TaskJuggler (http://www.taskjuggler.org)マニュアル

Chris Schlaeger

Marc Ruehrschneck

TaskJuggler (http://www.taskjuggler.org) マニュアル
by Chris Schlaeger and Marc Ruehrschneck

This document describes TaskJuggler version 2.4.1

Table of Contents

この文書について

このドキュメントは TaskJuggler のバージョン 2.4.1 について記述しています。

TaskJuggler マニュアルは 2 つのパートから成っています。 最初のパートは一般的な情報とチュートリアルから成っています。 チュートリアルは TaskJuggler を始めて使うすべてのユーザが読むことを強くお薦めします。 これは TaskJuggler の数々の概念や機能のとてもよい入門書です。 2 番目のパートは言語リファレンスです。

Chapter 1. イントロダクション(未訳)

1.1. TaskJugglerとは

TaskJugglerは現代的で強力なプロジェクト管理ツールです。 プロジェクト計画とトラッキングに対するこのツールの新しいアプローチはよく使われているガンチャート編集ツールに比べてたいへん進んでいます。 TaskJugglerはすでに数々のプロジェクトで成功裏に利用されていて何百のリソースや何千のタスクを持つプロジェクトに容易に展開しています。

TaskJugglerは真剣なプロジェクト管理者にとってのオープンソースのツールです。 このツールではアイデアの起草からプロジェクトの完遂までにおけるプロジェクト管理の種々の活動のすべての領域を取り扱えます。 プロジェクトスコープ作成の間、リソース割り当て、コストや収入の計画、リスクやコミュニケーションの管理といったことに関しても援助します。

TaskJugglerにはスケジュールを自動生成する機能が提供されています。 このスケジューラはプロジェクトのスケジュールやプロジェクトの概略に基いたリソースの割り当てやプロジェクトの制約を計算します。 組み込みのリソース平準化機能と整合性検査機構を使うことで今気にする必要の無い詳細からとき放たれ、もしプロジェクトが期日に間に合わなかったと気には通知されます。 この融通性のある「必要なだけたくさんの詳細」アプローチ使うことで、プロジェクトを進めたいようにプロジェクトを計画できます。 このアプローチは"Extreme Programming"や"Agile Project Management"のような新しい管理手法に対して理想的に働きます。

超高層ビルを建築しようとするあるいは来月の同僚のシフト計画を取りまとめたいという場合、TaskJugglerは正に適切なツールです。 上司や投資家を印象付けるために見た目の良いガンチャートを作成したいだけであれば、TaskJugglerは適切なツールでないかも知れません。 このツールは、その力を会得するには幾分努力を要しますが、一旦会得してしまえばもう失いたくなるほどの伴侶となるでしょう。

1.2. 特徴とハイライト

- タスク、リソース、コストを一つのパッケージで管理する
- 自動リソース平準化、タスクとリソースの矛盾発見、タスクフィルタリング
- 後半で融通性のあるビューとレポート作成機能で必要な時に必要な情報を得られる
- 開始する際に使える組み込みのテンプレート(将来に使うため自分で作成したスケジュールをテンプレートとして取っておくこともできる)
- プロジェクトを追跡し状況をレポートする
- プロジェクトを便利に編集し、ガンチャートやレポートを見て作成するためのグラフィカルユーザインタフェース

- 「もしこうなったら」分析をするために、一つのプロジェクト内で無制限にシナリオ (ベースライン) を作成できる
- オフィスプログラムとデータを交換するために CSV データを作成すること
- リスク分析
- プロジェクト分析と管理チームのサポート
- 融通性のある労働時間と休暇の取り扱い
- シフト勤務へのサポート
- 複数時間帯へのサポート
- タスクが初期時コストと終了時コストを持つことが可能
- リソースはランニングコストを持ってもよい
- 利益と損失を分析することへのサポート
- HTML と XML 形式のレポート作成機能
- マクロのサポートのある強力なプロジェクト記述法
- リソースを管理する集中管理 DB のサポート

1.3. Web 上の TaskJuggler

公式の TaskJuggler ウェブサイトは http://www.taskjuggler.org (http://www.taskjuggler.org) でアクセスできます。

開発者はほとんどが多忙なプロジェクトマネジャーでもあるので、ユーザ自身による互助のためのフォーラム (http://www.taskjuggler.org/FUDforum2) を作成しました。

1.4. Change Log(Ver 2.4.1 まで日本語訳なし)

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`.
- Several 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 question mark 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.

1.4.15. Version 2.1.1 (2005-08-04)

- The code for the generation of iCal reports has been revived again. iCal is a standard format to exchange data with calendar applications such as KOrganizer.
- The contents of export reports can now be customized with the properties attribute. The report interval is customizable as well now.
- Add new chapter to manual that describes how to use TaskJuggler as a project tracking tool.
- The HTML version of the manual has now a new look and many more syntax examples have been added to the property reference.
- The TaskJuggler editor now supports printing of project files.
- Fixed build with GCC 4.
- Fixed build problems in the doc directory on Debian Unstable and FC3.
- We are now using docbook-utils instead of docbook-toys to generate the documentation.
- Filtering resources and tasks in the TaskJuggler GUI reports now always works properly.
- Fixed generation of reports with absolute file names.

- Make sure that all dates specified in project files lie within the Unix time space. For technical reasons we need to limit this to 1971-01-01 - 2035-01-01.
- Fixed some crashes related to out of project time specifications.
- Warnings about pre 2.0 deprecated syntax elements have been converted to errors.
- Fixed sorting of task reports when not using the default scenario as first scenario.
- Fixed projection scheduling mode. Tasks with bookings equal or larger than the effort lead to scheduling errors.

1.4.16. Version 2.2.0 (2005-12-05)

- The two graphical front-ends that have been present in earlier TaskJuggler releases have been merged into one new front-end. It's called TaskJugglerUI. The `ktjview2` and `TaskJuggler` executables are no longer included. This was done also to avoid name clashes on Windows/Cygwin.
- The TaskJuggler user interface now supports printing of high-quality task and resource reports.
- Major usability improvements for the GUI. It's now fully navigable by keyboard. The scaling/zooming of the Gantt chart has been improved.
- HTML reports are now rendered with an embedded browser instead of launching an external browser.
- Export reports are now loaded into the editor when selected in the report browser.
- The GUI supports now multiple project templates. The templates can be customized on loading to reflect the current date.
- Added date picker to GUI editor. By pressing CTRL+D the user can insert or change a date using the comfortable date picker widget.
- The GUI editor now supports search and replace over all files.
- The computation of completion degrees of container task has been improved to produce more meaningful values for all milestone or all effort tasks.
- To get Gantt and resource reports in the GUI the column 'chart' must be specified. They are no longer displayed automatically. This was done to have more consistency between the printed version of the GUI reports and the other reports.
- The default separator for CSV files is now a semicolon since this is what OpenOffice.org uses by default. But this can be changed if needed.
- The projection scheduling mode has been fixed and extended. In strict mode bookings will be scheduled only after the current date. In sloppy mode, bookings will also be scheduled prior to the current date for tasks that have no bookings at all. The modes can be set in the scenario definition.
- Fixed reporting of value 1,000 in US currency format.
- Fixed reported task duration value in all report types. Value was only correct when unit 'days' was used.

- Fixed account reports which had summary lines that were all 0. In HTML reports the summary columns were rendered all black.
- Fixed detection of cyclic brother tasks. This caused taskjuggler to go into a memory hogging endless loop.
- Fixed bug in priority handling. Under certain circumstances resources were allocated to lower priority tasks even though they should have been assigned to higher priority tasks.
- Fixed critical bug that turned ’precedes’ of parent tasks into ’depends’ of child tasks.
- Fixed cost accounting for non-working resources. They were always accounted to 0.

1.4.17. Version 2.3.0 (2006-09-05)

- Added improved error checking for ’timezone’ attribute. Only values as determined by tools like tzselect are allowed, e. g. "Europe/Berlin".
- The GUI GANTT chart now can highlight critical pathes. See minslackrate for details how to use the feature.
- The htmlweeklycalendar has been reworked. It now only list tasks if they are being worked on that day. A new attribute weekdays can be used to only show some days of the week. This can be used to hide e. g. weekdays.
- ICal reports now include events. The completion value is now shown correctly and assigned resources are included.
- Fixed handling of tasks that have ’precedes’ and ’depends’ attributes.
- Single and double quoted strings may now contain single or double quotes when escaped by a preceding backslash.
- Fixed ’gaplength’ and ’gapduration’ handling for other scenarios than the default one.
- Fixed ’duration’ column in all reports. Values equal or smaller than 24 hours were reported too high.
- No longer show dependency arrows in Gantt chart for inherited dependencies.
- Generate proper warning when bookings are assigned to container tasks or milestones.
- It is now possible to book off-hour and vacation time slots with ’booking’ when the overtime attribute is used.
- Added support for a more compact way to specify bookings. It’s now possible to list multiple comma seperated time intervals in a single booking statement.
- Fix build system so that kde-config is no longer mandatory. This simplifies compiling on Windows/Cygwin and Qt-only installs.
- Speed improvements for the loop detector. Large projects with many top-level tasks should be scheduled significantly faster now.

- 'start' and 'end' attributes specified for derived scenarios no longer cause accidental changes of the scheduling direction. This guarantees that a task has always the same scheduling direction in all scenarios.
- Added support for more compact workinghour specifications. `workinghours mon - fri 8:00 - 15:00` as well as `workinghours mon, sat, sun off` are now possible.
- Added man pages for taskjuggler and TaskJugglerUI.
- Fixed infinite loop bug in critical path detectors. With certain task dependencies TaskJuggler could get stuck forever when processing the project.
- Fixed TaskJugglerUI crash when processing a project with many runaway tasks.
- Better fit of report interval for printed Gantt charts.
- Fixed header of weekly and monthly CSV reports.
- XML reports now use the gzip compressed version 2 XML format by default.
- Add check to forbid assigning account groups to tasks.
- Added Turkish translations for TaskJugglerUI.

1.4.18. Version 2.3.1 (2007-01-30)

- Added support for automatic macros like `now`, `projectstart` and `projectend`.
- Added monthly calendar HTML report.
- Added `period` property as a shortcut for the combined use of `start` and `end`.
- Added more details to the pop-up info for tasks and resources in the UI.
- Added support for warnings and turned some non-critical errors into warnings.
- Improved scheduling performance for projects with long dependency chains.
- Added new attribute `purge` to clear inherited allocations and flags from a task or resource.
- Fixed complexity explosion in loop detector and critical path analyzer. Larger projects that made heavy use of inherited dependencies had an exponentially growing run time of these components.
- The threshold for the critical path detection now defaults to 10%. All pathes that have less than 10% slack time will be marked as critical.
- Fixed problem where editor tools were doubled when "Open Recent" was used while editor was open. This also triggered a crash on program exit.
- Fixed double headline problem in most HTML reports.
- Fixed a bug in the coloring of tasks in HTML reports that had a 'complete' specification.
- Fixed documenation for usage of sorting modes for scenario specific columns in reports and make them work properly.
- Force `now` date to be aligned to the timing resolution.

- Fixed a major bug in the handling of multiple scenarios. Values were inherited from peer scenarios instead of their parents.
- More meaningful error messages for some impossible combinations of fixed start/end times and dependencies.
- For technical reasons we had to limit the timingresolution between 5 minutes and 1 hour. Larger resolutions caused too many hazards in corner case situations.
- Fixed `gaplength` for time units other than days.
- Really hide all inherited dependency arrows in Gantt chart.
- Fix several crashes when the user is viewing reports in the UI after an unsuccessfull scheduling run.
- Fix sorting of tasks in interactive resource reports and sorting of resources in interactive task reports.
- The resource reporting in htmlweeklycalendar reports has been fixed. The report can now be used in task or resource reporting mode. The default is the task reporting mode which is closest to the previous behaviour.
- Fixed values of the revenue and profit column in resource reports. Resources can never generate a revenue so the value must always be 0.
- Fixed crash when printing resource reports from the UI that had a `hierarchyindex` column.
- The completion degree of container tasks that have sub tasks with and without resource allocations was reported as 0. This has been replaced with "in progress" as the completion degree cannot be calculated.

1.4.19. Version 2.4.0 (2007-07-03)

- For consistency and readability the notation of intervals without a dash between start and end date is slowly being deprecated. It was silently accepted in the project header and booking statements. This is not flagged with a warning. The project will still schedule fine.
- The critical path detector has been rewritten to reduce the complexity explosion that is triggered by lots of inherited dependencies in combination with long dependency chains. The number of searched paths is now limited to 10 million to avoid very long scheduling runs by default. This limit can be changed with the `maxpaths` attribute. A value of 0 means no limit.
- The default minimum slack rate has been changed to 5%.
- Added support for C++ style single line comments. Comment lines can now start with // or #.
- Added a warning when the working hours do not align with the timing resolution.
- The booking statements in export reports now include a `overtime 2` attribute. This avoids the error messages when the scheduling was based on overtime bookings and the export file is read-in again.
- Added a `Generate all Reports` option to the menu of the GUI.
- The sloppiness 3 for booking statements is no longer supported. The booking statements are processed in no particular order, so it’s undefined which booking will actually get the resource in a conflict.
- Removed support for KoTrus database.

