitions of task, resources or accounts are not flagged as errors. The primary use of this attribute is for projects that are created by merging sub projects which are again the result of sub project merging. In certain situations enclosing tasks, accounts or resources can be included in more than one sub project. This attribute then avoids the error which is in most other cases a real user error. In case attributes of the same property must be specified in two different locations the supplement construct is the recommended way to do this.		
Context	project,		
Inheritable	No	Scenario Spec.	No

6.6. allocate <resource>

allocate <resource>			
Description	Specify which resources should be allocated to the task.		
Attributes	Name	Type	Description
	resource	ID	
Optional Attributes	alternative, limits, mandatory, persistent, select, shift		
Context	task,		
Inheritable	Yes	Scenario Spec.	No
See also	effort		

6.7. alternative <resource> [, <resource> ...]

alternative <resource> [, <resource> ...]			
Description	A list of alternative resources for an allocation. There is no difference between the allocated resource and its alternatives. If no selection criteria is given, TaskJuggler picks the resource that it finds most appropriate.		
Attributes	Name	Type	Description
	resource	ID	
Context	allocate,		
Inheritable	No	Scenario Spec.	No
See also	select		

```
project prj "Project" "1.0" 2000-01-01 2000-03-01
```

```
resource tuxus "Tuxus"
```

```
resource tuxia "Tuxia"
```

```
task t "Task" {
    start 2000-01-01
    effort 5d
    # Use tuxus or tuxia, whoever is available.
    allocate tuxus { alternative tuxia }
}
```

6.8. barlabels <mode>

barlabels <mode>			
Description	Specifies the contense of the Gantt chart like bars in HTML calendar columns. The default is to show load values.		
Attributes	Name	Type	Description

barlabels <mode>			
	mode	ID	See table below for possible values.
Context	htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	columns, showprojectids		

empty Do not show any values on calendar bars.

load Show task or resource load on calendar bars.

6.9. booking <start> <end> <task>

booking <start> <end> <task>			
Description	<p>The booking attribute can be used to report completed work. This can be part of the necessary effort or the whole effort. When the scenario is scheduled in projection mode, TaskJuggler assumes that only the work reported with bookings has been done up to now. It then schedules a plan for the still missing effort.</p> <p>This attribute is also used within export reports to describe the details of a scheduled project.</p> <p>The sloppy attribute can be used when you want to skip non-working time or other allocations automatically. If it's not given, all bookings must only cover working time for the resource.</p>		
Attributes	Name	Type	Description
	start	DATE	
	end	DATE	
	task	ID	
Optional Attributes	sloppy		
Context	resource,		
Inheritable	No	Scenario Spec.	Yes
See also	projection		

```

project prj "Project" "1.0" 2003-06-05 2003-07-05 {
  # The baseline date for the projection.
  now 2003-06-15
  scenario plan "Plan" {
    # Compute when the task will be ready based on the already done
    # work and the current date.
    projection
  }
}

resource r1 "Resource 1"

task t1 "Task 1" {
  start 2003-06-05
  effort 10d
  allocate r1
}

supplement resource r1 {
  # This is the work that has been done up until now by r1.
  booking 2003-06-06 2003-06-07 t1 { sloppy 2 }
  booking 2003-06-08 2003-06-09 t1 { sloppy 2 }
  booking 2003-06-11 2003-06-12 t1 { sloppy 2 }
}

```

6.10. caption <text>

caption <text>			
Description	Specifies the caption used for a report table.		
Attributes	Name	Type	Description
	text	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	complete, copyright, headline		

6.11. celltext <text>

celltext <text>			
Description	Specifies an alternative text that is used for all cells of the column. Usually such a text contains a runtime macro, otherwise all cells of the column will have the same fixed value.		
Attributes	Name	Type	Description
	text	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	cellurl		

6.12. cellurl <url>

cellurl <url>			
Description	Specifies an URL that is attached to the cell contense of the cells of the column in HTML reports.		
Attributes	Name	Type	Description
	url	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	celltext		

6.13. columns <columnid> [, <columnid> ...]

columns <columnid> [, <columnid> ...]

columns <columnid> [, <columnid> ...]			
Description	Specifies which columns shall be included in a report. All columns support macro expansion. Contrary to the normal macro expansion, these macros are expanded during the report generation. So the value of the macro is being changed after each table cell or table line. Consequently only build in macros can be used. To protect the macro calls against expansion during the initial file processing, the report macros must be prefixed with an additional \$.		
Attributes	Name	Type	Description
	columnid	ID	See table below for possible values.
Optional Attributes	celltext, cellurl, hidecelltext, hidecellurl, subtitle, subtitleurl, title, titleurl		
Context	csvaccountreport, csvresourcereport, csvtaskreport, htmlaccountreport, htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No

completed	The percentage the task has been completed already. This is either the value
cost	The accumulated costs of the task and its sub tasks
criticalness	The criticalness of the task. It is a measure for the probability that an effort
cost	The accumulated costs of the task and its sub tasks
daily	A day-by-day calendar view of the tasks
depends	The task index of the tasks on which this task depends
duration	The duration of the task
efficiency	The efficiency of the resource. It's a measurement of how much the resource
effort	The effort put into the task
end	The end date of a task
endbuffer	The percentage of the endbuffer
endbufferstart	The start time of the end buffer
flags	The list of flags assigned to the task or resource
follows	The task index of the tasks that depend on this task
freeload	The workload of the resource that has not been allocated.
hierarchindex	The hierarchical index of a task. The index is calculated from the hierarchic
hierarchno	The hierarchical number of a task. It is based on the order of declaration.
id	The global ID of a task
index	The index of a task. The index is calculated from the hierachical structure o
kotrusid	The KotrRUs ID of the resource
maxeffort	The maximum daily load wanted for the resource
maxend	The latest desired end date
maxstart	The latest desired start date
mineffort	The minimum daily load wanted for the resource
minend	The earliest desired end date

minstart	The earliest desired start date
monthly	A month-by-month calendar view of the tasks
name	The name of a task, resource, or account
no	The task index in the list. It starts with 1 and increases for every listed item
note	The description of the task
pathcriticalness	The overall criticalness of the task. It is a measure for the probability that a
priority	The scheduling priority
profit	The accumulated profit of the task and its sub tasks
projectid	The project ID of the task
projectids	The project IDs of the projects a resource is allocated to
quarterly	A quarter-by-quarter calendar view of the tasks
rate	The daily rate of the resource
reference	A reference to a URL that contains further information.
resources	The names of the used resources
responsibilies	A list of all tasks indicies for which a resource is responsible
responsible	The name of the resource responsible for a task
revenue	The accumulated revenue of the task and its sub tasks
scenario	The name of the scenario. This column is helpful when multiple scenarios a
seqno	The task index in the order of declaration. Each time a task declaration is c
schedule	A detailed schedule of the allocations for the resource.
start	The start date of a task
startbuffer	The percentage of the start buffer
startbufferend	The end time of the start buffer
status	The current status of the task. This is derived from the current date or the d
total	Total accumulated values
utilization	The ratio between the allocated work load of the resource and it's overall a
weekly	A week-by-week calendar view of the tasks
yearly	A year-by-year calendar view of the tasks

The following macros are supported for normal table cells:

<code>\$\$ {0}</code>	This is the original value of the table cell. This macro is useful if the user wou
<code>\$\$ {accountid}</code>	The ID of the account.
<code>\$\$ {resourceid}</code>	The ID of the resource.
<code>\$\$ {taskid}</code>	the id of the task.

Additionally the original contense of other cells of the same report line can be accessed by the column ID. The following columns are supported:

`index, no, hierachindex, hierachno, id, name`

For the title or sub title of the calendar columns (`daily, weekly, monthly, quarterly, yearly`) the following macros are supported:

`$$ {day}, $$ {month}, $$ {week}, $$ {quarter}, $$ {year}.`

6.14. complete <percent>

complete <percent>

complete <percent>			
Description	Specifies what percentage of the task is already completed. This can be useful for project tracking. Reports with calendar elements may show the completed part of the task in a different color. Tasks may not have subtasks if this attribute is used.		
Attributes	Name	Type	Description
	percent	INTEGER	The value must be between 0 and 100.
Context	task,		
Inheritable	No	Scenario Spec.	Yes

6.15. copyright <text>

copyright <text>			
Description	Adds a copyright notice to all subsequently defined reports. This notice may contain any text. The notice is added at the bottom of the report.		
Attributes	Name	Type	Description
	text	STRING	
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	headline, caption		

6.16. credit <date> <description> <amount>

credit <date> <description> <amount>

credit <date> <description> <amount>			
Description	Credits the specified amount to the account at the specified date. The description should contain some information about the reason for the transaction.		
Attributes	Name	Type	Description
	date	DATE	
	description	STRING	
	amount	REAL	
Context	account,		
Inheritable	No	Scenario Spec.	No

6.17. csvaccountreport <filename>

csvaccountreport <filename>			
Description	The report lists all specified account values as a comma separated list. This is usefull to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.		
Attributes	Name	Type	Description
	filename	STRING	
Optional Attributes	accumulate, columns, end, rollupaccount, hideaccount, sortaccounts, start		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	htmlaccountreport		

6.18. csvresourcereport <filename>

csvresourcereport <filename>

csvresourcereport <filename>			
Description	The report lists all specified resource values as a comma separated list. This is usefull to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.		
Attributes	Name	Type	Description
	filename	STRING	
Optional Attributes	accumulate, columns, end, rollupresrouce, hideresource, sortresources, start		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	htmlresourcereport		