- Fix HTML generation for HTMLWeeklyCalendar when cells are empty.
- Properly report durations in printed reports.
- Many editorial fixes were applied to the manual.
- Properly handle Pacific/Auckland DST.
- Fixed a number of memory leaks.
- Removed namespace collision for resource and account custom attributes and added support for user defined account attributes in the code.
- Make sure that files that have been modified on disk while edited by the TaskJugglerUI are detected properly. Probably with KDE 3.5.4 the behavior of the Kate library changed so that the test no longer worked properly and modified files were not detected.
- Fixed crash when non existant file was included.
- Detect usage of undefined macros again. Undefined macros were silently ignored. This should only happen when the macro name is prefixed with a questionmark in the macro call.
- Properly report effort and load of group resources that have children with an efficiency different than 0.
- Fixed a crash when an illegal date was specified in a project file.
- The XML reports now also include the accounts.
- Fixed a rounding error that caused dependency gaps to be one time slot short.
- The commandline version now properly returns a non-zero value if the report generation caused an error.
- Fixed the reversed sorting order for resource specific sorting criteria.
- Add workaround for new bahviour of tzset function in glibc 2.5.
- Fixed off-by-one-slot bug for limits on allocations with multiple resources.

1.4.20. Version 2.4.1 (2008-05-06)

- Fixed a serious bug in the optimizer part of the scheduling algorithm. When many tasks had the same priority, the path criticalness was not always respected resulting in longer running projects than necessary. The effect mostly showed under heavy resource pressure.
- Add `taskroot` support for XML reports.
- The task outlines in the UI resource reports have been replaced with relative load bars. One can now see what load is assigned to what resource at which point of time.
- Fixed calculation of effort based values in group task line items of resource reports.
- The isChildOf() function no longer returns true for self.
- Use alternating background pattern for Gantt chart to enhance readability.

- Make tasks and resources in Gantt charts selectable and enable RMB menu to edit and view them quickly.
- Added filter for GUI report list items.
- The report interval of interactive reports can now be changed from the GUI temporarily.
- Fixed the coloring of completed part of a container task in HTML task reports.
- Added new column `scheduling`, `completedeffort` and `remainingeffort` to reports. The first shows the scheduling direction of tasks and can be used to find potential sources for priority inversions. The other two show the task effort that has been completed already and the remaining effort.
- iCalendar files are now properly encoded when Unicode characters are used.
- The `hasAssignment()` function for logical expressions now properly accepts 3 parameters as documented.
- Enable `taskroot` support for csvtaskreports as well.
- Fixed a serious bug in the floating point formatter. Zeros right after the decimal separator were lost.
- Fixed display of progress bar in GUI Gantt chart. It sometimes extended the task bar in higher zooms.
- Changed the how-to-contribute section of the manual to use git instead of subversion. We no longer accept non-git patches now.
- Added 'criticalpath' attribute to task scenario section of XML reports.
- Fixed handling of multiple allocation with same mandatory resource set.
- Added new Pertt chart generator from Gregoire Barbier to the contrib directory.
- Improved the scheduling heuristic to generate projects with overall smaller project durations.
- New report column `hierarchlevel` was included.
- Hotkey for date/time picker in the GUI editor has been remapped to CTRL-SHIFT-T to avoid a conflict with built-in CTRL-D.
- Fixed sorting of resources in GUI resource reports.
- Unicode characters in macros no longer get corrupted.

1.5. 貢献の方法 (未訳)

1.5.1. なぜ貢献するの？ (未訳)

TaskJuggler is an Open Source Project. It was developed by volunteers mostly in their spare time. Made available under the GNU General Public license and similar licenses, TaskJuggler can be shared and used free of charge by anybody who respects the license conditions. Does that mean you can use it without worrying about anything? Clearly not! Though users have no legal obligation to contribute, you should feel a moral obligation to support Open Source in whatever way you can. This can range from helping

out other users with their first Linux installation to actively contributing to the TaskJuggler Project, not just as a programmer. The following section describes, how you can contribute to any of the components that are part of the TaskJuggler software releases.

1.5.2. 貢献するための準備 (未訳)

All TaskJuggler development is coordinated using the git (http://git.or.cz/) revision control system. All changes must be submitted using git so that we can track the authorship of each submission. To contribute you need to have at least git version 1.5.0 installed. As a first step, you need to checkout the latest version of the TaskJuggler. This will create a directory called `taskjuggler` in your current directory. It not only contains the latest sources, but also the full revision history of the code. It is your local copy of the TaskJuggler source repository.

```
git clone http://www.taskjuggler.org/git-repos/taskjuggler.git
```

If you have never used git before, you need to configure it first. You need to set your name and email address. This information will be present in all patches that you submit.

```
git config --global user.name "Your Name"
git config --global user.email "firstname.lastname@domain.org"
```

You then need to configure and install the TaskJuggler version. Make sure, you have removed all other instances of TaskJuggler removed from you system before doing so. It is a common mistake to have an old version of the TaskJuggler library used by a newer version of the executables.

```
cd taskjuggler
make -f Makefile.cvs
./configure # Put your options here (see ./configure --help
for details)
make
# Run as root
make install
```

Do not use the development snapshots and send your patches as plain diff files. After having switched to git, we no longer accept such patches.

Next you need to find the files where you want to make your modifications. Sometimes files will be generated from other files. Do not change those generated files. Your changes will be overwritten the next time you call the make utility. To identify those files, some familiarity with make and other Linux tools are helpful. Whenever there is a file with the same base name and the extension `.in` in the same directory, then the file is generated from the `.in`-file. You need to modify the `.in`-file, not the one with just the base name. Another indicator is the fact that the file is not part of the repository. With few exceptions the repository does not contain any generated files.

1.5.3. パッチを作成する (未訳)

When you are done with your changes, it's a good idea to test them. In the `taskjuggler` directory run the following commands.

```
make
# Run as root
make install
```

If there are no errors, you can check or test the result. If everything works fine, you can lock at your changes again.

```
git diff
```

The git-diff utility performs a line-by-line comparison of the files against the latest version in you local repository. Try to only make changes that have an impact on the generated files. Do not change indentation or line wrapping of paragraphs unless absolutely necessary. These kinds of changes increase the size of diff files and make it much harder to evaluate the patches. When making changes to the program code, please use exactly the same coding style. If your contribution is large enough to justify a copyright claim, please indicate what copyright you claim in the patch. For modifications to existing files, we will assume that your contribution falls under the same license as the modified file. All new files will need to contain a license declaration, preferably GPL version 2. In any case, the license must be an OSI accepted license (http://www.opensource.org/licenses) and be compatible with the rest of the project.

Review all changes carefully. In case you have created new source files, you need to register them with your repository.

```
git add FILENAME
```

If you think you are done, you can commit your changes to your local repository.

```
git commit -a
```

Whenever you have made a certain change or added a certain feature, you should commit your changes to your local repository. This keeps patches small and makes reviewing them easier. The easier your patches can be reviewed, the more likely they will get in.

The final step to submit your changes is to package them up and sign them. It is always a good idea to check for upstream changes again.

```
git pull
```

This makes sure you are really committing your patches against the latest version of the source code. In case there were upstream changes, you need to merge them first. Usually git does this automatically.

Refer to the git manual (http://www.kernel.org/pub/software/scm/git/docs/user-manual.html) for details on resolving conflicts during merges. Now you can create the patch or patch-set.

```
git format-patch -s origin
```

This will generate a number of files starting with 5-digit file names. You then need to attach these files to a posting in the TaskJuggler Developer Forum (http://www.taskjuggler.org/FUDforum2/index.php?t=thread&frm_id=5).

1.5.4. 翻訳に貢献する (未訳)

Another possible area of contribution are translations of TaskJuggler into languages other than US-English. Our development process is in principle prepared for translations, but the first translation will definitely be a bit difficult to do. TaskJuggler has several hundred messages and more than two hundred pages of documentation. Any translation is a significant effort and an ongoing commitment. TaskJuggler is still actively developed and this will require the translations to be updated as well. Please understand that we do not want to ship partial or outdated translations to our users, so please ensure you or a group of people are willing to maintain the translations.

1.5.5. 貢献したい方への最後の一言

すべての貢献を大変ありがたく思いますが、我々にはすでに述べたようなガイドラインに沿わないようなあるいは TaskJuggler チームのゴールと矛盾するようなあらゆる寄与に対して退ける権利を有していることをどうかご理解下さい。 あらゆる大きな試みを行なう前にチームに一言相談して下さるのはよい考えでしょう。

Chapter 2. インストール

2.1. TaskJuggler を手に入れる

TaskJuggler は以下の web サイトで手に入ります:

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

2.2. TaskJuggler のコンポーネント

TaskJuggler はいくつかの分離可能なコンポーネントからできています。 また TaskJuggler は他の追加のツールを使います。 私達 TaskJuggler 開発陣はそれが可能であるのならば車輪を再発明することを避けています。 私達はこれらの依存性が無理の無い小さなものにしようと努めてはいますが、にも関わらず未経験の方が TaskJuggler をソースコードから作成しインストールすることは挑戦的なことでしょう。 殆んどの人にとって、バイナリパッケージをディストリビュータから提供を受けるのが多分よりよいです。 最近では TaskJuggler は殆んどすべての主要なディストリビューションの最新のバージョンに含まれています。 もしそうでなければ、ディストリビュータに対して基本的なパッケージが抜けていることをお知らせしましょう。

このバージョンの TaskJuggler は SuSE Linux 10.0 と 10.1 の載った数々のハードウェアプラットフォームで作成してテストされました。

2.2.1. グラフィカル・ユーザ・インタフェース

バージョン 2.2 では新しいグラフィカルフロントエンドプログラムができました。 これは統合化されたプロジェクト管理環境です。

このフロントエンドは `TaskJugglerUI` といいます。 これと `taskjuggler` とを混同しないで下さい。 こちらはコマンドラインバージョンです。 シェルから TaskJugglerUI を起動したい時、大文字はこのとおり打って下さい。

グラフィカルフロントエンドは K Desktop Environment (http://www.kde.org) のライブラリを使っています。 最低でもバージョン 3.4 を使うことをお勧めします。 それ以前のバージョンは多分同様に動くでしょうが、テストはしていません。

2.2.2. コマンドラインツール **taskjuggler**

TaskJuggler を作って実行するために必要なものは:

- Qt — Qt C++ class library (ftp://ftp.trolltech.com/qt/source) version 3.3 以降
- GNU Compiler Collection — 現在 TaskJuggler を開発するのに使っているのは GCC 4.0 です。我々はプログラムコードをプラットフォームに依存せずそして ANSI 標準に沿うように書こうと努めていているので、他のほとんどのコンパイラーでも同様に動作するでしょう。

ここではプロジェクト記述を HTML や XML 形式のレポートに変換するコマンドラインプログラムを作成して使うのに必須のものをリストしています。 今御覧になっているドキュメントを作成したり XML ファイルを処理するツールを使用したいといった場合にはさらに追加の依存するツールをインストールする必要があります。 次に述べる依存関係を満足できない場合、configure スクリプトは警告を発しますが、失敗はしません。

2.2.3. **TaskJuggler** の文書

- DocBook4 — DocBook (http://www.oasis-open.org/docbook/) システムと他のスタイルシートとユーティリティー
- docbook-utils— エリック・ビショフ (Eric Bischoff) 作の jade wrapper scripts (ftp://sources.redhat.com/pub/docbook-tools)
- OpenJade — OpenJade (http://openjade.sourceforge.net/) システム
- JadeTeX — PostScript バージョンの文書を作成したいのならば、teTeX (http://www.tug.org/teTeX) の JadeTeX (ftp://ftp.dante.de/tex-archive/macros/jadetex) マクロパッケージが必要です。(訳注:2009-3-22 現在未対応)
- XSLT Processor — libxslt (http://xmlsoft.org/XSLT/) パッケージより xsltproc 他
- Meinproc — KDE (http://www.kde.org) ヘルプセンター用 XSLT プロセッサー。GUI を作成するときのみ meinproc が必要です。meinproc は KDE ディストリビューションの一部です。
- dvips — 通常 te_latex (http://www.tug.org/teTeX) のような TeX システムの一部です。

2.2.4. 寄与した方々

TaskJuggler に便利なツールを寄与した方々はたくさんいます。 これらのツールはソースコードパッケージの Contrib ディレクトリに置かれています。 インストールするための情報はそのなかに含まれている README ファイルを参照して下さい。

2.3. コンパイルとインストール

TaskJuggler をコンパイルする前に `QTDIR` 環境変数にあなたの環境の Qt のベースディレクトリを設定しなければなりません。 これは通常`/usr/lib64/qt3` や`/usr/lib/qt3` となっています。

あなたのシステムで TaskJuggler をコンパイルしてインストールするには、以下のコマンドを TaskJuggler のベースディレクトリでタイプします。

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

TaskJuggler は `autoconf` を使うので、コンパイルに当たり問題が発生することはないでしょう。 TaskJuggler にはいくつかの弱いあるいはまったく必要性の無い依存関係があります。 これらのうちいくつかが満足されないと警告が発せられます。 必須の依存関係が見つからない場合エラーが表示されて configure スクリプトは停止します。 もし問題に遭遇したら、そのことを TaskJuggler ユーザフォーラム http://www.taskjuggler.org/FUDforum2 に英語でレポートして下さい。 そこで誰かが救いの手を差し伸べてくれるかも知れません。

グラフィカルな機能無しで TaskJuggler を作成したい場合、コンフィギュレーションする際に機能をオフにできます。

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

Chapter 3. 使用法

3.1. 基本

TaskJuggler はプロジェクトを記述する際に一つあるいはそれ以上のテキストファイルを用います。メインプロジェクトは拡張子`.tjp`を持ったファイルに記述されなければなりません。 このメインプロジェクトはいくつかのサブプロジェクトを含んでも構いません。 そういったサブプロジェクトは拡張子`.tji`を持ったファイルに置かれなければなりません。 これらサブプロジェクトはメインプロジェクトにコンパイル時に取り込まれます。