6.19. csvtaskreport <filename>

csvtaskreport <filename>			
Description	The report lists all specified task values as a comma separated list. This is usefull to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.		
Attributes	Name	Type	Description
	filename	STRING	
Optional Attributes	accumulate, columns, end, rolluptask, hidetask, sorttasks, start		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	htmltaskreport		

6.20. currency <text>

currency <text>

currency <text>			
Description	The default currency unit.		
Attributes	Name	Type	Description
	text	STRING	
Context	project,		
Inheritable	No	Scenario Spec.	No
See also	currencyformat		

6.21. currencyformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>

currencyformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>			
Description	These values specify the default format used for all currency values. The <code>negativeprefix</code> and <code>negativesuffix</code> strings enclose negative currency values. The <code>thousandseparator</code> can be used to make large numbers more readable. The <code>fractionseparator</code> separates the fractional part from the rest. <code>fractiondigits</code> specifies how many fractional digits should be shown at a maximum.		
Attributes	Name	Type	Description
	<code>negativeprefix</code>	STRING	
	<code>negativesuffix</code>	STRING	
	<code>thousandseparator</code>	STRING	
	<code>fractionseparator</code>	STRING	
	<code>fractiondigits</code>	INTEGER	
Context	project,		
Inheritable	Yes	Scenario Spec.	No
See also	currency		

```
project prj "Project" "1.0" 2000-01-01 2000-03-01 {
```

```
# German number format: e. g.  -10000,20 5014,11
numberformat "-" " " " " ", " 2

# US currency format: e. g. (10,000.20) 5,014.11
currencyformat "(" " ")" " ", " "." 2
}

task t "Task" {
  start 2000-01-01
  milestone
}
```

6.22. dailymax <value> <unit>

dailymax <value> <unit>			
Description	Sets the daily limit of a resource usage or a resource allocation to a task.		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	limits,		
Inheritable	Yes	Scenario Spec.	No
See also	monthlymax, weeklymax		

6.23. dailyworkinghours <hours>

dailyworkinghours <hours>

dailyworkinghours <hours>			
Description	Set the average number of working hours per day. This is used as the base to convert working hours into working days. This affects for example the length task attribute. The default value is 8 hours and should work for most Western countries. The value you specify should match the settings you specified for workinghours.		
Attributes	Name	Type	Description
	hours	REAL	
Context	project,		
Inheritable	No	Scenario Spec.	No
See also	yearlyworkingdays		

6.24. depends <task> [, <task> ...]

depends <task> [, <task> ...]			
Description	<p>Specifies that the task cannot start before the task with the specified IDs have been finished. If multiple IDs are specified, they must be separated by commas. IDs must be either global or relative. A relative ID starts with a number of '!'. Each '!' moves the scope to the parent task. Global IDs do not contain '!', but have IDs separated by dots. Each task ID can have optional attributes enclosed in braces.</p> <p>By using the 'depends' attribute, the scheduling policy is automatically set to <i>asap</i>. If 'depends' and 'precedes' are used, the last policy counts.</p>		
Attributes	Name	Type	Description
	task	ID	
Optional Attributes	gapduration, gaplength		
Context	task,		
Inheritable	No	Scenario Spec.	No

depends <task> [, <task> ...]	
See also	complete, precedes

```
project p "P" "1.0" 2003-11-09 2003-12-24

task foo1 "foo1" {
  task foo2 "foo2" {
    start 2003-12-04
    milestone
  }
  task foo3 "foo3" {
    depends !foo2
    length 1d
  }
}
task bar "bar" {
  depends foo1.foo2
  length 2d
}

task bar1 "bar1" {
  depends foo1 { gapduration 2d }, bar { gaplength 1d }
  duration 2d
}
```

6.25. disabled

disabled			
Description	Disables the scenario for scheduling.		
Context	scenario,		
Inheritable	No	Scenario Spec.	No

6.26. duration <value> <unit>

duration <value> <unit>			
Description	<p>Specifies the time the task occupies the resources. This is calendar time, not working time. 7d means one week. If resources are specified they are allocated when available. Availability of resources has no impact on the duration of the task. It will always be the specified duration. Tasks may not have subtasks if this attribute is used.</p>		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	effort, length		

6.27. efficiency <value>

efficiency <value>			
Description	<p>The efficiency of a resource can be used for two purposes. First you can use it as a crude way to model a team. A team of 5 people should have an efficiency of 5.0. Keep in mind that you cannot track the member of the team individually if you use this feature. The other use is to model performance variations between your resources. All resources that do not contribute effort to the task, should have an efficiency of 0.0.</p>		
Attributes	Name	Type	Description
	value	REAL	
Context	resource,		
Inheritable	Yes	Scenario Spec.	No

efficiency <value>	
See also	load

6.28. effort <value> <unit>

effort <value> <unit>			
Description	<p>Specifies the effort needed to complete the task. An effort of 4d can be done with 2 full-time resources in 2 days. The task will not finish before the resources have contributed the specified effort. So the duration of the task will depend on the availability of the resources.</p> <p>WARNING: In almost all real world projects effort is not the product of time and resources. This is only true if the task can be partitioned without adding any overhead. For more information about this read "The Mythical Man-Month" by Frederick P. Brooks, Jr.</p> <p>Tasks may not have subtasks if this attribute is used.</p>		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	duration, length		

6.29. end <date>

end <date>	
Description	Specifies the end date of the report. In task reports only tasks that start before this end date are listed.

end <date>			
Attributes	Name	Type	Description
	date	DATE	
Context	csvaccountreport, csvresourcereport, csvtaskreport, htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	start		

6.30. end <date>

end <date>			
Description	The end date of the task		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	start, maxend, minend, endbuffer		

6.31. endbuffer <percent>

endbuffer <percent>			
Description	Specifies how much slack time you expect to have at the end of the task.		
Attributes	Name	Type	Description
	percent	REAL	Percent slack of the overall effort, duration or length of the task.

endbuffer <percent>			
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	duration, effort, length		

6.32. endcredit <amount>

endcredit <amount>			
Description	Specifies an amount that is credited to the account specified by the account property at the moment the tasks ends.		
Attributes	Name	Type	Description
	amount	REAL	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	startcredit		

6.33. export <filename>

export <filename>

export <filename>			
Description	<p>The export report looks like a regular taskjuggler file but contains fixed start and end dates for all tasks. The tasks only have start and end times, their description and their project id listed. No other attributes are exported unless they are requested using the taskattributes attribute. The contense also depends on the extension of the file name. If the file name ends with .tjp a complete project with header, resource and shift definitions is generated. In case it ends with .tji only the tasks and resource allocations are exported.</p> <p>If specified the resource usage for the tasks is reported as well. But only those allocations are listed that belong to tasks listed in the same export report.</p> <p>The export report can be used to share certain tasks or milestones with other projects. When an export report is included the project IDs of the included tasks must be declared first with the project id property.</p>		
Attributes	Name	Type	Description
	filename	STRING	
Optional Attributes	hideresource, hidetask, rollupresrouce, rolluptask, scenarios, taskattributes, taskroot		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	include		

6.34. extend <property>

extend <property>

extend <property>			
Description	Often it is desirable to collect more information in the project file than is necessary for task scheduling and resource allocation. To add such information to tasks, resources or accounts the user can extend these properties with user-defined attributes. The new attributes can be <code>text</code> or <code>reference</code> attributes. Optionally the user can specify if the attribute value should be inherited from the enclosing property.		
Attributes	Name	Type	Description
	property	ID	Possible values are <code>task</code> , <code>resource</code> or <code>account</code>
Context	project,		
Inheritable	No	Scenario Spec.	No

<type> <id> <name>

type: Specifies the type of the user-defined attribute. `text` is a simple text attribute. `reference`

id: An user-defined ID that is unique within the used-defined task attributes. To avoid conflicts

name: A short description of the attribute. It will be used as default column header in reports.

```
project ca "Custom Attributes" "$Id: CustomAttributes.tjp,v 1.4 2005/02/23
  extend task {
    reference MyLink "My Link"
    text MyText "My Text"
  }
}
```

```
task t "Task" {
  start 2003-05-28
  milestone
  MyLink "http://www.taskjuggler.org" { label "TJ Web" }
  MyText "TaskJuggler is great!"
}
```

6.35. flags <flag> [, <flag> ...]

flags <flag> [, <flag> ...]			
Description	Attach a set of flags. The flags can be used in logical expressions to filter properties from the reports.		
Attributes	Name	Type	Description
	flag	ID	
Context	resource, task,		
Inheritable	Yes	Scenario Spec.	No
See also	flags		

6.36. flags <flag> [, <flag> ...]

flags <flag> [, <flag> ...]			
Description	Declare a set of flags for later use.		
Attributes	Name	Type	Description
	flag	ID	
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	flags		

6.37. gapduration <value> <unit>

gapduration <value> <unit>	
Description	Specifies the minimum required gap between the end of a preceding task and the start of this task, or the start of a following task and the end of this task. This is calendar time, not working time. 7d means one week.

gapduration <value> <unit>			
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	depends, precedes,		
Inheritable	Yes	Scenario Spec.	Yes
See also	duration, gaplength		

6.38. gaplength <value> <unit>

gaplength <value> <unit>			
Description	Specifies the minimum required gap between the end of a preceding task and the start of this task, or the start of a following task and the end of this task. This is working time, not calender time. 7d means 7 working days, not one week. Whether a day is considered a working day or not depends on the defined working hours and global vacations.		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	depends, precedes,		
Inheritable	Yes	Scenario Spec.	Yes
See also	gapduration, length		

6.39. headline <text>

headline <text>			
Description	Specifies the headline for a report.		
Attributes	Name	Type	Description

headline <text>			
	text	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	caption, copyright		

6.40. hideaccount <logicalexpression>

hideaccount <logicalexpression>			
Description	Do not show accounts that match the specified logical expression. If the report is sorted in tree mode (default) then enclosing accounts are listed even if the expression matches the account.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvaccountreport, htmlaccountreport,		
Inheritable	No	Scenario Spec.	No

6.41. hidecelltext <expression>

hidecelltext <expression>			
Description	If the expression is true, the cell will be empty.		
Attributes	Name	Type	Description
	expression	LOGICALEXPRESSION	
Context	columns,		
Inheritable	No	Scenario Spec.	No

hidecelltext <expression>	
See also	hidecellurl

6.42. hidecellurl <expression>

hidecellurl <expression>			
Description	If the expression is true, no URL will be attached to the cell contense.		
Attributes	Name	Type	Description
	expression	LOGICALEXPRESSION	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	hidecelltext		

6.43. hideresource <logicalexpression>

hideresource <logicalexpression>			
Description	Do not show resources that match the specified logical expression. If the report is sorted in tree mode (default) then enclosing resources are listed even if the expression matches the resource.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvresourcereport, export, htmlresourcereport, htmltaskreport, htmlweeklycalendar, xmlreport,		
Inheritable	No	Scenario Spec.	No

6.44. hidetask <logicalexpression>

hidetask <logicalexpression>			
Description	Do not show tasks that match the specified logical expression. If the report is sorted in tree mode (default) then enclosing tasks are listed even if the expression matches the task.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvtaskreport, export, htmlresourcereport, htmltaskreport, htmlweeklycalendar, xmlreport,		
Inheritable	No	Scenario Spec.	No

6.45. htmlaccountreport

htmlaccountreport			
Description	The report lists all specified account values as a HTML page.		
Attributes	Name	Type	Description
Optional Attributes	accumulate, caption, columns, end, headline, hideaccount, rawhead, rawstylesheet, rawtail, rollupaccount, scenarios, shorttimeformat, sortaccounts, start, timeformat		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	csvaccountreport, htmlresourcereport, htmltaskreport		

6.46. htmlresourcereport

htmlresourcereport

htmlresourcereport			
Description	The report lists all resources and their respective values as a HTML page. The tasks that the resources are allocated to can be listed as well.		
Attributes	Name	Type	Description
Optional Attributes	accumulate, barlabels, caption, columns, end, headline, hideresource, hidetask, loadunit, rawhead, rawstylesheet, rawtail, rollupresrouce, rolluptask, scenarios, shorttimeformat, showprojectids, sortresources, sorttasks, start, timeformat		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	csvresourcereport, htmlaccountreport, htmltaskreport		

6.47. htmlstatusreport <file>

htmlstatusreport <file>			
Description	Generates a HTML status report. The report consits of 4 tables: Overdue tasks, ongoing tasks, finished tasks and upcoming tasks. The default reporting interval is 1 week.		
Attributes	Name	Type	Description
	file	STRING	
Optional Attributes	scenarios, start, end, headline, caption, rawhead, rawstylesheet, rawtail, loadunit, timeformat, shorttimeformat		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	htmlaccountreport, htmlresourcereport, htmltaskreport, htmlweeklycalendar		

6.48. htmltaskreport

htmltaskreport			
Description	The report lists all tasks and their respective values as a HTML page. The resources that are allocated to the tasks can be listed as well.		
Attributes	Name	Type	Description
Optional Attributes	accumulate, barlabels, caption, columns, end, headline, hideresource, hidetask, loadunit, rawhead, rawstylesheet, rawtail, rollupresrouce, rolluptask, scenarios, shorttimeformat, showprojectids, sortresources, sorttasks, start, timeformat		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	csvtaskreport, htmlaccountreport, htmlresourcereport		

6.49. htmlweeklycalendar <file>

htmlweeklycalendar <file>			
Description	Generates a calendar like HTML report.		
Attributes	Name	Type	Description
	file	STRING	
Optional Attributes	accumulate, barlabels, caption, columns, end, headline, hideresource, hidetask, loadunit, rawhead, rawstylesheet, rawtail, rollupresrouce, rolluptask, scenarios, shorttimeformat, showprojectids, sortresources, sorttasks, start, timeformat		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	htmlaccountreport, htmlresourcereport, htmltaskreport, htmlstatusreport		

6.50. include <file>

include <file>			
Description	Includes the specified file name as if its contents would be written instead of the include property. <code>include</code> commands can be used within global scope or between property declarations of tasks, resources, and accounts. For technical reasons you have to supply the optional pair of curly brackets if the include is followed immediately by a macro that is defined within the included file.		
Attributes	Name	Type	Description
	file	STRING	
Optional Attributes	taskprefix		
Context	The TJP File, project,		
Inheritable	No	Scenario Spec.	No
See also	export		

6.51. journalentry <date> <text>

journalentry <date> <text>			
Description	Journal entries are meant for documentation purposes. They consist of a date and a text entry. Each journal entry adds a new entry to the journal of the property.		
Attributes	Name	Type	Description
	date	DATE	
	text	STRING	
Context	project, resource, task,		
Inheritable	No	Scenario Spec.	No

```
project journal "Project" "$Id" 2000-01-01 2000-01-04 {
```

```

    journalentry 2000-01-02 "The project started."
    journalentry 2000-01-03 "We made some progress."
}

resource tux "Tux" {
    journalentry 2000-01-02 "This guy is a bummer."
}

task t1 "Task1" {
    journalentry 2000-01-01 "Probably will be done sooner."
    journalentry 2000-01-03 "Maybe not."
    start 2000-01-01
    milestone
}

```

6.52. kotrusid <id>

kotrusid <id>			
Description	The KoTrus ID of the account (cost object). This is a special reserved keyword.		
Attributes	Name	Type	Description
	id	STRING	
Context	account, resource,		
Inheritable	Yes	Scenario Spec.	No

6.53. label <text>

label <text>			
Description	Specifies the text for the URL in HTML reports.		
Attributes	Name	Type	Description
	text	STRING	

label <text>			
Context	reference,		
Inheritable	No	Scenario Spec.	No

6.54. length <value> <unit>

length <value> <unit>			
Description	Specifies the time the task occupies the resources. This is working time, not calender time. 7d means 7 working days, not one week. Whether a day is considered a working day or not depends on the defined working hours and global vacations. A task with a length specification may have resource allocations. Resources are allocated when they are available. The availability has no impact on the duration of the task. A day where none of the specified resources is available is still considered a working day, if there is no global vacation of or global working time defined. Tasks may not have subtasks if this attribute is used.		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	duration, effort		

6.55. limits

limits

limits			
Description	<p>Specifies limits on the usage of a resource in general, or of the allocation of a resource to a task. This property replaces the less flexible properties maxeffort and load.</p> <p>When applied to an allocation this limits the use of all alternative resources or group members as a whole. There has been a bug in version 2.0.x that resulted in faulty limit computation. This has been fixed with version 2.1.</p>		
Optional Attributes	dailymax, weeklymax, monthlymax		
Context	The TJP File, allocate, resource,		
Inheritable	Yes	Scenario Spec.	No

```

project limits "Limits" "1.0" 2004-03-01 2004-05-01

# Default limit that affects all subsequently defined resources
limits {
    weeklymax 4d
}

resource r1 "R1" {
    limits { dailymax 2h weeklymax 6h monthlymax 2.5d }
}

resource r2 "R2"

task t1 "Task 1" {
    start 2004-03-01
    duration 60d
    # allocation is subject to resource limits
    allocate r1
}

task t2 "Task 2" {
    start 2004-03-01
    duration 60d
    # limits can also be specified per allocation
    allocate r2 {
        limits { dailymax 4h weeklymax 3d monthlymax 2w }
    }
}

```

6.56. load <factor>

load <factor>			
Description	<p>This property has been replaced by limits. The further usage of load is strongly discouraged. It will be dropped from future versions of TaskJuggler.</p> <p>Specifies the daily load of a resource for an allocation. A load of 1.0 (default) means the resource is allocated for as many hours as specified by dailyworkinghours. A load of 0.5 means half that many hours. This only works if enough working hours have been specified for the particular day.</p>		
Attributes	Name	Type	Description
	factor	REAL	
Context			
Inheritable	No	Scenario Spec.	No
See also	workinghours, vacation		

6.57. loadunit <unit>

loadunit <unit>			
Description	Specifies the unit in which loads are reported in a report.		
Attributes	Name	Type	Description
	unit	ID	See table below for possible values.
Context	htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	dailyworkinghours, yearlyworkingdays		

days	Show load in man or resource-days.
hours	Show load in man resource-hours.
longauto	Show load in the most appropriate unit and show long unit name.
minutes	Show load in man or resource-minutes.
months	Show load in man or resource-months.
shortauto	Show load in the most appropriate unit and show short unit name.
weeks	Show load in man or resource-weeks.
years	Show load in man or resource-years.

6.58. macro <id>

macro <id>			
Description	Defines a text fragment that can later be inserted by using the specified ID.		
Attributes	Name	Type	Description
	id	ID	
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No

The body is not optional. It must be enclosed in []. Macros can be declared like this:

```
macro FOO [ This text ];
```

If later `${FOO}` is found in the project file, it is expanded to ' This text '. Macros may have arguments. Arguments are special macros with numbers as names. The number specifies the index of the argument.

```
macro FOO [ This ${1} text ]
```

will expand to ' This stupid text ' if called as `${FOO "stupid"}`. Macros may call other macros.