TaskJuggler が他のツールと共に使われる時、プロジェクト記述あるいはレポートは XML 形式であってもよいです。 そういうファイルでは拡張子を`.tjx`にすることが推奨されています。

3.2. 一般的な使用法

グラフィカル版のフロントエンドを使用する時には、メニューから単に起動するだけです。 あなたのお使いの Linux のディストリビューションによりますが通常オフィスのプロジェクトマネジメントあるいは似たようなところで見つけることができるでしょう。 シェルから起動するには、以下をタイプして下さい。

```
% TaskJugglerUI
```

コマンドラインバージョンの TaskJuggler はコンパイラのように動きます。 ソースファイルを準備して、TaskJuggler が内容を計算して出力ファイルを作成します。

`AcSo.tjp`というプロジェクトファイルがあるとしましょう。 これにはプロジェクトのタスクとそれらの依存性に関して記載されています。 プロジェクトをスケジュールしてレポートファイルを生成するために TaskJuggler に処理を行なうよう依頼をしなければなりません。

```
% taskjuggler AcSo.tjp
```

TaskJuggler はすべてのタスクを指定された条件でスケジュールをし、入力ファイルの htmltaskreport, htmlresourcereport または他のレポートの属性で指定されたレポートを作成しようとします。

コマンドライン版のと比較して GUI 版はすべてのレポートをスケジュール後に直接作成することはしません。 直前に表示していたレポートだけが作成されます。 エクスポートファイルを含む他のすべてのレポートは画面に表示されるかマウスの右ボタンメニューで指示された時に作成されます。

3.3. コマンドラインオプション

--help	すべてのコマンドラインオプションを短い説明で表示する。
--version	バージョンと著作健情報を表示する。
-v	'--version' と同じ
-s	文法チェックを終えた後で TaskJuggler の実行を止める。このオプションはテストやデバッグ
-M	処理した TaskJuggler ファイルの依存関係をリストした Makefile の断片を出力する。
--makefile <file>	-M と同様だが依存情報を指定したファイルに書き出す。
--maxerrors N	TaskJuggler が入力ファイルのチェックを停止するエラーの数を指定する。もし N が 0 ならば
--nodepcheck	依存関係の輪のチェックをしない。 The loop detector uses an algorithm that needs exponentially
--debug N	デバッグ用出力を表示する; N は 0 と 4 との間でなければならない。 高い N はより多くの出
--dbmode N	デバッグ出力をコードのあるモジュールに限ります; N はビットマスクです。 各々のビットに
--warnerror	警告はエラーとして取り扱います。

TaskJuggler にファイル名を. として呼び出すと、標準入力から読み出すようになります。 標準出力に出力をリダイレクトするにはレポートファイル名として--を指定できます。 この機能は例えば CGI スクリプトから動的に HTML ページを生成するのに使われます。

3.4. バグの報告とフィードバックを送る

公式にリリースされた TaskJuggler は特別に記述されていない限り安定版とされます。 しかし我々のテスト範囲は非常に小さくまたいくつかの機能は自動でテストされているわけではありません。 それ故あなたが使用しているバージョンではバグがあるかもしれません。 バグを見つけた場合は、以下の手続きを踏んで下さい:

- このマニュアルを読んでそれが本当にバグであって機能ではないことを確かめる。
- TaskJuggler web page (http://www.taskjuggler.org) を確認して下さい。 もしかしたらそのバグは既に誰かに見つけられてパッチや回避法が存在しているかもしれません。
- バグが再現する可能な限り小さなプロジェクトを作成する。
- 再現したテストプロジェクトと問題を詳細に記述したものを開発フォーラム http://www.taskjuggler.org/FUDforum2 に送る。

Chapter 4. Tutorial: 最初のプロジェクト

TaskJuggler ではプロジェクトのスケジュールを通常のテキストファイルを用いることは既に述べました。 直に解るように、このファイルの文法はわかりやすくてとても直感的です。 この章では最初のプロジェクトを段階的に見ていきます。 まずは、会計ソフトウェアのプロジェクトを作るプロジェクトの計画を作成します。 例をすべて見る場合は Chapter 8(2008-7-21 現在未翻訳) を参照してください。 このプロジェクトでは TaskJuggler の基本的な機能のいくつかを使います。 より高度な機能は Chapter 6(2008-7-21 現在未翻訳) を参照してください。

4.1. プロジェクトを開始する

TaskJuggler でプロジェクトを開始するには、project property ファイルを使う必要があります。

```
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" {
    # 10%未満の slack 時間しか持たない全てのパスはクリティカルと示す
    minslackrate 10.0
    scenario delayed "Delayed"
  }
}
```

このファイルではデフォルトのプロジェクト ID とプロジェクトの短縮名とバージョン番号と開始日、終了日を指定します。 開始日と終了日は不正確でよいのですが、全てのタスクを覆わなくてはなりません。 TaskJuggler のスケジューラはこれらの日付け間隔を使用してタスクを調整します。そのため、すべてのタスクが収まるように十分長くしましょう。 しかし、余り長くし過ぎないようにしてください。 というのも、結果としてスケジューリングにより長く時間がかかりメモリ使用量も高くなります。

TaskJuggler の property には決った属性 (attribute) とオプション属性とがあります。 オプション属性は中括弧 ({}) で囲みます。 ここの例では、オプション属性 now を用い、スケジューラに TaskJuggler を起動時刻から他の値に現在日時を設定しています。 指定されたプロジェクト期間から一日を選び、この最初のプロジェクトを実行した時期に関らずいつでも同じ結果を出します。 通貨 (currency) 属性は通貨の単位を指定します。

日付けの指定方法は個々の文化で異っているので、書式は構成可能です。 規定の書式を指定するためには timeformat 属性を使います。 これはレポート作成時に使用される書式であり、

TaskJuggler のプロジェクトファイルで使う書式ではありません。 この時の書式は固定したものであり、年-月-日-時:分:秒-時間帯です。 “ 日 ” より後はすべてオプションです。

このチュートリアルでは二つのシナリオ (scenario) を比較します。 最初のシナリオは計画したものです。 二つ目のシナリオは実際に起こることです。 二つのシナリオは同じタスク構造を持ちますが、開始日と終了日は変わるでしょう。 現実世界ではプロジェクトが遅延することが想定できるので、二つ目のシナリオを “ 遅 延 (Delayed) ” と呼びます。 scenario property 属性はシナリオを指定するために使います。 遅延シナリオは計画シナリオの中に収められます。 この属性より TaskJuggler は、遅延シナリオで独自に値を設定していない全ての値を計画シナリオから流用します。 シナリオに値を指定する方法は、後でより深く見て行きます。

ある種の属性はシナリオ専用です。 そういった属性はシナリオ定義に含まれなければなりません。 この例では “ slack 時間 ” をパーセントで指定するために minslackrate 属性を使っています。 “ slack 時間 ” とはタスクのパスがクリティカルパスと看做される前に少なくとも持たなければならない時間です。 クリティカルパス上のタスクは GUI 上では赤枠で示されます。

4.2. Global 属性

プロジェクトの適切な開始日と終了日をみつけることの他に、単純な利益と損失の解析もしたいでしょう。 そのためには各従業員の規定の日次コストを指定しなければなりません。 これは後に特定の従業員に対して変更できますが、TaskJuggler の重要な 概念 – 属性の継承を例示します。 TaskJuggler プロジェクトファイルのサイズ可読な最小に収めるため、 property は数々のオプション属性をそれを囲むスコープから継承します。 これが実際に意味するところは、後により詳細に見ていきます。 今最上位のスコープにいると、これは以後の全ての属性に対する規定値となります。

```
rate 310.0
```

rate 属性はリソースの日々のコストを指定するために使います。 続く全てのリソースは別に指定されない限りこのレートを使用します。

全てのリソースに関係ある休日に関する情報を指定したいこともあると思います。 Global 休日を指定すると、その期間はどのタスクにもリソース割り当てを行いません。

```
# "Good Friday"をすべてのリソースの Global 休日として登録する
vacation "Good Friday" 2002-03-29
```

vacation 属性を使って Global 休日を定義します。 Global 休日は名称と日付あるいは日付範囲を持たなくてはなりません。 これは個々人のリソースの休日と若干異なります。 リソースに対しては vacation 属性で定義しそこには名称はありません。

マクロはプロジェクトファイルを小さくするための TaskJuggler のもう一つの機能です。 マクロは一度定義すればプロジェクトファイル中で何度も挿入できるテキストパターンです。 マクロは名前を持ち、そのテキストパターンは角括弧 ([]) で囲まれています。

```
macro allocate_developers [
  allocate dev1
  allocate dev2 { limits { dailymax 4h } }
  allocate dev3
]
```

マクロを使用するためには単に`${allocate_developers}`と記述します。 TaskJuggler は`${allocate_developers}`をパターンで置き換えます。 後にこの例中でこのマクロを用いてパターンの意味を説明します。

4.3. リソース定義

TaskJuggler の機能のうちで、おそらく頻繁に使用されるものは flags でしょう。 一度宣言すればたくさんの属性に付与することができます。 TaskJuggler の結果を生成する際、flags を使用することで、情報を取り除いて、載せるべき詳細を正確にレポートするように限定することができます。

```
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" { limits { dailymax 6.4h } rate 240.0 }
  resource doc "Dim Sung" { rate 280.0 }

  flags team
}
```

例の中のこの一部で resource 属性の使い方がわかります。 resouce は常に ID と名称を持ちます。 ID は ASCII 文字、数字そして下線からのみ成ります。 すべての大域的な TaskJuggler の属性は ID を持ちます。 そういった ID は各々の属性クラス内で唯一でなければなりません。 ID は必須で、そうすることで後に再度属性をより長い名称を記述することなく参照できます。 名称は文字列で二重引用符 (") で囲まれます。 文字列は、非 ASCII 文字を含み、どんな文字から成っていてもよいです。 見ておわかりのとおり、resource 属性は入れ子にできます:`dev` は仮想リソースで、それはさらに 3 つのリソースから構成されています。

`dev1` - これは Paul Smith の別名ですが - は通常の従業員よりコスト高です。 ですから、`dev1` の宣言は継承した既定値を 330.0 という新しい値で置き換えます。 既定値は入れ子のスコープであるリソース `dev`、から継承されます。 このリソース `dev` は今度はコストを大域スコープより継承します。

Klaus Mueller の宣言ではもう一つのオプション属性を使っています。 vacation でそのリソースが利用できない期間を指定できます。

ここで TaskJuggler がどのように期間を取り扱うかを理解する必要があります。 TaskJuggler は、内部的には 1970 年 1 月 1 日から経過した秒数を用いてすべての日付を取り扱っています。 ですから、すべての日付は実際に 1 秒の精度で取り扱われます。 `2002-02-01` という記述は、2002 年 2 月 1 日の真夜中を意味します。 ここでもう一度 TaskJuggler は必要な最小限の情報を要求し、それ以外を意味のある既定値で補うコンセプトであることを受け、特に指定されない限り 0:00:00 を補います。 ですから、休暇は 2002 年 2 月 5 日の真夜中に終了します。 いやぁ危うい。 日付期間を指定したときはいつも、終了日は期間に含まれず、指定した日付の直前の秒までです。それゆえ Klaus Mueller の休暇は 2002 年 2 月 4 日の 23:59:59 に終了します。

Peter Murphy は一日に 6.4 時間しか働きません。 それゆえ、limits 属性を用いて労働時間を制限します。 あるいは正確な労働時間を shift 属性を用いることで定義できますが、ここでは割愛します。

特筆すべきことは、子リソースの宣言の後に `team` フラグをチームリソースに付与したことです。このようにこれらのフラグは子リソースに引き継がれません。 フラグを子リソースの前に宣言したならば、そのリソースは同様にフラグを付与するでしょう。

4.4. 口座を定義する

リソースを使用することでコストが発生します。 利益と損失の分析をするために、コストを顧客の支払と対比させなければなりません。 種々の全ての元利合計を失わないために、３つの口座を定義して元利合計を信用貸しするようにします。 一つは開発者用、もう一つは文書作成コスト用そして顧客支払用に一つ作成します。

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

account 属性には 3 つの決った属性 - ID, name, type - があります。 type は `cost` か `revenue` のどちらかでなければいけません。 分析するために、TaskJuggler はすべての cost 合計をすべての revenue 合計から引算します。

account はまた入れ子にできます。 入れ子になった account は型を特定しなくてもよいです。 その場合は上位の account の型を引き継ぎます。

4.5. タスクを定義する

ここで実際の作業に目を向けて見ましょう。 このプロジェクトではある問題を解くとします。 それは会計ソフトを作成することです。 その作業は大変複雑なので、まずいくつかのサブタスクに分解します。 分解したタスクには仕様を策定しソフトウェアを開発しテストをしてマニュアルを作成することが含まれています。 TaskJuggler の文法では、これは以下のようになります:

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

リソース同様、タスクも task キーワードを用いて宣言します。 task キーワード定義の後には ID と名前が続きます。 TaskJuggler 内の全ての属性には、それぞれの名前空間があります。 ですから、リソースとタスクとが同じ ID を持っても全く問題ありません。 タスクにはオプションの属性を置くことができ、それはタスクでもよいです。 このことからタスクはネストできることになります。 他のすべての TaskJuggler の属性と違い、タスク ID は囲まれているタスク ID を自身の ID の接頭辞として引き継ぎます。 例えば `spec` タスクの完全な ID は `AcSo.spec` となります。