Macro IDs should have at least one uppercase letter as all lowercase letter IDs may be used in a later version for built-in macros like 'if', 'expr' or 'for'. Macro names can be prefixed by a questionmark. In this case the macro will expand to nothing if

the macro is not defined. Otherwise the undefined macro would be flagged with an error message.

This macro call `${?foo}` will expand to nothing if `foo` is undefined.

6.59. mandatory

mandatory			
Description	Makes a resource allocation mandatory. This means, that for each time slot only then resources are allocated when all mandatory resources are available. So either all mandatory resources can be allocated for the time slot, or no resource will be allocated.		
Context	allocate,		
Inheritable	No	Scenario Spec.	No

```
project prj "Project" "1.0" 2000-01-01 2000-03-01

resource tuxus "Tuxus"
resource truck "Truck" {
    # Truck does not do any work!
    efficiency 0.0
}

task t "Ship stones to customers" {
    start 2000-01-01
    effort 5d
    # We need the truck to deliver the stones, so only allocate
    # tuxus when the truck is available.
    allocate tuxus
    allocate truck { mandatory }
}
```


6.60. maxeffort <workingdays>

maxeffort <workingdays>			
Description	<p>This property has been replaced by limits. The further usage of maxeffort is strongly discouraged. It will be dropped from future versions of TaskJuggler.</p> <p>The daily maximum effort for a resource. Resources will not be scheduled to be used more than this value. A value of 1.0 means a full working day. 0.5 means half a working day.</p>		
Attributes	Name	Type	Description
	workingdays	REAL	
Context	The TJP File, resource,		
Inheritable	No	Scenario Spec.	No
See also	dailyworkinghours, workinghours		

6.61. maxend <date>

maxend <date>			
Description	<p>Specifies the maximum wanted end time of the task. The value is not used during scheduling, but is checked after all tasks have been scheduled. If the end of the task is later than the specified value, then an error is reported.</p>		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	maxstart, minend, minstart		

6.62. maxstart <date>

maxstart <date>			
Description	Specifies the maximum wanted start time of the task. The value is not used during scheduling, but is checked after all tasks have been scheduled. If the start of the task is later than the specified value, then an error is reported.		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	maxend, minend, minstart		

6.63. minend <date>

minend <date>			
Description	Specifies the minimum wanted end time of the task. The value is not used during scheduling, but is checked after all tasks have been scheduled. If the end of the task is earlier than the specified value, then an error is reported.		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	maxend, maxstart, minstart		

6.64. minstart <date>

minstart <date>

minstart <date>			
Description	Specifies the minimum wanted start time of the task. The value is not used during scheduling, but is checked after all tasks have been scheduled. If the start of the task is earlier than the specified value, then an error is reported.		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	maxend, maxstart, minend		

6.65. milestone

milestone			
Description	<p>Turns the task into a special task that has no duration. You may not specify a duration, length, effort or subtasks for a milestone task.</p> <p>A task that only has a start or an end specification and no duration specification or sub tasks, will be recognized as milestone automatically.</p>		
Context	task,		
Inheritable	No	Scenario Spec.	No

6.66. note <text>

note <text>			
Description	Attach a note to the task. This is usually a more detailed specification of what the task is about.		
Attributes	Name	Type	Description

note <text>			
	text	STRING	
Context	task,		
Inheritable	No	Scenario Spec.	No
See also	statusnote		

6.67. monthlymax <value> <unit>

monthlymax <value> <unit>			
Description	Sets the monthly limit of a resource usage or a resource allocation to a task.		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	limits,		
Inheritable	Yes	Scenario Spec.	No
See also	dailymax, weeklymax		

6.68. now <date>

now <date>			
Description	Specify the date that TaskJuggler uses for calculation as current date. If no value is specified, the current value of the system clock is used.		
Attributes	Name	Type	Description
	date	DATE	
Context	project,		
Inheritable	Yes	Scenario Spec.	No

6.69. numberformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>

numberformat <negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>			
Description	These values specify the default format used for all numerical real values. The <code>negativeprefix</code> and <code>negativesuffix</code> strings enclose negative currency values. The <code>thousandseparator</code> can be used to make large numbers more readable. The <code>fractionseparator</code> separates the fractional part from the rest. <code>fractiondigits</code> specifies how many fractional digits should be shown at a maximum.		
Attributes	Name	Type	Description
	<code>negativeprefix</code>	STRING	
	<code>negativesuffix</code>	STRING	
	<code>thousandseparator</code>	STRING	
	<code>fractionseparator</code>	STRING	
	<code>fractiondigits</code>	INTEGER	
Context	project,		
Inheritable	Yes	Scenario Spec.	No

```

project prj "Project" "1.0" 2000-01-01 2000-03-01 {
  # German number format: e. g.  -10000,20 5014,11
  numberformat "-" "" "" " ", " 2

  # US currency format: e. g. (10,000.20) 5,014.11
  currencyformat "(" " ")" " ", " "." 2
}

task t "Task" {
  start 2000-01-01
  milestone
}

```

6.70. persistent

persistent			
Description	Specifies that once a resource is picked from the list of alternatives this resource is used for the whole task. This is usefull when several alternative resources have been specified. Normaly the selected resource can change after each break. A break is an interval of at least one timeslot where no resources were available.		
Context	allocate,		
Inheritable	No	Scenario Spec.	No
See also	alternative		

```

project prj "Project" "1.0" 2003-06-05 2003-07-05

resource r1 "Resource 1"
resource r2 "Resource 2"

task t1 "Task 1" {
    start 2003-06-05
    effort 5d
    # Pick one of them and use it for the entire task
    allocate r1 { alternative r2 persistent }
}

```

6.71. priority <value>

priority <value>			
Description	Specifies a priority between 1 and 1000. A task with higher priority is more likely to get the requested resources. This attribute is inherited by subtasks if specified prior to the definition of the subtask.		
Attributes	Name	Type	Description

priority <value>			
	value	INTEGER	
Context	The TJP File, task,		
Inheritable	No	Scenario Spec.	No

6.72. precedes <task> [, <task> ...]

precedes <task> [, <task> ...]			
Description	Specifies that the tasks with the specified IDs cannot start before the task has been finished. If multiple IDs are specified, they must be separated by commas. IDs must be either global or relative. A relative ID starts with a number of '!'. Each '!' moves the scope to the parent task. Global IDs do not contain '!', but have IDs separated by dots. By using the 'precedes' attribute, the scheduling policy is automatically set to <code>alap</code> . If both 'precedes' and 'precedes' are used within a task, the last policy counts.		
Attributes	Name	Type	Description
	task	ID	
Optional Attributes	gapduration, gaplength		
Context	task,		
Inheritable	No	Scenario Spec.	No
See also	complete, depends		

```
project p "P" "1.0" 2003-11-09 2003-12-24
```

```
task foo1 "foo1" {
  task foo2 "foo2" {
    start 2003-12-04
    milestone
  }
  task foo3 "foo3" {
    precedes !foo2
  }
}
```

```

        length 1d
    }
}
task bar "bar" {
    precedes foo1.foo2
    length 2d
}

```

6.73. project <id> <name> <version> <start> <end>

project <id> <name> <version> <start> <end>			
Description	The project property is mandatory and should be the first property in a project file. <id> is the default project ID used to register resource allocations in a global database. <name> is the name of the project. <version> is the version of the project file. Typically this is the CVS ID. <start> and <end> define the time frame of the project. The end may be well after the end of the last task, but must be specified to terminate the scheduling process.		
Attributes	Name	Type	Description
	id	ID	The default project ID.
	name	STRING	The name of the project.
	version	STRING	The version of the project file. This could be a revision number from a revision control system.
	start	DATE	All task must start after this date.
	end	DATE	All task must end before this date.

project <id> <name> <version> <start> <end>			
Optional Attributes	allowredefinition, currencyformat, currency, dailyworkinghours, extend, include, journalentry, now, numberformat, scenario, shorttimeformat, timeformat, timezone, timingresolution, weekstartsmunday, weekstartssunday, workinghours, yearlyworkingdays		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No

6.74. projectid <id>

projectid <id>			
Description	At global scope it declares a new project id and activates it. All subsequent task definitions will inherit this ID. If used within a task it simply assigns this project ID to the task. The tasks of a project can have different IDs. This is particularly helpful if the project is merged from several sub projects that each have their own ID.		
Attributes	Name	Type	Description
	id	ID	
Context	The TJP File, task,		
Inheritable	Yes	Scenario Spec.	No
See also	project, projectids		

6.75. projectids <projectid> [, <projectid> ...]

projectids <projectid> [, <projectid> ...]			
Description	Declares a list of project IDs.		
Attributes	Name	Type	Description
	projectid	ID	

projectids <projectid> [, <projectid> ...]			
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No
See also	project, projectid		

6.76. projection

projection			
Description	Enables the projection mode for the scenario. All tasks will be scheduled taking the manual bookings into account. The tasks will be extended after the last manual bookings until the specified effort, length or duration has been reached.		
Context	scenario,		
Inheritable	No	Scenario Spec.	No
See also	booking		

6.77. rate <value>

rate <value>			
Description	Specifies the daily costs of the resource. The amount are credited to the account specified with the task that makes use of the resource.		
Attributes	Name	Type	Description
	value	REAL	
Context	The TJP File, resource,		
Inheritable	Yes	Scenario Spec.	No
See also	account, task		

6.78. rawhead <html>

rawhead <html>			
Description	Specifies a section of raw HTML code that will be inserted at the top of the report.		
Attributes	Name	Type	Description
	html	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	rawstylesheet, rawtail		