プロジェクトの重要なマイルストンを追跡するため、"Milestones"と呼ぶタスクを更に加えました。 このタスクは、多くの他のタスク同様、後にいくつかのサブタスクを持つことになるでしょう。 "Specification"タスクは十分単純に扱いたいため、それ以上のサブタスクに分解する必要はありません。 そこでこのタスクにいくつか詳細を加えましょう。

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

タスク完了までの期間は 20 人月と指定します。他にも length 属性や duration 属性というものがあります。 `length` 属性は労働日でのタスクの期間を指定し、`duration` 属性は (非稼働日含めた) カレンダー上での期間を指定します。 effort 属性に反して、length 属性と duration 属性は関係するリソースの詳細を持つ必要はありません。 `effort` 属性は人月での期間を指定するので、誰をタスクに割り当てるか述べなければなりません。 タスクが終了するには、指定された effort 期間内に十分リソースが割り当てられなければなりません。 `length` または `duration` 基準が付与されると共にリソースを確保されたタスクは要求された期間だけ継続します。 リソースは可能な場合のみ割り当てます。

ここで上で述べた `allocate_developers` マクロを使います。

```
${allocate_developers}
```

という記法は単に

```
  allocate dev1
  allocate dev2 { limits { dailymax 4h } }
  allocate dev3
```

と展開されます。

同じグループの人々をいくつかのタスクに割り当てるのであれば、マクロを使うことでキー入力を削減できます。 マクロを使う代わりに allocate 属性を直接記述することもできます。 複数のリソースを一つのタスクに割り当てるのにマクロを使用するのはごく自然なことなので、この例でもそのようにします。

更に興味深いことに、リソース `dev2` はこのタスクでは 1 日の 50%だけ仕事させるため、オプション属性 limits を使って指定しています。

TaskJuggler がタスクをスケジュールできるようにするため、タスクの開始と終了の基準日かまたはそのどちらかと期間が指定されていなければなりません。 開始基準日と終了基準日は固定日付か相対日付のどちらかです。 相対日付は"タスク A が終了後タスク B が開始する"という形式の指定方法です。 言い換えると、タスク B はタスク A に依存するのです。 この例では `spec` タスクは `deliveries` タスクのサブタスクに依存します。 まだ定義していませんが、それは ID `start` を持ちます。

2 つのタスク間の依存性を指定するには、depends 属性を使います。 この属性の後には一つ以上のタスク ID を続けなければなりません。 2 つ以上の ID が指定された場合、各々の ID はコンマ (,) 記号で分けます。 タスク ID は絶対 ID でも相対 ID でも構いません。タスクの絶対 ID とはこのタスクを囲むすべてのタスクの ID が前に付いた ID です。各々のタスク ID はドット (.) で区切ります。 "specification"タスクの絶対 ID は `AcSo.spec` となります。

相対 ID は一つ以上の感嘆符 (!) で始まります。 一つの感嘆符はスコープを次の抱合しているタスクへと移動します。 そのため、`AcSo` が `deliveries` を囲むタスクである場合、`!deliveries.start` は `AcSo.deliveries.start` と展開されます。 相対タスク ID は最初は少しとまどいますが、絶対 ID より実際役に立ちます。 遅かれ早かれプロジェクト内でタスクを移動することになり、その結果相対 ID の依存関係を修正する必要が殆どないことがわかるでしょう。

ソフトウェア開発のタスクは複雑で直接定義することができません。それゆえサブタスクに分割します。

```
task software "Software Development" {
  priority 1000
  task database "Database coupling"
  task gui "Graphical User Interface"
  task backend "Back-End Functions"
}
```

ここで priority 属性を使ってタスクの重要度を明示します。 500 は最上位のタスクのデフォルトの優先度です。 この属性に設定できる範囲は 1(全く重要でない) から 1000(究極の重要度) であるので、優先度を 1000 にすると、そのタスクは最重要タスクとして明示することになります。 `priority` はサブタスクの宣言の前に指定されると、サブタスクに継承される属性です。 ですから自身の優先度を持たない限り、すべての `software` のサブタスクは同様に 1000 の優先度を持っています。

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

"database coupling"の作業は"specification"が終了するまで開始しません。 それゆえ今回も depends 属性を使い、TaskJuggler にそのことを指示します。 今回相対 ID の指定で 2 つの感嘆符を使います。 最初のは `software` タスクのスコープを意味します。 2 つ目のは `spec` タスクを含んでいる `AcSo` スコープになります。 今回はリソースを割り当てるのにマクロを使用せずに行います。

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

TaskJuggler はプロジェクトを 2 つのシナリオで定めます。 最初のシナリオは"plan"シナリオと呼び、2 つ目を"delayed"シナリオと呼びます。 双方のシナリオの数値を隣合わせで記載できる報告書が多数あるため、2 つのシナリオを比較することができます。 "delayed"シナリオに対して明示していないシナリオ限定のすべての値は"plan"シナリオから引き継がれます。 そのため"delayed"シナリオと違った値を指定するだけでよいです。 2 つのシナリオは同じタスク構造を持ち同じ依存性を持たなければなりません。 しかしタスクの開始日と終了日は期間同様変化してもよいです。 例ではグラフィカルユーザインタフェースの作業を 35 人日と計画しました。 実際には 40 人日必要だということが分かりました。effort 属性を `delayed:`という接頭辞を付けることで、"delayed"シナリオの"effort"値を定義できます。

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

既定では、TaskJuggler はすべてのタスクが計画どおりとみなします。 時としてタスクが実際に完了している度合いを見せるレポートが必要になることもあるでしょう。 now 属性を使って他の日を指定しない限り、TaskJuggler はレポートを作成するのに現在の日を使用します。 タスクがスケジュールに先行しているかあるいは遅延しているならば、それは complete 属性で指定できます。この属性でタスクが現在までに何%完了したのかを指定します。 レポートからわかるように、我々のケースでは back-end の実装がスケジュールに若干先んでいます。

```
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
  }
}
```

"software testing"タスクはαテストタスクとβテストタスクに分割されました。 ここで興味深いことは effort が人月 (man day) としてだけでなく、人週 (man week) や人時 (man hour) などでも指定できることです。 既定では、TaskJuggler は人週は 40 人時あるいは 5 人日と仮定します。 これらの値は dailyworkinghours 属性を用いて変更できます。

もう一度最も外のタスクに戻りましょう。 この例の最初で、すべての開発活動を IDdev の口座一つに課金し、すべての文書化活動を doc 口座に課金すると述べました。 この目的を達するため、account 属性を用いて dev 口座に対するすべてのタスクに課金します。

```
task AcSo "Accounting Software" {

  account dev

  task software "Software Development" {
```

この属性を他のすべてのサブタスクを宣言する前に最上位のタスクで定義したので、この属性はすべてのサブタスクとサブタスクのサブタスクなどなどに継承されます。 唯一の例外はマニュアルの作成です。このタスクは AcSo のサブタスクでもあるので、口座は再度変更する必要があります。

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

4.6. マイルストーンを指定する

これまで述べてきたタスクすべてはある期間を持っていました。 duration はいつも明示的に指定しませんでしたが、ある一定期間続くと期待しています。 時にプロジェクト計画の中にある瞬間を取り込みたくなるでしょう。 こういった瞬間は、プロジェクトの進捗に対してある程度の重要性を持つので、通常マイルストーンと呼ばれます。

TaskJuggler はマイルストーンのサポートもします。 マイルストーンはある特殊な型を持ったタスクとして取り扱います。 オプションの属性 milestone をタスクに使用することで、このタスクをマイルストーンとして宣言します。 マイルストーンは期間を持たないため、期間を指定したり異なった開始時刻と終了時刻を指定することはどんな形でも間違いです。

```
  task deliveries "Milestones" {

    account rev

    task start "Project start" {
      milestone
      start 2002-01-16
      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
    }
  }
}
```

我々はプロジェクトのすべての重要なマイルストーンを `deliveries` タスクのサブタスクに置きました。 このようにすることでレポートによくまとめられます。 すべてのマイルストーンはそれぞれ依存関係があるか固定した開始日があります。 最初のマイルストーンとして属性 start を固定した開始日をセットして使いました。 他のすべてのタスクはこのタスクと直接あるいは間接に依存関係があります。 開始日を変えることですべてのプロジェクトをスライドすることができます。これは実際起こりました。そのため再度 `delayed:`前置詞を用いて遅延シナリオに開始日を指定します。

すべてのマイルストーンは顧客の支払に結びつけられます。 startcredit 属性を用いることで指定された合計をこのタスクに割り当てられた口座に負課できます。 `rev` 口座を取り囲むタスクに割り当てたので、すべてのマイルストーンはこの口座を同じように使います。

ハッシュ(#) で始まった行が `done` タスクにあるのに気が付きましたか？ このラインはコメントアウトです。 ハッシュが現れると TaskJuggler は行の残りを無視します。 このようにプロジェクト内にコメントを含めることができます。 maxend 属性を付けるとタスクが指定された日時を越えない

ようにします。 この情報はスケジューリングでなく後にスケジュールをチェックするのに使われます。 タスクは指定された日より後に終らないので、ラインをコメントアウトすることで警告が発せられます。

さてプロジェクトは完全に定義されました。 ここで正しい TaskJuggler ファイルができあがり、それは処理されスケジュールが作成されました。 しかし結果を可視化するために作成するレポートはありません。

4.7. スケジュールされたプロジェクトのレポートを作成する

TaskJuggler は数々のレポート形式を提供します。 たぶん最も知られているものは対話的レポートと HTML レポートでしょう。

4.7.1. 対話的なレポートを作成する

対話的レポートは TaskJuggler GUI でのみ利用可能です。 コマンドラインバージョンは対話的なレポートの定義を単に無視します。 GIU でタスクレポート (task report) を見るために、プロジェクトに以下の行を追加して下さい。 この行でプロジェクトのすべてのタスクのリストと伝統的なガンチャートを得ます。

```
taskreport "Gantt Chart" {
  headline "Project Gantt Chart"
  columns hierarchindex, name, start, end, effort, duration, chart
  timeformat "%a %Y-%m-%d"
  loadunit days
  hideresource 1
}
```

GUI は `name` 列がレポートにあることを要します。 指定しなければ自動的に加えます。 インデックス列のサポートはなく、GUI では表示されません。 にも関わらず、対話的レポートの印刷されたものが他のレポートのように振舞うので、それを要求することは良い考えです。 レポートでは指定された列のみ表示されます。 レポートが 1 ページに収まらない場合、印刷用レポートは最初の列を毎ページに印刷します。 GUI からレポートを印刷するためには、`File->Print` をメニューから選択するかツールバーのプリンタアイコンをクリックして下さい。

このレポートのために日付の前に省略された曜日を入れましょう。 `%a` はこのためのタグです。 すべてのオプションの完全なリストは manual を参照して下さい。

このレポートではどのリソースも見せたくないので、すべて隠します。 1 はリソースをいつも隠すことを意味します。 すべてのリソースを表示するためには 0 とします。 嗣ぎのレポートで行なったように、葉のリソースのみ表示して名前でソートする論理表現を記述することもできます。 これは個々のタスクに割り当てられたリソースを表示してすべてのタスクをリストするものです。

```
taskreport "Task Usage" {
  headline "Task Usage Report"
  columns hierarchindex, name, start, end, effort { title "Work" }, duration,
    cost, revenue
  timeformat "%Y-%m-%d"
  loadunit days
  hideresource ~isLeaf()
  sortresources nameup
}
```

`effort` 列に対する既定の表題は指定された表題で置き換えます。 加えてすべてのタスクとリソースで使われたコストと収入も表示します。 すべての仕事量はリソース日として表示します。

次のレポートは最初のものに似ていますが、完了度を追加の列として表示します。

```
# A list of all tasks with the percentage complete for each task
taskreport "Tracking Gantt" {
  headline "Tracking Gantt Chart"
  columns hierarchindex, name, start, end, effort { title "Work" }, duration,
    completed, chart
  timeformat "%a %Y-%m-%d"
  loadunit days
  hideresource 1
}
```

リソースを中心としたレポートも作成できます。 それには resourcereport レポート型を付けます。 次のレポートはリソースの割当を表示します。 これでは各リソースが割り当て不足あるいは割り当て過多であるかどうかがわかります。

```
resourcereport "Resource Graph" {
  headline "Resource Allocation Graph"
  columns no, name, rate, utilization, freeload, chart
  loadunit days
  hidetask 1
}
```

次のレポートは人的あるいは物質的双方のすべてのプロジェクトリソースを割り当てられたコストと共にリストします。

```
resourcereport "Resource Sheet" {
  headline "Resource Sheet"
  columns no, name, efficiency, id, maxeffort, rate
  loadunit days
  hidetask 1
}
```

次のレポートは先のものに似ていますが、各々のリソースに対するタスクをリストします。 このレポートでは各々のリソースとタスクに対するコストについての詳細を表示します。

```
# A list of resources and each task associated with each resource.
resourcereport "Resource Usage" {
  headline "Resource Usage Report"
  columns no, name, utilization, freeload, cost
  loadunit days
  hidetask 0
}
```

4.7.2. HTML レポートを作成する

対話的レポートに加えて、TaskJuggler は HTML レポートも提供します。 これらのレポートは web サーバで配布できるレポートファイルを作成できるといった利点があります。 TaskJuggler は数々の HTML レポートを提供します。 完全なリストはマニュアルを参照して下さい。 このチュートリアルではその中のいくつかを見るだけにします。

最初の HTML レポートは通常のカレンダーのように見えます。 これではそれぞれの日で実施されるタスクすべてを一覧します。

```
# This report looks like a regular calendar that shows the tasks by
# their dates.
htmlweeklycalendar "Calendar.html" {
}
```