6.79. rawstylesheet <stylesheet>

rawstylesheet <stylesheet>			
Description	Specifies a stylesheet for HTML reports.		
Attributes	Name	Type	Description
	stylesheet	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No

6.80. rawtail <html>

rawtail <html>			
Description	Specifies a section of raw HTML code that will be inserted at the bottom of the report.		
Attributes	Name	Type	Description

rawtail <html>			
	html	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	rawhead, rawstylesheet		

6.81. reference <url>

reference <url>			
Description	A reference to an external document. If you need more than one reference, you can create your own URL placeholders.		
Attributes	Name	Type	Description
	url	STRING	Should be a well formed URL.
Optional Attributes	label		
Context	task,		
Inheritable	No	Scenario Spec.	No
See also	extend		

6.82. resource <id> <name>

resource <id> <name>			
Description	Task use resources to fullfill the specified efforts.		
Attributes	Name	Type	Description
	id	ID	
	name	STRING	
Optional Attributes	booking, efficiency, flags, journalentry, kotrusid, maxeffort, limits, rate, resource, shift, vacation, workinghours		

resource <id> <name>			
Context	The TJP File, resource,		
Inheritable	No	Scenario Spec.	No
See also	task		

6.83. responsible <resource>

responsible <resource>			
Description	The ID of the resource that is responsible for this task.		
Attributes	Name	Type	Description
	resource	ID	
Context	task,		
Inheritable	Yes	Scenario Spec.	No
See also	resource		

6.84. rollupaccount <logicalexpression>

rollupaccount <logicalexpression>			
Description	Do not show sub-accounts of accounts that match the specified logical expression.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvaccountreport, htmlaccountreport,		
Inheritable	No	Scenario Spec.	No
See also	rollupresrouce, rolluptask		

6.85. rollupresrouce <logicalexpression>

rollupresrouce <logicalexpression>			
Description	Do not show sub-resources of resources that match the specified logical expression.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvresourcereport, export, htmlresourcereport, htmltaskreport, htmlweeklycalendar, xmlreport,		
Inheritable	No	Scenario Spec.	No
See also	rollupaccount, rolluptask		

6.86. rolluptask <logicalexpression>

rolluptask <logicalexpression>			
Description	Do not show sub-tasks of tasks that match the specified logical expression.		
Attributes	Name	Type	Description
	logicalexpression	LOGICALEXPRESSION	
Context	csvtaskreport, export, htmlresourcereport, htmltaskreport, htmlweeklycalendar, xmlreport,		
Inheritable	No	Scenario Spec.	No
See also	rollupaccount, rollupresrouce		

6.87. scenario <id> <name>

scenario <id> <name>			
Description	Specifies the different project scenarios. A scenario that is nested into another one inherits all values from the enclosing scenario except those values that were specified specifically for this scenario.		
Attributes	Name	Type	Description
	id	ID	
	name	STRING	
Optional Attributes	disabled, projection, scenario		
Context	project, scenario,		
Inheritable	No	Scenario Spec.	No
See also	scenarios		

6.88. scenarios <scenarioid> [, <scenarioid> ...]

scenarios <scenarioid> [, <scenarioid> ...]			
Description	List of scenarios that should be included in the report.		
Attributes	Name	Type	Description
	scenarioid	ID	The ID of the scenario.
Context	export, htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar, xmlreport,		
Inheritable	No	Scenario Spec.	No
See also	scenario		

6.89. scheduled

scheduled			
Description	This is mostly for internal use. It specifies that the task can be ignored for scheduling in the scenario.		
Context	task,		
Inheritable	No	Scenario Spec.	Yes

6.90. scheduling <type>

scheduling <type>			
Description	Specifies the scheduling policy for the task. A task can be scheduled from start to end (As Soon As Possible, <i>asap</i>) or from end to start (As Late As Possible, <i>alap</i>).		
Attributes	Name	Type	Description
	type	ID	Possible values are <i>alap</i> or <i>asap</i> .
Context	task,		
Inheritable	Yes	Scenario Spec.	No

6.91. select <mode>

select <mode>

select <mode>			
Description	<p>The select functions controls which resource is picked from an allocation and it's alternatives. The selection is re-evaluated each time the resource used in the previous time slot becomes unavailable.</p> <p>Even for non-persistent allocations a change in the resource selection only happens if the resource used in the previous (or next for ASAP tasks) time slot has become unavailable.</p>		
Attributes	Name	Type	Description
	mode	ID	See table below for possible values.
Context	allocate,		
Inheritable	No	Scenario Spec.	No
See also	persistent		

maxloaded
minloaded
minallocated
order
random

Pick the available resource that has been used the most so far.
Pick the available resource that has been used the least so far.
Pick the resource that has the smallest allocation factor. The allocation factor is c
Pick the first available resource from the list.
Pick a random resource from the list.

```

project prj "Project" "1.0" 2000-01-01 2000-03-01

resource tuxus "Tuxus"
resource tuxia "Tuxia"

task t1 "Task 1" {
    start 2000-01-01
    effort 5d
    # First try to allocate Tuxus. When he is not available try Tuxia.
    allocate tuxus { alternative tuxia select order }
}

task t2 "Task 2" {
    start 2000-01-01
    effort 5d
    # Use tuxus or tuxia, whoever is available and try to balance
    # the allocated load.
    allocate tuxus { alternative tuxia select minloaded}
}

task t3 "Task 3" {

```

```
start 2000-01-01
effort 5d
# For slave drivers: Always pick the resource that has been loaded
# the most already.
allocate tuxus { alternative tuxia select maxloaded}
}
```

6.92. sloppy <value>

sloppy <value>			
Description	Controls how strict TaskJuggler checks booking intervals for conflicts with vacation and other bookings. In case the error is suppressed the booking will not overwrite the existing bookings. It will avoid the already assigned intervals during booking.		
Attributes	Name	Type	Description
	value	INTEGER	Number between 0 and 3. See table below.
Context	booking,		
Inheritable	No	Scenario Spec.	No

sloppy 0: Period may not contain any off-duty hours, vacation or other task assignments.
sloppy 1: Period may contain off-duty hours, but not vacation or other task assignments.
sloppy 2: Period may contain off-duty hours and vacation, but no other task assignments.
sloppy 3: Period may overlap with off-duty hours, vacation or other task assignments. These are

6.93. shift <id> <name>

shift <id> <name>

shift <id> <name>			
Description	When several resource have the same working hours, these working hours should be defined as shifts. Each shift must have a unique ID. Resources can be assigned to shifts for certain intervals.		
Attributes	Name	Type	Description
	id	ID	
	name	STRING	
Optional Attributes	shift, workinghours		
Context	The TJP File, shift,		
Inheritable	No	Scenario Spec.	No
See also	shift		

6.94. shift <shiftid> [<dateinterval>]

shift <shiftid> [<dateinterval>]			
Description	Limits the resource working time or work on a task to a defined shift during the specified interval. Multiple shifts can be defined, but shift intervals may not overlap.		
Attributes	Name	Type	Description
	shiftid	ID	The ID of the selected shift.

shift <shiftid> [<dateinterval>]			
	dateinterval	DATEINTERVAL	If an interval is specified, no allocations will be made outside the shift intervals unless other shifts have been selected for other time intervals. If the interval is omitted, the shift is assigned for the whole project time frame.
Context	allocate, resource, task,		
Inheritable	No	Scenario Spec.	No
See also	shift		

6.95. shorttimeformat <format>

shorttimeformat <format>			
Description	Specifies time format for time short specifications. This is normal just the hour and minutes.		
Attributes	Name	Type	Description
	format	STRING	
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar, project,		
Inheritable	Yes	Scenario Spec.	No
See also	timeformat		

6.96. showprojectids

showprojectids			
Description	Specifies that calendar columns in reports should contain the project ID after the load value.		
Context	htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	barlabels, columns		

6.97. sortaccounts <criteria> [, <criteria> ...]

sortaccounts <criteria> [, <criteria> ...]			
Description	Determines how the accounts are sorted in the report. Up to 3 criteria can be specified. If one criteria is not sufficient to sort a group of accounts, the next criteria will be used to sort the accounts within this group.		
Attributes	Name	Type	Description
	criteria	SORTINGCRITERIA	Possible values are fullnamedown, fullnameup, iddown, idup, indexdown, indexup, namedown, nameup, sequencedown, sequenceup, tree
Context	csvaccountreport, htmlaccountreport,		
Inheritable	No	Scenario Spec.	No

6.98. sortresources <criteria> [, <criteria> ...]

sortresources <criteria> [, <criteria> ...]			
Description	Determines how the resources are sorted in the report. Up to 3 criteria can be specified. If one criteria is not sufficient to sort a group of accounts, the next criteria will be used to sort the accounts within this group.		
Attributes	Name	Type	Description
	criteria	SORTINGCRITERIA	Possible values are fullnamedown, fullnameup, iddown, idup, indexdown, indexup, maxeffortdown, maxeffortup, mineffortdown, mineffortup, namedown, nameup, ratedown, rateup, sequencedown, sequenceup, tree
Context	csvresourcereport, htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No

6.99. sorttasks <criteria> [, <criteria> ...]

sorttasks <criteria> [, <criteria> ...]	
Description	Determines how the tasks are sorted in the report. Up to 3 criteria can be specified. If one criteria is not sufficient to sort a group of accounts, the next criteria will be used to sort the accounts within this group.

sorttasks <criteria> [, <criteria> ...]			
Attributes	Name	Type	Description
	criteria	SORTINGCRITERIA	Possible values are completeddown, completedup, criticalnessdown, criticalnessup, enddown, endup, fullnamedown, fullnameup, iddown, idup, indexdown, indexup, namedown, nameup, pathcriticalnessdown, pathcriticalnessup, prioritydown, priorityup, responsibledown, responsibleup, sequencedown, sequenceup, startdown, startup, statusdown, statusup, tree
Context	csvtaskreport, htmlresourcereport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No