次の HTML レポートは週次のステータスレポートです。 毎週チームメンバーや管理職そして顧客に現在のプロジェクトステータスを報告するためにこういったレポートを作成します。 このレポートでは最近の週におきたすべてのイベントと次週に行なう新規のタスクをすべて一覧します。

```
# This report is a status report for the current week. It also
# provides an outlook for the next week.
htmlstatusreport "Status-Report.html" {
}
```

HTML レポートを締めくくるためにプロジェクトがどのくらい貧乏かを計算して見せるレポートを生成します。 会社はこのプロジェクトでお金持ちになりません。 スケジュールの延期のため会社は銀行からお金を借りて給料を払う必要があります。

```
htmlaccountreport "Accounting.html" {
  columns no, name, scenario, total, monthly
  headline "P&L for the 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
}
```

htmlaccountreport 属性で上で見たようなレポートを作成しますが、それはタスクやリソースの代わりに口座の一覧を表示します。 `total` 列はレポート作成日の時点での口座の額を表示します。 accumulate 属性はカレンダーを累積モードにします。 月次列は月末での口座の額をリストします。 通常アカウントに追加されたあるいは引かれた量が表示されます。

Chapter 5. 利用ガイド

5.1. トラッキングとプロジェクト

初期の計画を作成してプロジェクトが開始したら、TaskJuggler は計画用のツールからトラッキングのツールへの変貌します。 そうするためにたくさんの変更をしなくてもよいです。 結局、初期のプランというものはほとんどいつも単に最初の推測でしか過ぎないので、新しいことが明らかになるにつれてプロジェクトを計画し続ける必要があるのです。 それゆえ本当に要することは計画を作業が完了するにしたがって徐々に固めていく方法なのです。 そうすることですべての将来の作業に完全な自由度を持ち続けることになります。

プロジェクト計画に時間を投資することは一般的に受け入れられている一方、一旦プロジェクトが開始されると、プロジェクトマネジャーはプロジェクトの適切なトラッキングを避けがちです。 プロジェクト計画のほとんど大多数は管理職や投資家の承認を得るために作成された我々は予想します。 一方厳格なプロジェクト管理の技術を使って終了したプロジェクトがあります。 その技術は詳細にステータスをトラッキングするのです。 双方の極端な例は多分に支援者がいて TaskJuggler は双方に対してそれらの間に位置するさまざまな技術同様に適切にサポートします。

5.1.1. 進捗を記録する

すでに述べたように、初期のプロジェクト計画というものは単にプロジェクトの進捗を見積もった第一番目のものに過ぎません。 プロジェクトの進行中に新しい情報を採り入れる必要が出た時に計画を変更しなければならなかったり、プロジェクトの進捗を決まった形式でトラッキングしたくなるでしょう。 TaskJuggler はプロジェクトのこういったフェーズも同様にサポートしますが、あなたの助けを必要とします。 それは追加の情報をプロジェクトファイルに記述しなければならないということです。 代わりに現在のステータスレポートとプロジェクトの現在のステータスに基づいた更新されたプロジェクト計画が手に入ります。

プロジェクトの現在のステータスを最も簡単に捉えるためには complete 属性を使います。

```
task impl "Implementation" {
  depends !spec
  effort 4w
  allocate dev1, dev2
  complete 50
}
```

これは現在までにこのタスクの 50%が完遂していることを TaskJuggler に知らせます。 complete 指定の無いタスクは予定通り進捗していると仮定します。 TaskJuggler は現在の日時をベースに完了

度合を計算します。 complete 指定はスケジュールに先だって行なわれているか遅延しているタスクに対してのみ付ける必要があります。 完了度合はスケジューリングやリソース割り当てに影響を与えないことを肝に命じなければなりません。 これは単にレポートを作成するためのものです。この指定はまた TaskJuggler にどのリソースがこのタスクで実際に作業をしたかを知らせるものでもありません。

もし現在までの作業をスケジュールに反映したければ次の節で概要を述べる booking 文を使わなければなりません。

あなたのプロジェクトがプロジェクトの現在の状況を反映しているとき、TaskJuggler は素晴らしいステータスレポートを作成することができます。 先週分の HTML 形式のレポートを、遅延していたり実施されていたり先週完了したり次週に開始されるすべてのタスクをリストして表示するには、このように指定します。

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

5.1.2. リソースの使用を記録する

初期のプロジェクト計画はタスク依存性や工数といった必要な最小限の情報を入力することで作成されます。 その後 TaskJuggler は不足するすべての情報をこの初期入力を元に計算します。 これはプロジェクトのベースラインです。 プロジェクトが進捗すると今度はリソースが完了した作業を記録することで作業が既に完了したことを追跡します。 元の計画で次のようなタスクがあると仮定しましょう：

```
task impl "Implementation" {
  depends !spec
  effort 4w
  allocate dev1, dev2
}
```

このタスクの作業開始から 1 週間経った後、２つのリソースが本当に作業の半分を完成させることができました。 これをプロジェクト計画で booking 属性を使って捉えることができます。 booking はリソース指定ですので、タスク定義ではなくリソース定義にに booking を付与しなければなりません。

```
resource dev1 "Developer 1" {
  booking impl 2005-04-11 2005-04-16 { sloppy 2 }
}
```

```
resource dev2 "Developer 2" {
  booking impl 2005-04-11 2005-04-16 { sloppy 2 }
}
```

sloppy(ずさんな) 属性で booking の正確さを定義します。 もしこれが無いか 0 ならば、booking は労働時間中の連続した労働期間だけを記載しなければなりません。 より大きな値を付与すると、期間は残業や休日の時間と重なっても良いです。 details on sloppy を参照して下さい。

リソース定義とその booking を混ぜたくなければ、booking を supplement 文で指定できます。 この supplement 文は他のファイルに置くこともできます。 web のフロントエンドを作成して開発者に完了した作業をプロジェクトプランにレポートさせる企業もあります。 レポートはデータベースに記録され、ここから TaskJuggler が取り込むファイルをさくせいします。 このようにしてプロジェクト管理者はプロジェクトの最も最新の状況を取得してこれらのデータを元にしたもっとも最近のプロジェクト計画を大きな労力なしに計算することができます。 これに興味があるのでしたら TaskJuggler web site (http://www.taskjuggler.org) のダウンロードセクションをちょっと見て下さい。

タスク定義にある effort 値より大きな値を booking に指定してもよいです。 booking が指定された effort または length に満たないにも関わらずタスクを完了したと宣言したい場合、scheduled 属性を使うことができます。

```
supplement task impl {
  actual:scheduled
}
```

注釈として国によっては被雇用者の労働時間を記録することは労働法による気息があることを述べたいと思います。 時間記録のツールを使う前に労働評議会からの承認が必要なこともあるかも知れません。

プロジェクトの実際の進捗が計画から大きく逸脱していない場合は、booking 文をもったファイルを自動的に生成することができます。

```
export "DoneWork-Week15.tji" {
  hideresource 0
  start 2005-04-11
  end 2005-04-16
  properties bookings
}
```

これで TaskJuggler のインクルードファイルを生成します。 この中には指定した期間のプロジェクト計画に関するすべての booking を含んでいます。 その後このファイルをベースラインとして用いて指定した期間に起こった事実を反映するように修正できます。 その後このファイルをプロジェクト計画にインクルードできます。

```
include "DoneWork-Week15.tji"
```

このインクルードファイルがプロジェクトのタスクとリソースを参照しているので、すべてのタスクとリソースが定義された後にインクルードしなければなりません。

To make TaskJuggler aware that you want to compute the end date based on the bookings and the effort you need to enable the projection mode for the scenario. このことはプロジェクトの冒頭のシナリオ定義で行なわれなければなりません。 もしあなたが組み込みのデフォルトシナリオしか使わないという理由でシナリオ定義をしないのであれば、シナリオ定義を加える必要があります。

```
project prj "Project" "1.0" 2005-04-01 2005-05-01 {
  scenario plan "Plan" {
    # Compute when the task will be ready based on the already
    # done work and the current date.
    projection
  }
}
```

TaskJuggler は booking 属性を持つすべてのタスクに対しするすべての作業は現在まで booking 属性を持っていると指定されます。 その後タスクの終了日をまだ残っている労力に基づいて計算します。 さらに complete 値を指定された booking に基づいて計算します。 それゆえタスクに booking 属性を指定するならば complete 値を指定してはなりません。 指定してもそれは無視され、代わりに指定した booking に基づいて計算された値で置き換えられます。

次にプロジェクトを再度スケジュールするとき、これら booking が考慮されて新しいプロジェクト計画を現在のプロジェクト状況に基づいて作成します。

プロジェクトの状況をレビューする時はいつもこのようなインクルードファイルを作成します。 それでプロジェクトの中身を要員から得た情報で動機を取り、ファイルをプロジェクト計画に加えます。

5.2. プロジェクトが進捗するにつれてプロジェクトをフリーズする

プロジェクトの計画段階で、必要最低限を指定して TaskJuggler に変化する残りの部分を計算させたいでしょう。 プロジェクトが進捗するにつれ、より多くの変数が定数に変わっていきます。 こういうことが起こった時はいつも TaskJuggler にそれを設定しなければなりません。 そうしなければ、TaskJuggler は自由にこれらの値を設定してプロジェクトの過ぎ去った部分と何ら関係の無いプロジェクト計画で終えるでしょう。 とくに十分に密なリソースを配置している時、計画は非常に変化することになるでしょう。 少々計画に変化があるだけでスケジュールは十分に違ったものになります。 TaskJuggler が使用しているスケジュール作成方法はかなり複雑でいつも最も可能な結果を導き出そうとします。 TaskJuggler はこのプロジェクトが以前にスケジュールを立てられていて変更の後に似た結果を期待されていることを知りません。 スケジュールの結果は入力された内訳にしたがっていつも正しいです。 しかし正に少々変更された後でさえかなり違ったものになり得ます。 こういう理由でスケジュールされた値が現実になりもはや自由にならないという時に TaskJuggler に知らせることが必要になるのです。

一度タスクが完了したならば allocation 文をこのタスクから取り除くのは良い考えです。 これは booking が要求された effort に完全に満たない時に間違ってリソースを割り当ててしまうのを避けるためです。 もしタスクのすべてのワークを booking で指定していたならば、すべての明記された start と end を dependency と共に取り除くことができますがそれは必須ではありません。 単にタスクに余計な指定があるだけです。 矛盾する文がある場合、エラーメッセージが TaskJuggler から通知されます。 これは例えば固定した start とこの start より前に開始する booking があるような場合です。

一般的にプロジェクトに余計な指定があるのはまったく問題無いです。 例えば、タスクに固定の start を depend と共に指定できます。 そういった指定がお互い矛盾しない限り、双方共に問題無く記述でき補助的なチェックポイントとして使うことができます。

以下にプロジェクトが進捗するにつれて凍結するための最小限の要素を含む例です。

```
project prj  "Project" "1.0" 2000-01-01-0:00-CET - 2000-03-01-0:00-CET {
  timezone "CET"
  now 2000-01-08
  scenario plan "Plan" {
    projection { strict }
  }
}

resource r "Resource"

task t "Task" {
  start 2000-01-01
  effort 10d
  allocate r
}
```

```
# Include the data from previous scheduling run.
# We assume that the exported data has been frozen.
# By importing it, we make sure they don't get changed any more.
include "CompletedWork.tji"

# Export only bookings for 1st week as resource supplements
export "CompletedWork.tji" {
  start 2000-01-01
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

now 属性はプロジェクト開始から一週間後のプロジェクト状況を説明する時のみ用います。 scenario 定義はスケジューラを予測 (projection) モードに変える時に必要です。 これは現在日以前のすべてのリソースの配置は指定されたと仮定します。 その日以後のリソースのみを配置します。 export レポートは現在日以前のすべての割当を出力します。 これにはスケジューラによって生成された割当かあるいはインクルードファイルで用意されたものがあります。 しかしスケジューラは今日以後の割当しか生成しないことを心に止めておいて下さい。 それゆえプロジェクトを空のインクルードファイルで始める場合、now をプロジェクト開始の日に設定する必要があります。 たとえインクルードファイルがレポートを作成する度に毎回書き換えられても、このファイルに行なった変更は失われません。 TaskJuggler はファイルを潜在的に追加された情報で書き換えるのに先だって現在のバージョンをいつもインクルードします。 それゆえ割当を生成したバージョンで開始しその後実際に起こったことを反映することが可能です。

Chapter 6. 言語レファレンス(未訳)

6.1. コメント

プロジェクトファイルをコメントで注釈する方法は 3 つあります。 ハッシュ記号 '#' や 2 つのスラッシュ'//' から行末までのすべてのテキストは無視します。 複数行に渡るコメントは '/*' で始まり '*/' で終わらなければなりません。

6.2. Attribute Classes(未訳)

6.2.1. DATE

DATE は `YYYY-MM-DD[-hh:mm[:ss]][-TIMEZONE]` のフォーマットで表される ISO 準拠の日付です。 時、分、秒そして `TIMEZONE` はオプションです。 指定されなければ 0 を設定します。 もし指定が無ければ、現地のタイムゾーンか既定の timezone を用います。 タイムゾーンがわからなければ、TaskJuggler は UTC(GMT) を使います。 TIMEZONE の値はタイムゾーン名か–これは曖昧になるので–`+HHMM` や `-HHMM` といった GMT への差を使えます。 詳細はソースコード (`taskjuggler/Utility.cpp`) を見て下さい。

6.2.2. DATEINTERVAL(未訳)

There are three ways to specify a date interval. The first is the most obvious. A date interval consists of a start and end DATE. The start and end dates may be separated by a hyphen character. In the second form, the end date is omitted. A 24 hour interval is assumed. The third form specifies the interval duration. In this form the start date is followed by a plus character. The plus character must be separated from the start date by a space or newline character. The plus must be followed by a number and a UNIT:

```
2006-02-18 - 2006-02-19
2006-02-18
2006-02-18 +1d
```

6.2.3. GLOBAL_ID(未訳)

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

6.2.4. ID(未訳)

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

6.2.5. INTEGER(未訳)

A number that is an integer.

6.2.6. LOGICALEXPRESSION(未訳)

A logical expression consists 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. Flag names and certain functions are supported as symbols as well. The expression is evaluated from left to right. ’~’ has a higher precedence than other operators. Use parentheses 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:

hasAssignment(ID, DATE, DATE)

: true if the task or resource currently has allocations in the specified time interval in the scenario with the specified ID.

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.

isOnCriticalPath(ID)

true if the task is on a critical path in scenario 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.

6.2.7. REAL(未訳)

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

6.2.8. SORTINGCRITERIA(未訳)

See attribute description for allowed values.

6.2.9. STRING(未訳)

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. To include single quotes in a single quoted string the single quotes have to be preceded by a backslash character to escape them. This works for double quoted strings as well.

6.2.10. TIME(未訳)

A time in the format HH:MM.

6.2.11. TIME(未訳)

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

6.2.12. UNIT(未訳)

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 differently for length, effort, and duration intervals. For length and effort only working days are counted. The number or

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.

6.2.13. WEEKDAY(未訳)

May be one of

`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 curly braces {}.

6.3. Macros(未訳)

6.3.1. Automatic macros(未訳)

Automatic macros are implicitly defined and updated in conjuction with the setting of the corresponding project property. Automatic macros always have all-lowercase names.

TaskJuggler supports the following automatic macros, which all default to the moment when TaskJuggler was invoked:

`now`	The current date and time as defined in the project header.
`projectend`	The project end date and time as defined in the project header.
`projectstart`	The project start date and time as defined in the project header.

6.3.2. User defined macros(未訳)

User defined macros are defined in the project file by using the macro property. The name of a user defined macro must have at least one uppercase character in order not to conflict with automatic macros that might be added with later versions of TaskJuggler.

Chapter 7. Property Reference(未訳)

7.1. The TJP File(未訳)

The TJP File(未訳)			
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(未訳), copyright(未訳), csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), export(未訳), flags(未訳), htmlaccountreport(未訳), htmlresourcereport(未訳), htmlstatusreport(未訳), htmltaskreport(未訳), htmlmonthlycalendar(未訳), htmlweeklycalendar(未訳), include(未訳), macro(未訳), maxeffort(未訳), limits(未訳), priority(未訳), projectid(未訳), projectids(未訳), project(未訳), rate(未訳), resource(未訳), shift(未訳), supplement(未訳), task(未訳), vacation(未訳), xmlreport(未訳)		
Context			
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}
```

```
taskreport "My Tasks"
```

7.2. account(未訳) <*id*> <*name*> [<*type*>]

account(未訳) <*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. sub-accounts inherit this type.		
Attributes	**Name**	**Type**	**Description**
	id	ID	Each account must have a unique ID.
	name	STRING	
	type	ID	The type may be `cost` or `revenue`.
Optional Attributes	`account`, credit(未訳)		
Context	The TJP File(未訳), `account`,		
Inheritable	No	**Scenario Spec.**	No
See also	csvaccountreport(未訳), htmlaccountreport(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost
```

```
  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.3. account(未訳) <*accountid*>

account(未訳) <*accountid*>			
Description	All amounts associated with the task will be credited to the specified account. The account must not be an account group.		
Attributes	**Name**	**Type**	**Description**
	accountid	ID	
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	account(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
```

```
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.4. accumulate(未訳)

accumulate(未訳)			
Description	If this attribute is specified the values in the calendar columns are accumulated over the reported interval.		
Context	csvaccountreport(未訳), htmlaccountreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
```

```
  columns index, name, weekly
}
```

7.5. allowredefinition(未訳)

allowredefinition(未訳)			
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 message which is in most other cases a real user error. So use this feature with care as real errors might go undetected. 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(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.6. allocate(未訳) <*resource*>

allocate(未訳) <*resource*>			
Description	Specify which resources should be allocated to the task. The optional attributes provide numerous ways to control which resource is used and when exactly it will be assigned to the task. Shifts and limits can be used to restrict the allocation to certain time intervals or to limit them to a certain maximum per time period.		
Attributes	**Name**	**Type**	**Description**
	resource	ID	
Optional Attributes	alternative(未訳), limits(未訳), mandatory(未訳), persistent(未訳), purge(未訳), select(未訳), shift(未訳)		
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	No

allocate(未訳) <resource>	
See also	effort(未訳)

```
project allocate "allocate" "1.0" 2003-06-05 - 2003-07-05

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

task t1 "Task 1" {
  start 2003-06-05
  # All sub-tasks inherit this allocation of r1
  allocate r1
  task t2 "Task 2" {
    effort 10d
  }
  task t3 "Task 3" {
    effort 20d
    # This task has r1 and r2 allocated
    allocate r2
  }
  task m1 "Milestone 1" {
    milestone
  }
}
```

7.7. alternative(未訳) <*resource*> [, <*resource*> ...]

alternative(未訳) <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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	select(未訳)		

```
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 tuxux or tuxia, whoever is available.
  allocate tuxus { alternative tuxia }
}
```

7.8. barlabels(未訳) <*mode*>

barlabels(未訳) <*mode*>			
Description	Specifies the contents of the Gantt chart like bars in HTML calendar columns. The default is to show load values.		
Attributes	**Name**	**Type**	**Description**
	mode	ID	See table below for possible values.
Context	htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	columns(未訳), showprojectids(未訳)		

`empty`	Do not show any values on calendar bars.
`load`	Show task or resource load on calendar bars.

7.9. baseline(未訳)

baseline(未訳)	
Description	Disables the projection mode for the scenario. This is the default for the top-level scenario. In baseline mode it is assumed that no bookings are provided for any task.
Context	scenario(未訳),

baseline(未訳)			
Inheritable	Yes	**Scenario Spec.**	Yes
See also	projection(未訳)		

7.10. booking(未訳) <*task*> <*period*> [, <*period*> ...]

booking(未訳) <*task*> <*period*> [, <*period*> ...]			
Description	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. This attribute is also used within export reports to describe the details of a scheduled project. 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.		
Attributes	**Name**	**Type**	**Description**
	task	ID	
	period	DATEINTERVAL	
Optional Attributes	overtime(未訳), sloppy(未訳)		
Context	resource(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	complete(未訳), projection(未訳), sloppy(未訳), strict(未訳)		

```
project prj "Project" "1.0" 2003-06-05 +1m {
  # 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 { strict }
  }
}

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 t1 2003-06-06 +8h { sloppy 2 }
  booking t1 2003-06-08 +4h,
             2003-06-09 +4h { sloppy 2 }
  # Book interval that extends into off-hours.
  booking t1 2003-06-11-8:00 +10h { overtime 1 }
}
```

7.11. caption(未訳) <*text*>

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

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

copyright "Bucks Beavis Inc."

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }
```

```
  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

htmltaskreport "ProjectBreakdown.html" {
  caption "This is the project breakdown"
  headline "Project Breakdown"
  columns name, start, end, daily
  # Don't hide any resource, meaning show them all.
  hideresource 0
}
```

7.12. celltext(未訳) <*text*>

celltext(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	cellurl(未訳)		

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
```

```
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.13. cellurl(未訳) <*url*>

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

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
```

```
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.14. columns(未訳) <*columnid*> [, <*columnid*> ...]

columns(未訳) <*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(未訳), cellurl(未訳), hidecelltext(未訳), hidecellurl(未訳), subtitle(未訳), subtitleurl(未訳), title(未訳), titleurl(未訳)		
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

`chart`	Use this column to get Gantt and resource charts. It's only supported by the interactive reports
`completed`	The percentage the task has been completed already. This is either the value specified by comp
`completedeffort`	The effort of a task that has already been completed.
`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 task gets the reques
`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 can contribute to t
`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 hierarchical structure of the li
`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 hierarchical structure of the list as well as
`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 by 1.
`note`	The description of the task
`pathcriticalness`	The overall criticalness of the task. It is a measure for the probability that an effort task gets sch
`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.
`remainingeffort`	The effort of a task that still needs to be done to complete the task..
`resources`	The names of the used resources
`responsibilities`	A list of all tasks indices 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 are shown in the tab
`schedule`	A detailed schedule of the allocations for the resource.
`scheduling`	The scheduling direction of the task. ASAP (As Soon As Possible) tasks are scheduled from sta
`seqno`	The task index in the order of declaration. Each time a task declaration is completed, the seque
`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 date specified by now
`statusnote`	Some comment about the current status of the task.
`total`	Total accumulated values
`utilization`	The ratio between the allocated work load of the resource and it's overall available work load.
`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:

`$${0}`	This is the original value of the table cell. This macro is useful if the user would like to extend the
`$${accountid}`	The ID of the account.
`$${resourceid}`	The ID of the resource.
`$${taskid}`	the id of the task.

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

```
index, no, hierarchindex, hierarchno, 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}.
```

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
```

```
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.15. complete(未訳) <*percent*>

complete(未訳) <*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. The completion percentage has no impact on the scheduler. It's meant for documentation purposes only. 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(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	booking(未訳)		

```
project simple "Some task" "1.0" 2005-06-06 - 2005-06-26 {
```

```
  now 2005-06-15
}

resource tux "Tux"

task t "Task" {
  start 2005-06-06
  effort 10d
  allocate tux
  # This task should have be be completed much more on Jun 15, but
  # it's only 20% done.
  complete 20
}
```

7.16. copyright(未訳) <*text*>

copyright(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	headline(未訳), caption(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

copyright "Bucks Beavis Inc."

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
```

```
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

htmltaskreport "ProjectBreakdown.html" {
  caption "This is the project breakdown"
  headline "Project Breakdown"
  columns name, start, end, daily
  # Don't hide any resource, meaning show them all.
  hideresource 0
}
```

7.17. credit(未訳) <*date*> <*description*> <*amount*>

credit(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
```

```
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.18. csvaccountreport(未訳) <*filename*>

csvaccountreport(未訳) <*filename*>	
Description	The report lists all specified account values as a comma separated list. This is useful to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.

csvaccountreport(未訳) <*filename*>			
Attributes	**Name**	**Type**	**Description**
	filename	STRING	
Optional Attributes	accumulate(未訳), columns(未訳), end(未訳), rollupaccount(未訳), hideaccount(未訳), period(未訳), scenario(未訳), separator(未訳), sortaccounts(未訳), start(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmlaccountreport(未訳)		

7.19. csvresourcereport(未訳) <*filename*>

csvresourcereport(未訳) <*filename*>			
Description	The report lists all specified resource values as a comma separated list. This is useful to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.		
Attributes	**Name**	**Type**	**Description**
	filename	STRING	
Optional Attributes	columns(未訳), end(未訳), rollupresource(未訳), hideresource(未訳), loadunit(未訳), period(未訳), scenario(未訳), separator(未訳), shorttimeformat(未訳), sortresources(未訳), start(未訳), timeformat(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmlresourcereport(未訳), resourcereport(未訳)		

7.20. csvtaskreport(未訳) <*filename*>

csvtaskreport(未訳) <*filename*>			
Description	The report lists all specified task values as a comma separated list. This is useful to export TaskJuggler data to Office Suites like OpenOffice.org or KOffice.		
Attributes	**Name**	**Type**	**Description**

csvtaskreport(未訳) <filename>			
	filename	STRING	
Optional Attributes	columns(未訳), end(未訳), rolluptask(未訳), hidetask(未訳), loadunit(未訳), period(未訳), scenario(未訳), separator(未訳), shorttimeformat(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmltaskreport(未訳), taskreport(未訳)		

7.21. currency(未訳) <text>

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

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
```

```
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.22. currencyformat(未訳) *<negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>*

currencyformat(未訳) *<negativeprefix> <negativesuffix> <thousandseparator> <fractionseparator> <fractiondigits>*			
Description	These values specify the default format used for all currency values. The `negativeprefix` and `negativesuffix` strings enclose negative currency values. The `thousandseparator` can be used to make large numbers more readable. The `fractionseparator` separates the fractional part from the rest. `fractiondigits` specifies how many fractional digits should be shown at a maximum.		
Attributes	**Name**	**Type**	**Description**
	negativeprefix	STRING	
	negativesuffix	STRING	

currencyformat(未訳) *<negativeprefix>* *<negativesuffix>* *<thousandseparator>* *<fractionseparator>* *<fractiondigits>*			
	thousandseparator	STRING	
	fractionseparator	STRING	
	fractiondigits	INTEGER	
Context	project(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	currency(未訳)		

```
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
}
```

7.23. dailymax(未訳) *<value>* *<unit>*

dailymax(未訳) *<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(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	monthlymax(未訳), weeklymax(未訳)		

```
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" {
  # Limit the usage of this resource to a maximum of 2 hours per day,
  # 6 hours per week and 2.5 days per month.
  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 }
  }
}
```

7.24. dailyworkinghours(未訳) <*hours*>

dailyworkinghours(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	yearlyworkingdays(未訳)		

```
project prj "Example Project" "1.0" 2000-01-01 - 2000-03-09 {
  # The following attributes are all optional. They illustrate the
  # default values. These attributes are only needed if you want to
  # specify different values than those listed below.
  dailyworkinghours 8
  yearlyworkingdays 260.714
  timingresolution 60min
  timeformat "%Y-%m-%d %H:%M"
  shorttimeformat "%H:%M"
  currencyformat "(" ")" "," "." 0
  weekstartsmonday
    workinghours mon - fri 9:00 - 12:00, 13:00 - 18:00
    workinghours sat, sun off
  scenario plan "Plan" {
  }
}

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

7.25. drawemptycontainersastasks(未訳)

drawemptycontainersastasks(未訳)			
Description	If set, every container that has no subtask visible will be drawn as a regular task. The default behaviour is to draw containers as containers even if all their subtasks are hidden.		
Context	project(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.26. depends(未訳) <*task*> [, <*task*> ...]

depends(未訳) <*task*> [, <*task*> ...]

depends(未訳) <`task`> [, <`task`> ...]			
Description	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. By using the 'depends' attribute, the scheduling policy is automatically set to `asap`. If both `depends` and `precedes` are used, the last policy counts.		
Attributes	**Name**	**Type**	**Description**
	`task`	ID	
Optional Attributes	gapduration(未訳), gaplength(未訳)		
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	scheduling(未訳), precedes(未訳)		

```
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
}
```

7.27. disabled(未訳)

disabled(未訳)			
Description	Disables the scenario for scheduling.		
Context	scenario(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	enabled(未訳)		

```
project prj "Example" "1.0" 2005-05-29 - 2005-07-01 {
  scenario plan "Planned Scenario" {
    scenario actual "Actual Scenario"
    scenario test "Test Scenario" {
      disabled
    }
  }
}

task t "Task" {
  start 2005-05-29
  actual:start 2005-06-03
  test:start 2005-06-07
}
```

7.28. duration(未訳) <*value*> <*unit*>

duration(未訳) <*value*> <*unit*>			
Description	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.		
Attributes	**Name**	**Type**	**Description**
	value	REAL	
	unit	UNIT	

duration(未訳) <value> <unit>			
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	effort(未訳), length(未訳)		

```
project duration "Duration Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task t "Enclosing" {
  start 2005-06-06
  task durationTask "Duration Task" {
    # This task is 10 calendar days long.
    duration 10d
  }

  task intervalTask "Interval Task" {
    # This task is similar to the durationTask. Instead of a start
    # date and a duration it has a fixed start and end date.
    end 2005-06-17
  }

  task lengthTask "Length Task" {
    # This task 10 working days long. So about 12 calendar days.
    length 10d
  }

  task effortTask "Effort Task" {
    effort 10d
    allocate tux
  }
}
```

7.29. efficiency(未訳) <value>

efficiency(未訳) <value>

efficiency(未訳) <*value*>			
Description	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.		
Attributes	**Name**	**Type**	**Description**
	value	REAL	
Context	resource(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	load(未訳)		

```
project prj "Resource Efficiency Example" "1.0" 2005-07-21 - 2005-07-22

# A team of 5 people. They can only be assigned en block. Either all
# or nobody works.
resource tuxies "Tuxies" { efficiency 5.0 }

# A hard-working guy
resource tux1 "Tux 1" { efficiency 1.2 }

# And a lazy one
resource tux2 "Tux 2" { efficiency 0.9 }

# And a thing that cannot do any work
resource confRoom "Conference Room" { efficiency 0 }

task t "An important date" {
  start 2005-07-21
}
```

7.30. effort(未訳) <*value*> <*unit*>

effort(未訳) <*value*> <*unit*>

effort(未訳) <*value*> <*unit*>			
Description	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. 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. Tasks may not have subtasks if this attribute is used.		
Attributes	**Name**	**Type**	**Description**
	value	REAL	
	unit	UNIT	
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	duration(未訳), length(未訳)		

```
project duration "Duration Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task t "Enclosing" {
  start 2005-06-06
  task durationTask "Duration Task" {
    # This task is 10 calendar days long.
    duration 10d
  }

  task intervalTask "Interval Task" {
    # This task is similar to the durationTask. Instead of a start
    # date and a duration it has a fixed start and end date.
    end 2005-06-17
  }

  task lengthTask "Length Task" {
    # This task 10 working days long. So about 12 calendar days.
    length 10d
  }

  task effortTask "Effort Task" {
    effort 10d
    allocate tux
  }
```

```
}
```

7.31. enabled(未訳)

enabled(未訳)			
Description	Enable the scenario for scheduling. This is the default for the top-level scenario.		
Context	scenario(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	disabled(未訳)		

7.32. end(未訳) <*date*>

end(未訳) <*date*>			
Description	Specifies the end date of the report. In task reports only tasks that start before this end date are listed.		
Attributes	**Name**	**Type**	**Description**
	date	DATE	
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), export(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	period(未訳), start(未訳)		

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

resource r "Resource"

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

```
  effort 10d
  allocate r
}

# Export the project as fully scheduled project.
export "FullProject.tjp" {
  taskattributes all
  hideresource 0
}

# Export only bookings for 1st week as resource supplements
export "Week1Bookings.tji" {
  start 2000-01-01
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

7.33. end(未訳) <*date*>

end(未訳) <*date*>			
Description	The end date of the task. When specified for the top-level (default) scenario this attributes also implicitly sets the scheduling policy of the tasks to `alap`.		
Attributes	**Name**	**Type**	**Description**
	date	DATE	
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	period(未訳), start(未訳), maxend(未訳), minend(未訳), scheduling(未訳), endbuffer(未訳)		

```
project duration "Duration Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task t "Enclosing" {
  start 2005-06-06
  task durationTask "Duration Task" {
    # This task is 10 calendar days long.
    duration 10d
  }
```

```
  task intervalTask "Interval Task" {
    # This task is similar to the durationTask. Instead of a start
    # date and a duration it has a fixed start and end date.
    end 2005-06-17
  }

  task lengthTask "Length Task" {
    # This task 10 working days long. So about 12 calendar days.
    length 10d
  }

  task effortTask "Effort Task" {
    effort 10d
    allocate tux
  }
}
```

7.34. endbuffer(未訳) <*percent*>

endbuffer(未訳) <*percent*>			
Description	Specifies how much slack time you expect to have at the end of the task. This has no impact on the scheduling of the process. This information is for documentation purposes only.		
Attributes	**Name**	**Type**	**Description**
	percent	REAL	Percent slack of the overall effort, duration or length of the task.
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	duration(未訳), effort(未訳), length(未訳), startbuffer(未訳)		

```
project simple "Simple Project" "$Id" 2000-01-01 - 2000-01-20

resource tux1 "Tux1"

task t1 "Task1" {
  start 2000-01-01
  length 10d
```

```
  # 20% of the working time of this task are marked as buffer at the
  # beginning.
  startbuffer 20
  # An additional 10% of the working time of this task are marked as
  # buffer at the end.
  endbuffer 10.0
  allocate tux1
}

# Generate a report that lists the start end end dates for the
# buffers.
htmltaskreport "Buffer.html" {
  columns no, name, start, startbufferend, endbufferstart, end,
  startbuffer, endbuffer, duration, effort, daily
  hideresource 0
}
```

7.35. endcredit(未訳) <*amount*>

endcredit(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	startcredit(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}
```

```
task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.36. export(未訳) <*filename*>

export(未訳) <*filename*>

export(未訳) <`filename`>			
Description	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 contents 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. 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. 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.		
Attributes	**Name**	**Type**	**Description**
	`filename`	STRING	
Optional Attributes	end(未訳), hideresource(未訳), hidetask(未訳), properties(未訳), rollupresource(未訳), rolluptask(未訳), period(未訳), scenarios(未訳), start(未訳), taskattributes(未訳), taskroot(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	include(未訳)		

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

resource r "Resource"

task t "Task" {
  start 2000-01-01
  effort 10d
  allocate r
}

# Export the project as fully scheduled project.
export "FullProject.tjp" {
  taskattributes all
  hideresource 0
}

# Export only bookings for 1st week as resource supplements
export "Week1Bookings.tji" {
  start 2000-01-01
```

```
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

7.37. extend(未訳) <*property*>

extend(未訳) <*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 `text` or `reference` 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 `task`, `resource` or `account`
Optional Attributes	inherit(未訳)		
Context	project(未訳),		
Inheritable	No	**Scenario Spec.**	No

<type> <id> <name>

type: Specifies the type of the user-defined attribute. `text` is a simple text attribute. `reference` is an URL to anoth
id: An user-defined ID that is unique within the used-defined task attributes. To avoid conflicts with future built-in
name: A short description of the attribute. It will be used as default column header in reports.

```
project ca "Custom Attributes" "1.0" 2003-05-28 - 2003-06-28 {
  extend task {
    reference MyLink "My Link"
    text MyText "My Text" { inherit }
  }
}

task t "Task" {
  start 2003-05-28
  milestone
```

```
  MyLink "http://www.taskjuggler.org" { label "TJ Web" }
  MyText "TaskJuggler is great!"
}
```

7.38. flags(未訳) <*flag*> [, <*flag*> ...]

flags(未訳) <*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(未訳), task(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	flags(未訳)		

```
project prj "Flags Example" "1.0" 2005-07-21 - 2005-08-26

# Declare the flag to mark important tasks
flags important

task items "Project breakdown" {
  start 2005-07-22

  task plan "Plan work" {
    length 3d
    flags important
  }

  task implementation "Implement work" {
    length 5d
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    flags important
  }
}
```

```
taskreport "My Tasks" {
  # Show only the important tasks
  hidetask ~important
  # Turn treemode off so parent tasks are not automatically included.
  sorttasks nameup
}
```

7.39. flags(未訳) <*flag*> [, <*flag*> ...]

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

```
project prj "Flags Example" "1.0" 2005-07-21 - 2005-08-26

# Declare the flag to mark important tasks
flags important

task items "Project breakdown" {
  start 2005-07-22

  task plan "Plan work" {
    length 3d
    flags important
  }

  task implementation "Implement work" {
    length 5d
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
```

```
    flags important
  }
}

taskreport "My Tasks" {
  # Show only the important tasks
  hidetask ~important
  # Turn treemode off so parent tasks are not automatically included.
  sorttasks nameup
}
```

7.40. gapduration(未訳) <*value*> <*unit*>

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

```
project prj "Example Project" "1.0" 2005-05-29 - 2005-07-01

task t1 "Task 1" {
  start 2005-05-29
}

task t2 "Task 2" {
  # starts 5 calendar days after t1
  depends !t1 { gapduration 5d }
}

task t3 "Task 3" {
  # starts 5 working days after t1
  depends !t1 { gaplength 5d }
}
```

7.41. gaplength(未訳) *<value>* *<unit>*

gaplength(未訳) *<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 calendar 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(未訳), precedes(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	gapduration(未訳), length(未訳)		

```
project prj "Example Project" "1.0" 2005-05-29 - 2005-07-01

task t1 "Task 1" {
  start 2005-05-29
}

task t2 "Task 2" {
  # starts 5 calendar days after t1
  depends !t1 { gapduration 5d }
}

task t3 "Task 3" {
  # starts 5 working days after t1
  depends !t1 { gaplength 5d }
}
```

7.42. headline(未訳) *<text>*

headline(未訳) <*text*>			
Description	Specifies the headline for a report.		
Attributes	**Name**	**Type**	**Description**
	text	STRING	
Context	htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmlstatusreport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	caption(未訳), copyright(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

copyright "Bucks Beavis Inc."

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

htmltaskreport "ProjectBreakdown.html" {
  caption "This is the project breakdown"
  headline "Project Breakdown"
  columns name, start, end, daily
  # Don't hide any resource, meaning show them all.
  hideresource 0
}
```

7.43. hideaccount(未訳) <*logicalexpression*>

hideaccount(未訳) <*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(未訳), htmlaccountreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.44. hidecelltext(未訳) <*expression*>

hidecelltext(未訳) <*expression*>			
Description	If the expression is true, the cell will be empty.		
Attributes	**Name**	**Type**	**Description**
	expression	LOGICALEXPRESSION	
Context	columns(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	hidecellurl(未訳)		

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}
```

```
# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.45. hidecellurl(未訳) <*expression*>

hidecellurl(未訳) <*expression*>	
Description	If the expression is true, no URL will be attached to the cell contents.

hidecellurl(未訳) <*expression*>			
Attributes	**Name**	**Type**	**Description**
	expression	LOGICALEXPRESSION	
Context	columns(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	hidecelltext(未訳)		

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
```

```
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.46. inherit(未訳)

inherit(未訳)			
Description	If this flag is present a user defined attribute gets inherited by child properties when specified for a parent.		
Context	extend(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.47. hideresource(未訳) <*logicalexpression*>

hideresource(未訳) <*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(未訳), export(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), icalreport(未訳), resourcereport(未訳), taskreport(未訳), xmlreport(未訳),		

hideresource(未訳) <*logicalexpression*>			
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

copyright "Bucks Beavis Inc."