6.100. start <date>

start <date>			
Description	Specifies the start date of the report. In task reports only tasks that end after this end date are listed.		
Attributes	Name	Type	Description

start <date>			
	date	DATE	
Context	csvaccountreport, csvresourcereport, csvtaskreport, htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar,		
Inheritable	No	Scenario Spec.	No
See also	end		

6.101. start <date>

start <date>			
Description	The start date of the task		
Attributes	Name	Type	Description
	date	DATE	
Context	task,		
Inheritable	Yes	Scenario Spec.	Yes
See also	end, maxstart, minstart, startbuffer		

6.102. startbuffer <percent>

startbuffer <percent>			
Description	Specifies how much slack time you expect to have at the beginning of the task.		
Attributes	Name	Type	Description
	percent	REAL	Percent slack of the overall effort, duration or length of the task.
Context	task,		

startbuffer <percent>			
Inheritable	No	Scenario Spec.	Yes
See also	duration, effort, length		

6.103. startcredit <amount>

startcredit <amount>			
Description	Specifies an amount that is credited to the account specified by the account property at the moment the tasks starts.		
Attributes	Name	Type	Description
	amount	REAL	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	endcredit		

6.104. statusnote <text>

statusnote <text>			
Description	A note that describes the current status of the task.		
Attributes	Name	Type	Description
	text	STRING	
Context	task,		
Inheritable	No	Scenario Spec.	Yes
See also	note		

6.105. subtitle <text>

subtitle <text>			
Description	Specifies an alternative subtitle for a report column.		
Attributes	Name	Type	Description
	text	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	subtitleurl		

6.106. subtitleurl <url>

subtitleurl <url>			
Description	Specifies an URL that is attached to the column subtitle of HTML reports.		
Attributes	Name	Type	Description
	url	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	subtitle		

6.107. supplement <type>

supplement <type>

supplement <type>			
Description	<p>The supplement keyword provides a mechanism to add more attributes to already defined tasks or resources. The additional attributes must obey the same rules as in regular task or resource definitions and must be enclosed by curly braces.</p> <p>This construct is primarily meant for situations where the information about a task or resource is split over several files. E. g. the vacation dates for the resources may be in a separate file that was generated by some other tool.</p>		
Attributes	Name	Type	Description
	type	ID	Possible values are resource or task.
Context	The TJP File, task,		
Inheritable	No	Scenario Spec.	No
See also	resource, task		

6.108. task <id> <name>

task <id> <name>			
Description	<p>Tasks are the central elements of a project plan. Use a task to specify which resource should be allocated for how long to what task.</p>		
Attributes	Name	Type	Description
	id	ID	
	name	STRING	
Optional Attributes	account, allocate, complete, depends, duration, effort, endbuffer, endcredit, end, flags, journalentry, length, maxend, maxstart, milestone, minend, minstart, note, precedes, priority, projectid, reference, responsible, scheduled, scheduling, shift, startbuffer, startcredit, start, statusnote, supplement, task		
Context	The TJP File, task,		

task <id> <name>			
Inheritable	No	Scenario Spec.	No
See also	resource		

6.109. taskattributes <attribute> [, <attribute> ...]

taskattributes <attribute> [, <attribute> ...]			
Description	<p>The list of attribute names specifies which task attributes should be listed in the report in addition to the ones exported by default. The following values are supported. They correspond to the respective attributes of a task.</p> <p>complete, depends, flags, maxend, maxstart, minend, minstart, note, priority, responsible</p> <p>By specifying the ID of a user-defined attribute, these can be included as well.</p> <p>A special case the is <code>all</code> keyword. If this is part of the list, all supported task attributes will be included in the report. This includes all user-defined task attributes.</p>		
Attributes	Name	Type	Description
	attribute	ID	
Context	export,		
Inheritable	No	Scenario Spec.	No

6.110. taskprefix <prefix>

taskprefix <prefix>

taskprefix <prefix>			
Description	All tasks in the included file are added as sub-tasks of the task specified by <code>taskprefix</code> .		
Attributes	Name	Type	Description
	prefix	STRING	
Context	include,		
Inheritable	No	Scenario Spec.	No
See also	task		

6.111. taskroot <root>

taskroot <root>			
Description	Only tasks below the specified root-level tasks are exported. The exported tasks will have the id of the root-level task stripped from their ID, so that the sub-tasks of the root-level task become top-level tasks in the exported file.		
Attributes	Name	Type	Description
	root	ID	ID of a task that specifies the new root level
Context	export,		
Inheritable	No	Scenario Spec.	No

6.112. timezone <zone>

timezone <zone>

timezone <zone>			
Description	Sets the default timezone of the project. All times that have no time zones specified will be assumed to be in this timezone. The value must be a string just like those used for the TZ environment variable. Most Linux systems have a command line utility called <code>tzselect</code> to lookup possible values. The project start and end time are not affected by this setting. You have to explicitly state the timezone for those dates or the system defaults are assumed.		
Attributes	Name	Type	Description
	zone	STRING	
Context	project,		
Inheritable	No	Scenario Spec.	No

6.113. timeformat <format>

timeformat <format>			
Description	Determines how time specifications in reports look like.		
Attributes	Name	Type	Description
	format	STRING	See table below for possible values.
Context	htmlaccountreport, htmlresourcereport, htmlstatusreport, htmltaskreport, htmlweeklycalendar, project,		
Inheritable	Yes	Scenario Spec.	No
See also	shorttimeformat		

Ordinary characters placed in the format string are copied to `s` without conversion. Conversion specifiers are introduced by a ‘%’ character, and are replaced in `s` as follows:

- %a The abbreviated weekday name according to the current locale.
- %A The full weekday name according to the current locale.
- %b The abbreviated month name according to the current locale.

%B	The full month name according to the current locale.
%c	The preferred date and time representation for the current locale.
%C	The century number (year/100) as a 2-digit integer. (SU)
%d	The day of the month as a decimal number (range 01 to 31).
%D	Equivalent to %m/%d/%y. (Yecch - for Americans only. Americans should note that in other
%e	Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space
%E	Modifier: use alternative format, see below. (SU)
%F	Equivalent to %Y-%m-%d (the ISO 8601 date format). (C99)
%G	The ISO 8601 year with century as a decimal number. The 4-digit year corresponding to the
%g	Like %G, but without century, i.e., with a 2-digit year (00-99). (TZ)
%h	Equivalent to %b. (SU)
%H	The hour as a decimal number using a 24-hour clock (range 00 to 23).
%I	The hour as a decimal number using a 12-hour clock (range 01 to 12).
%j	The day of the year as a decimal number (range 001 to 366).
%k	The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by
%l	The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by
%m	The month as a decimal number (range 01 to 12).
%M	The minute as a decimal number (range 00 to 59).
%n	A newline character. (SU)
%O	Modifier: use alternative format, see below. (SU)
%p	Either 'AM' or 'PM' according to the given time value, or the corresponding strings for the
%P	Like %p but in lowercase: 'am' or 'pm' or %a corresponding string for the current locale. (C
%r	The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to '%I:%M:%S %p
%R	The time in 24-hour notation (%H:%M). (SU) For a version including the seconds, see %T b
%s	The number of seconds since the Epoch, i.e., since 1970-01-01 00:00:00 UTC. (TZ)
%S	The second as a decimal number (range 00 to 61).
%t	A tab character. (SU)
%T	The time in 24-hour notation (%H:%M:%S). (SU)
%u	The day of the week as a decimal, range 1 to 7, Monday being 1. See also %w. (SU)
%U	The week number of the current year as a decimal number, range 00 to 53, starting with the
%V	The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, v
%w	The day of the week as a decimal, range 0 to 6, Sunday being 0. See also %u.
%W	The week number of the current %year as a decimal number, range 00 to 53, starting with th
%x	The preferred date representation for the current locale without the time.
%X	The preferred time representation for the current locale without the date.
%y	The year as a decimal number without a century (range 00 to 99).
%Y	The year as a decimal number including the century.
%z	The time zone as hour offset from GMT. Required to emit RFC822-conformant dates (using
%Z	The time zone or name or abbreviation.
%+	The date and time in date(1) format. (TZ)
%%	A literal '%' character.

Some conversion specifiers can be modified by preceding them by the E or O modifier to indicate that an alternative format should be used. If the alternative format or specification does not exist for the current locale, the behavior will be as if the unmodified conversion specification were used. (SU) The Single Unix Specification mentions %Ec, %EC, %Ex, %%EX, %Ry, %EY, %Od, %Oe, %OH, %OI, %Om, %OM, %OS, %Ou, %OU, %OV, %Ow, %OW, %Oy, where the effect of the O modifier is to use alternative numeric symbols (say, Roman numerals), and that of the E modifier is to use a locale-dependent alternative representation.

The documentation of the `timeformat` attribute has been taken from the man page of the GNU `strftime` function.

6.114. `timingresolution <value> <unit>`

timingresolution <value> <unit>			
Description	<p>Sets the minimum timing resolution. The smaller the value, the longer the scheduling process lasts and the more memory the application needs. The default is 1 hour. The smallest value is 5 min.</p> <p>This value is a pretty fundamental setting of TaskJuggler. It has a severe impact on memory usage and scheduling performance. You should set this value to the minimum required resolution. Make sure that all values that you specify are aligned with the resolution.</p>		
Attributes	Name	Type	Description
	value	INTEGER	
	unit	UNIT	
Context	project,		
Inheritable	No	Scenario Spec.	No

6.115. `title <text>`

title <text>			
Description	Specifies an alternative title for a report column.		
Attributes	Name	Type	Description
	text	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	titleurl		

6.116. **titleurl <url>**

titleurl <url>			
Description	Specifies an URL that is attached to the column title of HTML reports.		
Attributes	Name	Type	Description
	url	STRING	
Context	columns,		
Inheritable	No	Scenario Spec.	No
See also	title		

6.117. **vacation <name> <interval>**

vacation <name> <interval>			
Description	Specify a global vacation day. This vacation is respected by all resources that are defined hereafter.		
Attributes	Name	Type	Description
	name	STRING	
	interval	DATEINTERVAL	
Context	The TJP File,		
Inheritable	Yes	Scenario Spec.	No
See also	vacation		

6.118. **vacation <interval>**

vacation <interval>			
Description	Specify a vacation period for the resource.		
Attributes	Name	Type	Description
	interval	DATEINTERVAL	
Context	resource,		
Inheritable	Yes	Scenario Spec.	No
See also	vacation		

6.119. version <number>

version <number>			
Description	Specifies which XML format should be generated. Currently version 2 is highly recommended.		
Attributes	Name	Type	Description
	number	INTEGER	
Context	xmlreport,		
Inheritable	No	Scenario Spec.	No

6.120. weeklymax <value> <unit>

weeklymax <value> <unit>			
Description	Sets the weekly limit of a resource usage or a resource allocation to a task.		
Attributes	Name	Type	Description
	value	REAL	
	unit	UNIT	
Context	limits,		
Inheritable	Yes	Scenario Spec.	No

weeklymax <value> <unit>	
See also	dailymax, monthymax

6.121. weekstartsmunday

weekstartsmunday			
Description	Specify that you want to base all week calculation on weeks starting on Monday. This is common in many European countries.		
Context	project,		
Inheritable	No	Scenario Spec.	No
See also	weekstartssunday		

6.122. weekstartssunday

weekstartssunday			
Description	Specify that you want to base all week calculation on weeks starting on Sunday. This is common in the United States of America.		
Context	project,		
Inheritable	No	Scenario Spec.	No
See also	weekstartsmunday		

6.123. workinghours <weekday> <interval> [, <interval> ...]

workinghours <weekday> <interval> [, <interval> ...]			
Description	The working hours specification limits the availability of resources to certain time slots of week days.		
Attributes	Name	Type	Description
	weekday	WEEKDAY	
	interval	TIMEINTERVAL	
Context	project, resource, shift,		
Inheritable	Yes	Scenario Spec.	No
See also	dailyworkinghours, yearlyworkingdays		

6.124. xmlreport <file>

xmlreport <file>			
Description	Generates a XML report. TaskJuggler 2.x has a much improved XML format. This is not yet the default, but will be in later versions. So you should always specify which version of the XML format should be generated.		
Attributes	Name	Type	Description
	file	STRING	
Optional Attributes	hideresource, hidetask, rollupresrouce, rolluptask, scenarios, version		
Context	The TJP File,		
Inheritable	No	Scenario Spec.	No

6.125. yearlyworkingdays <days>

yearlyworkingdays <days>

yearlyworkingdays <days>			
Description	Specifies the number of average working days per year. This should correlate to the specified workinghours and vacation. It affects the conversion of working hours, working days, working weeks, working months and working years into each other.		
Attributes	Name	Type	Description
	days	REAL	
Context	project,		
Inheritable	No	Scenario Spec.	No
See also	dailyworkinghours, loadunit, vacation, workinghours		

Chapter 7. The Example: Accounting Software

```
/*
 * This file contains an example project. It is part of the
 * TaskJuggler project management tool. It uses a made up software
 * development project to demonstrate some of the basic features of
 * TaskJuggler. Please see the TaskJuggler manual for a more detailed
 * description of the various syntax elements.
 */
project acso "Accounting Software" "1.0" 2002-01-16 2002-05-28 {
    # Pick a day during the project that will be reported as 'today' in
    # the project reports. If not specified the current day will be
    # used, but this will likely be outside of the project range, so it
    # can't be seen in the reports.
    now 2002-03-05-13:00
    # Hide the clock time. Only show the date.
    timeformat "%Y-%m-%d"
    # The currency for all money values is EUR.
    currency "EUR"

    # We want to compare the baseline scenario, to one with a slightly
    # delayed start.
    scenario plan "Plan" {
        scenario delayed "Delayed"
    }
}

# The daily default rate of all resources. This can be overridden for each
# resource. We specify this, so that we can do a good calculation of
# the costs of the project.
rate 310.0

# This is one way to form teams
macro allocate_developers [
    allocate dev1
    allocate dev2 { load 0.5 }
    allocate dev3
]

flags team

resource dev "Developers" {
    resource dev1 "Paul Smith" { rate 330.0 }
    resource dev2 "Sébastien Bono"
```

Chapter 7. The Example: Accounting Software

```
resource dev3 "Klaus Müller" { vacation 2002-02-01 - 2002-02-05 }

flags team
}
resource misc "The Others" {
  resource test "Peter Murphy" { maxeffort 0.8 rate 240.0 }
  resource doc "Dim Sung" { rate 280.0 vacation 2002-03-11 - 2002-03-16 }

  flags team
}

# In order to do a simple profit and loss analysis of the project we
# specify accounts. One for the development costs, one for the
# documentation costs and one account to credit the customer payments
# to.
account dev "Development" cost
account doc "Dokumentation" cost
account rev "Payments" revenue

# Now we specify the work packages. The whole project is described as
# a task that contains sub tasks. These sub tasks are then broken down
# into smaller tasks and so on. The innermost tasks describe the real
# work and have resources allocated to them. Many attributes of tasks
# are inherited from the enclosing task. This saves you a lot of
# writing.
task AcSo "Accounting Software" {

  # All work related costs will be booked to this account unless the
  # sub tasks specifies it differently.
  account dev

  task spec "Specification" {
    # The effort to finish this task is 20 man days.
    effort 20d
    # Now we use the above declared macro to allocate the resources
    # for this task. Since they can work in parallel, this task may be
    # finished earlier than 20 working days.
    ${allocate_developers}
    # Each task that does not have sub tasks must have a start or end
    # criteria and a duration. For this task we use a reference to a
    # further down defined milestone as a start criteria. So this task
    # cannot start, before the specified milestone has been reached.
    # References to other tasks may be relative. Each ! means 'in the
    # scope of the enclosing task'. To descent into a task the .
    # together with the id of the tasks have to be specified.
    depends !deliveries.start
  }
}
```



```
task software "Software Development" {

    # The software is the most critical task of the project. So we set
    # the priority of this tasks (and all sub tasks) to 1000, the top
    # priority. The higher the priority the more likely will the task
    # get the requested resources.
    priority 1000

    task database "Database coupling" {
        effort 20d
        # This task depends on a task in the scope of the enclosing
        # tasks enclosing task. So we need 2 ! to get there.
        depends !!spec
        allocate dev1, dev2
    }

    task gui "Graphical User Interface" {
        effort 35d
        # This task has taken 5 man days more than originally planned.
        # We record this as well, so that we can generate reports that
        # compare the delayed schedule of the project to the original plan.
        delayed:effort 40d
        depends !database, !backend
        allocate dev2, dev3
    }

    task backend "Back-End Functions" {
        effort 30d
        # This task is behind schedule since it should have been
        # finished already. To document this we specify that the tasks
        # is 95% completed. If nothing is specified, TaskJuggler assumes
        # that the task is on schedule and computes the completion rate
        # according to the current day and the plan data.
        complete 95
        depends !database, !!spec
        allocate dev1, dev2
    }
}

task test "Software testing" {

    task alpha "Alpha Test" {
        # Efforts can not only be specified as man days, but also man
        # weeks, man hours, etc. Per default taskjuggler assumes a man
        # week is 40 man hours or 5 man days. These values can be
        # changed though.
        effort 1w
        depends !!software
    }
}
```

Chapter 7. The Example: Accounting Software

```
        allocate test, dev2
        note "Hopefully most bugs will be found and fixed here."
    }

    task beta "Beta Test" {
        effort 4w
        depends !alpha
        allocate test, dev1
    }
}

task manual "Manual" {
    effort 10w
    depends !deliveries.start
    allocate doc, dev3
    account doc
}

task deliveries "Milestones" {

    # Some milestones have customer payments associated with them. We
    # credit these payments to the 'rev' account.
    account rev

    task start "Projectstart" {
        # A task that has no duration is a milestone. It only needs a
        # start or end criteria. All other tasks depend on this task.
        milestone
        start 2002-01-16
        # For some reason the actual start of the project got delayed.
        # We record this, so that we can compare the plan run to the
        # delayed run of the project.
        delayed:start 2002-01-20
        # At the begining of this task we receive a payment from the
        # customer. This is credited to the account associated with this
        # task when the task starts.
        startcredit 33000.0
    }

    task prev "Technology Preview" {
        milestone
        depends !!software.backend
        startcredit 13000.0
    }

    task beta "Betaversion" {
        milestone
        depends !!test.alpha
    }
}
```

```
        startcredit 13000.0
    }

    task done "Ship Product to customer" {
        milestone
        # The next line can be uncommented to trigger a warning about
        # the project being late. For all tasks limits for the start and
        # end value can be specified. Those limits are checked after the
        # project has been scheduled. For all violated limits a warning
        # is issued.
        # maxend 2002-04-17
        depends !!test.beta, !!manual
        startcredit 14000.0
    }
}

# Now the project has been completely specified. Stopping here would
# result in a valid TaskJuggler file that could be processed and
# scheduled. But no reports would be generated to visualize the
# results. So we request 7 HTML reports and 1 XML report to be
# generated. The XML report is used to create a Postscript Gantt
# chart.

# This task report is for use with the TaskJuggler GUI
taskreport "Project Overview" {
    columns start, end, effort, duration, completed, status, note, cost, rev
}

# A resource report for use with the TaskJuggler GUI
resourcereport "Resource Usage" {
    columns effort, freeload, utilization, rate
    hideresource 0
}

# For convenience we would like each report to contain links to the
# other reports. So we declare a macro with a fragment of raw HTML
# code to be embedded into all the HTML reports.
macro navbar [
rawhead
    '<table align="center" border="2" cellpadding="10"
    style="background-color:#f3ebae; font-size:105%">
    <tr>
        <td><a href="Tasks-Overview.html">Tasks Overview</a></td>
        <td><a href="Staff-Overview.html">Staff Overview</a></td>
        <td><a href="Accounting.html">Accounting</a></td>
        <td><a href="Calendar.html">Calendar</a></td>
    </tr>
```

```
<tr>
  <td><a href="Tasks-Details.html">Tasks Details</a></td>
  <td><a href="Staff-Details.html">Staff Details</a></td>
  <td><a href="Status-Report.html">Status Report</a></td>
  <td><a href="acso.eps">GANTT Chart (Postscript)</a></td>
</tr>
</table>
<br>'
]
```

```
# As the first report, we would like to have a general overview of all
# tasks with their computed start and end dates. For better
# readability we include a calendar like column that lists the effort
# for each week.
htmltaskreport "Tasks-Overview.html" {
  # This report should contain the navigation bar we have declared
  # above.
  ${navbar}
  # The report should be a table that contains several columns. The
  # task and their information form the rows of the table. Since we
  # don't like the title of the effort column, we change it to "Work".
  columns hierarchindex, name, duration, effort { title "Work"},
    start, end, weekly
  # For this report we like to have the abbreviated weekday in front
  # of the date. %a is the tag for this.
  timeformat "%a %Y-%m-%d"

  # Don't show load values.
  barlabels empty
  # Set a title for the report
  headline "Accounting Software Project"
  # And a short description what this report is about.
  caption "This table presents a management-level overview of the
project. The values are days or man-days."
}

# Now a more detailed report that shows all jobs and the people
# assigned to the tasks. It also features a comparison of the plan and
# delayed scenario.
htmltaskreport "Tasks-Details.html" {
  ${navbar}
  # Now we use a daily calendar.
  columns no, name, start, end, scenario, daily
  #start 2002-03-01
  #end 2002-04-01
  # Show plan and delayed scenario values.
  scenarios plan, delayed
  headline "Accounting Software Project - March 2002"
```

Chapter 7. The Example: Accounting Software

```
caption "This table shows the load of each day for all the tasks.
Additionally the resources used for each task are listed. Since the
project start was delayed, the delayed schedule differs significantly
from the original plan."
# Don't hide any resources, that is show them all.
hideresource 0
}

# The previous report listed the resources per task. Now we generate a
# report the lists all resources.
htmlresourcereport "Staff-Overview.html" {
    ${navbar}
    # Add a column with the total effort per task.
    columns no, name { cellurl "http://www.tj.org" }, scenario, weekly, effort
    scenarios plan, delayed
    # Since we want to see the load values as hours per week, we switch
    # the unit that loads are reported in to hours.
    loadunit hours
    headline "Weekly working hours for the Accounting Software Project"
}

# Now a report similar to the above one but with much more details.
htmlresourcereport "Staff-Details.html" {
    ${navbar}
    columns name, daily, effort
    # To still keep the report readable we limit it to show only the
    # data for March 2002.
    start 2002-01-16
    end 2002-04-01
    hidetask 0
    # The teams are virtual resources that we don't want to see. Since
    # we have assigned a flag to those virtual resource, we can just
    # hide them.
    hideresource team
    # We also like to have the report sorted alphabetically ascending by
    # resource name.
    sortresources nameup
    loadunit hours
    headline "Daily working hours for the Accounting Software Project -
    March 2002"
}

htmlweeklycalendar "Calendar.html" {
    ${navbar}
    headline "Ongoing Tasks - March 2002"
    start 2002-03-01
    end 2002-04-01
}
```

```
htmlstatusreport "Status-Report.html" {
    ${navbar}
}

# To conclude the HTML reports a report that shows how badly the
# project is calculated is generated. The company won't get rich with
# this project. Due to the slip, it actually needs some money from the
# bank to pay the salaries.
htmlaccountreport "Accounting.html" {
    ${navbar}
    # Besides the number of the account and the name we have a column
    # with the total values (at the end of the project) and the values
    # for each month of the project.
    columns no, name, scenario, total, monthly
    headline "P&L for the Accounting Software Project"
    caption "The table shows the profit and loss
        analysis as well as the cashflow situation of the Accounting
        Software Project."
    # Since this is a cashflow calculation we show accumulated values
    # per account.
    accumulate
    scenarios plan, delayed
}

# Finally we generate an XML report that contains all info about the
# scheduled project. This will be used by tjsx2ganttt to create a nice
# Gantt chart of our project.
xmlreport "AccountingSoftware.tjx" {
    # version 2
}
```

Chapter 8. Migrating from TaskJuggler 1.x to 2.0

8.1. Achiving compatibility

Are you also frustrated by tools that can't read the data of their earlier incarnations? After all those files contain your valuable data and the first impression that the wonderful new version makes is it's rejection to read your old files. With TaskJuggler we like to spare you such situations as much as possible. But TaskJuggler 1.x has been written to solve the problems that we encountered. By releasing it to the general public we learned that TaskJuggler is also very usefull to many other people. Some contacted us to tell us that it would be even more usefull to them, if TaskJuggler would have this or that new feature. In many cases we added these new features but we learned more and more that some parts of the original TaskJuggler design were not flexible enough to support some new features. For TaskJuggler 2.x we decided to change TaskJuggler to a more flexible design even if this meant that some syntax constructs could no longer be supported.

As TaskJuggler uses plain text file as its main data format, you will always to able to read in your old files. But in some cases, you need to change certain syntax constructs to the syntax. When TaskJuggler processes a file with deprecated syntax it will generate an error message. This usually contains a hint, how the statement should look like in new syntax. The following sections discuss the conceptual changes and what statements need to be changed.

8.1.1. Syntax changes

TaskJugger 1.x could only handle two scenarios with the fixed name `plan` and `actual`. TaskJuggler 2.0 can now handle any number of scenarios. Scenario specific task attributes have to be prefixed with the scenario ID followed by a colon. The attributes starting with `'plan'` or `'actual'` have been deprecated.

HTML reports are now a lot more flexible. New CSS elements have being used and the table elements are customizable now. Old stylesheets will no longer work, since the attribute names have changed. A HTML report contains CSS attribute class specification if you provide a custom stylesheet definition with `rawstylesheet`.

The scenario name is no longer displayed by default if more than one scenario is included in a report. A column `scenario` must be explicitly added if the scenario name should be reported for each line. The attributes `'showactual'` and `'hideplan'`

have been deprecated. The scenarios attribute now controls which scenarios should be shown.

The format of numbers and currency values can now be specified with `numberformat` and `currencyformat`. The old keyword `currencydigits` has been deprecated.

`workinghours` and `currency` are no longer global properties. They are now optional attributes of the project property.

Container tasks in export reports no longer have fixed start and end date if they have their sub tasks exported as well.

The functions for Logical Expressions are now using capital letters to improve their readability. The all lowercase versions are still supported, but the recommended versions are now the ones with intermixed uppercase letters. `isTaskOfProject` was added as new query function.

Support for new XML format has been added. The old format is still supported. TJ can read both old and new format XML files but will use the new XML format for output.

8.1.2. Scheduler changes

Length and duration tasks with resource allocations are no longer trimmed to the first and last resource allocation. This can lead to different schedules.

'length' based task now use the global working hours and global vacation settings as a criteria of what is a working day. The tasks now always end during working hours and not at midnight.

The maximum allocation of a resource for a task is no longer limited by default. `maxeffort` now defaults to 0 (unlimited) instead of 1.0 (8 hours per day). To have the same behaviour as in TaskJuggler 1.x version you need to specify `maxeffort 1.0` before any resource definition. This change was made since many users were confused when after increasing the daily working hours resources were still only allocated 8 hours per day.

Chapter 9. Questions and Answers

9.1. General Questions

Q: Why does taskjuggler use Qt when it's not an X11 application?

A: Qt is a very powerfull library that is much more than just a widget library. TaskJuggler uses Qt for all kinds of internal data types like lists and arrays. It also uses the Unicode functions, the SQL database interface and the XML support of Qt.

9.2. Compiling and installation

Q: Can TaskJuggler be compiled and used on Windows?

A: Probably yes, but we have never tried it. It should compile but require possible some minor tweaks of the sources. You should have a good knowledge of C++ and Qt when you try this. Let us know if you were successfull.

9.3. Usage

Nothing here yet.

Chapter 10. Copyright

TaskJuggler Copyright 2001, 2002, 2003 Chris Schlaeger <cs@suse.de>

This program is free software. You can redistribute it and modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Chapter 11. Trademarks

Linux is a registered trademark of Linus Torvalds.

KDE is a trademark of KDE e. V.

TaskJuggler is a trademark of Chris Schläger.

UNIX is a registered trademark and The Open Group is a trademark of The Open Group in the US and other countries.