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

htmltaskreport "ProjectBreakdown.html" {
  caption "This is the project breakdown"
  headline "Project Breakdown"
  columns name, start, end, daily
  # Don't hide any resource, meaning show them all.
  hideresource 0
}
```

7.48. hidetask(未訳) <*logicalexpression*>

hidetask(未訳) <*logicalexpression*>

hidetask(未訳) <*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(未訳), export(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), icalreport(未訳), resourcereport(未訳), taskreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"
resource tuxia "Tuxia"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    task phase1 "Phase 1" {
      effort 5d
      allocate tuxia
    }
    task phase2 "Phase 2" {
      effort 2d
      allocate tux
    }
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

taskreport "Project Breakdown" {
  columns start, end, effort
  # Open only the first level of tasks
```

```
  rolluptask treelevel() > 1
}

resourcereport "Resource Allocations" {
  columns id, effort
  # We only want to see the tasks with real work (without parents),
  # sorted by name
  sorttasks nameup
  hidetask ~isleaf()
}
```

7.49. htmlaccountreport(未訳) <*file*>

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

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
```

```
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.50. htmlmonthlycalendar(未訳) <*file*>

htmlmonthlycalendar(未訳) <*file*>			
Description	Generates a calendar like HTML report with one column for each month.		
Attributes	**Name**	**Type**	**Description**
	file	STRING	
Optional Attributes	barlabels(未訳), caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rawhead(未訳), rawstylesheet(未訳), rawtail(未訳), rollupresource(未訳), rolluptask(未訳), scenarios(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳), weekdays(未訳)		

htmlmonthlycalendar(未訳) *<file>*			
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmlaccountreport(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlstatusreport(未訳), htmlweeklycalendar(未訳)		

7.51. htmlresourcereport(未訳) *<file>*

htmlresourcereport(未訳) *<file>*			
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**
	file	STRING	
Optional Attributes	barlabels(未訳), caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rawhead(未訳), rawstylesheet(未訳), rawtail(未訳), rollupresource(未訳), rolluptask(未訳), scenarios(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	csvresourcereport(未訳), htmlaccountreport(未訳), htmltaskreport(未訳), resourcereport(未訳)		

7.52. htmlstatusreport(未訳) *<file>*

htmlstatusreport(未訳) *<file>*			
Description	Generates a HTML status report. The report consists 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	

htmlstatusreport(未訳) <file>			
Optional Attributes	headline(未訳), caption(未訳), rawhead(未訳), rawstylesheet(未訳), rawtail(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmlaccountreport(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlmonthlycalendar(未訳), htmlweeklycalendar(未訳)		

7.53. htmltaskreport(未訳) <file>

htmltaskreport(未訳) <file>			
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**
	file	STRING	
Optional Attributes	barlabels(未訳), caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rawhead(未訳), rawstylesheet(未訳), rawtail(未訳), rollupresource(未訳), rolluptask(未訳), scenarios(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	csvtaskreport(未訳), htmlaccountreport(未訳), htmlresourcereport(未訳), taskreport(未訳)		

7.54. htmlweeklycalendar(未訳) <file>

htmlweeklycalendar(未訳) <file>			
Description	Generates a calendar like HTML report with one row for each week.		
Attributes	**Name**	**Type**	**Description**
	file	STRING	

htmlweeklycalendar(未訳) *<file>*			
Optional Attributes	barlabels(未訳), caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rawhead(未訳), rawstylesheet(未訳), rawtail(未訳), resourcereport(未訳), rollupresource(未訳), rolluptask(未訳), scenarios(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳), weekdays(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlstatusreport(未訳)		

7.55. icalreport(未訳) *<file>*

icalreport(未訳) *<file>*			
Description	Generates an ICal report according to RFC2445. This is standardized format used by many calendar applications such as KOrganizer. The filename should have a `.ics` extension.		
Attributes	**Name**	**Type**	**Description**
	file	STRING	
Optional Attributes	hideresource(未訳), hidetask(未訳), rollupresource(未訳), rolluptask(未訳), scenarios(未訳)		
Context			
Inheritable	No	**Scenario Spec.**	No

```
project prj "ICal Export Demo" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"
resource tuxia "Tuxia"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    task phase1 "Phase 1" {
      effort 5d
      allocate tuxia
```

```
    }
    task phase2 "Phase 2" {
      effort 2d
      allocate tux
    }
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

icalreport "Tux-TODO-List.ics" {
  # Only export tasks that tux is assigned to
  hidetask ~isDutyOf(tux, plan)
}
```

7.56. include(未訳) <*file*>

include(未訳) <*file*>			
Description	Includes the specified file name as if its contents would be written instead of the include property. The only exception is the `include` statement itself. When the included files contains other include statements or report definitions, the filenames are relative to file where they are defined in. `include` commands can be used in the project header, at 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 call that is defined within the included file.		
Attributes	**Name**	**Type**	**Description**
	file	STRING	
Optional Attributes	taskprefix(未訳)		
Context	The TJP File(未訳), project(未訳),		
Inheritable	No	**Scenario Spec.**	No

include(未訳) *<file>*	
See also	export(未訳)

```
project yourId "Your Project" "1.0" 2005-04-05 - 2005-05-01

task main "Main task" {
}

include "Include2.tji" { taskprefix main }
```

7.57. journalentry(未訳) *<date>* *<text>*

journalentry(未訳) *<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(未訳), resource(未訳), task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	note(未訳), statusnote(未訳)		

```
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
}

taskreport "Tasks"
```

7.58. label(未訳) <*text*>

label(未訳) <*text*>			
Description	Specifies the text for the URL in HTML reports. If no label is specified, the URL will be displayed. If a label has been specified, the URL will not be shown.		
Attributes	**Name**	**Type**	**Description**
	text	STRING	
Context	reference(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project ca "Custom Attributes" "1.0" 2003-05-28 - 2003-06-28 {
  extend task {
    reference MyLink "My Link"
    text MyText "My Text" { inherit }
  }
}

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

7.59. length(未訳) <*value*> <*unit*>

length(未訳) <*value*> <*unit*>			
Description	Specifies the time the task occupies the resources. This is working time, not calendar 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 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(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	duration(未訳), effort(未訳)		

```
project duration "Duration Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task t "Enclosing" {
  start 2005-06-06
  task durationTask "Duration Task" {
    # This task is 10 calendar days long.
    duration 10d
  }

  task intervalTask "Interval Task" {
    # This task is similar to the durationTask. Instead of a start
    # date and a duration it has a fixed start and end date.
    end 2005-06-17
  }

  task lengthTask "Length Task" {
    # This task 10 working days long. So about 12 calendar days.
    length 10d
  }

  task effortTask "Effort Task" {
    effort 10d
    allocate tux
  }
}
```

7.60. limits(未訳)

limits(未訳)			
Description	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. 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.		
Optional Attributes	dailymax(未訳), weeklymax(未訳), monthlymax(未訳)		
Context	The TJP File(未訳), allocate(未訳), resource(未訳),		
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" {
  # Limit the usage of this resource to a maximum of 2 hours per day,
  # 6 hours per week and 2.5 days per month.
  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 }
  }
}
```

7.61. load(未訳) <*factor*>

load(未訳) <*factor*>			
Description	This property has been replaced by limits. The further usage of load is strongly discouraged. It will be dropped from future versions of TaskJuggler. 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.		
Attributes	**Name**	**Type**	**Description**
	factor	REAL	
Context			
Inheritable	No	**Scenario Spec.**	No
See also	workinghours(未訳), vacation(未訳)		

7.62. loadunit(未訳) <*unit*>

loadunit(未訳) <*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	csvresourcereport(未訳), csvtaskreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

loadunit(未訳) <*unit*>	
See also	dailyworkinghours(未訳), yearlyworkingdays(未訳)

`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.

7.63. macro(未訳) <*id*>

macro(未訳) <*id*>			
Description	Defines a text fragment that can later be inserted by using the specified ID. See the description of the `include` statement for details on how to use a macro immediately after including the file where the macro has been defined in.		
Attributes	**Name**	**Type**	**Description**
	id	ID	
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also			

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 question mark. 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.
```

7.64. mandatory(未訳)

mandatory(未訳)			
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(未訳),		
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 }
}
```

7.65. maxeffort(未訳) <*workingdays*>

maxeffort(未訳) <*workingdays*>			
Description	This property has been replaced by limits. The further usage of maxeffort is strongly discouraged. It will be dropped from future versions of TaskJuggler. 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.		
Attributes	**Name**	**Type**	**Description**
	workingdays	REAL	
Context	The TJP File(未訳), resource(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	dailyworkinghours(未訳), workinghours(未訳)		

7.66. maxend(未訳) <*date*>

maxend(未訳) <*date*>			
Description	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.		
Attributes	**Name**	**Type**	**Description**
	date	DATE	
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	maxstart(未訳), minend(未訳), minstart(未訳)		

```
project minmax "Min Max Example" "1.0" 2005-06-06 - 2005-06-26
```

```
task items "Project breakdown" {
  start 2005-06-07

  task plan "Plan work" {
  note "Some more information about this task."
    # Set acceptable interval for task start
    minstart 2005-06-06
    maxstart 2005-06-08
    length 3d
    # Set acceptable interval for task end
    minend 2005-06-09
    maxend 2005-06-11
  }
}
```

7.67. maxpaths(未訳) <*paths*>

maxpaths(未訳) <*paths*>			
Description	The scheduler contains a critical path detector. Detecting a critical path requires the analysis of every possible path between 2 end tasks. The number of paths that have to be checked can grow exponentially with the number of tasks and their dependencies. By default, only 10 million paths will be checked. With this parameter the user can define how many paths should be checked. Using 0 as value will trigger an exhaustive search under all circumstances. Scheduling times of hours if not days are certainly possible for large projects.		
Attributes	**Name**	**Type**	**Description**
	paths	INTEGER	
Context	scenario(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	minslackrate(未訳)		

7.68. maxstart(未訳) <*date*>

maxstart(未訳) <*date*>

maxstart(未訳) *<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(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	maxend(未訳), minend(未訳), minstart(未訳)		

```
project minmax "Min Max Example" "1.0" 2005-06-06 - 2005-06-26

task items "Project breakdown" {
  start 2005-06-07

  task plan "Plan work" {
  note "Some more information about this task."
    # Set acceptable interval for task start
    minstart 2005-06-06
    maxstart 2005-06-08
    length 3d
    # Set acceptable interval for task end
    minend 2005-06-09
    maxend 2005-06-11
  }
}
```

7.69. minend(未訳) *<date>*

minend(未訳) *<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(未訳),		

minend(未訳) <*date*>			
Inheritable	Yes	**Scenario Spec.**	Yes
See also	maxend(未訳), maxstart(未訳), minstart(未訳)		

```
project minmax "Min Max Example" "1.0" 2005-06-06 - 2005-06-26

task items "Project breakdown" {
  start 2005-06-07

  task plan "Plan work" {
  note "Some more information about this task."
    # Set acceptable interval for task start
    minstart 2005-06-06
    maxstart 2005-06-08
    length 3d
    # Set acceptable interval for task end
    minend 2005-06-09
    maxend 2005-06-11
  }
}
```

7.70. minslackrate(未訳) <*rate*>

minslackrate(未訳) <*rate*>	
Description	Specifies the minimum percentage of slack a task path must have before it is marked as critical. A path is any list of explicitely or implicitely connected tasks measured from first task to last task. The slack is the time between start of the first task and end of the last task that is not covered by any task of the path. The default value is 5%. Larger values in combination with a project that uses lots of inherited dependencies and long dependency pathes can result in very long scheduling times. The more slack you require, the more pathes have to be searched till the end. For larger projects an increase of 5% can turn a 10 second scheduling run into a 1 hour or more scheduling run. If you need larger slack rate values, avoid the use of inherited dependencies. A value of 0 turns off the critical path detector.

minslackrate(未訳) <*rate*>			
Attributes	**Name**	**Type**	**Description**
	rate	REAL	
Context	scenario(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	maxpaths(未訳)		

```
project prj "Critical Path Example" "1.0" 2006-08-22 +1m {
  scenario plan "Planned Scenario" {
    # All pathes with less than 15% slack should be marked as
    # critical.
    minslackrate 15.0
  }
}

task t1 "Task 1" {
  start 2006-08-22
  duration 2d
}

task t2 "Task 2" {
  depends t1 { gaplength 2d }
  duration 3d
}

task t3 "Task 3" {
  depends t1 { gaplength 1d }
  duration 4d
}

taskreport "Tasks" {
  columns no, name, chart
}

htmltaskreport "CriticalTasks.html" {
  # Generate a list of all tasks that are on a critical path.
  hidetask ~isOnCriticalPath(plan)
}
```

7.71. minstart(未訳) <*date*>

minstart(未訳) <*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(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	maxend(未訳), maxstart(未訳), minend(未訳)		

```
project minmax "Min Max Example" "1.0" 2005-06-06 - 2005-06-26

task items "Project breakdown" {
  start 2005-06-07

  task plan "Plan work" {
  note "Some more information about this task."
    # Set acceptable interval for task start
    minstart 2005-06-06
    maxstart 2005-06-08
    length 3d
    # Set acceptable interval for task end
    minend 2005-06-09
    maxend 2005-06-11
  }
}
```

7.72. milestone(未訳)

milestone(未訳)

milestone(未訳)			
Description	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. A task that only has a start or an end specification and no duration specification or sub tasks, will be recognized as milestone automatically.		
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project prj "Milestone demo" "1.0" 2005-07-15 - 2005-08-01

task project_start "Project Start" {
  start 2005-07-15
  milestone
}

task deadline "Important Deadline" {
  start 2005-07-20
  # A task with only a start or end date and no duration specification
  # is automatically assumed to be a milestone.
}
```

7.73. note(未訳) <*text*>

note(未訳) <*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**
	text	STRING	
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	journalentry(未訳), statusnote(未訳)		

```
project minmax "Min Max Example" "1.0" 2005-06-06 - 2005-06-26
```

```
task items "Project breakdown" {
  start 2005-06-07

  task plan "Plan work" {
  note "Some more information about this task."
    # Set acceptable interval for task start
    minstart 2005-06-06
    maxstart 2005-06-08
    length 3d
    # Set acceptable interval for task end
    minend 2005-06-09
    maxend 2005-06-11
  }
}
```

7.74. monthlymax(未訳) <*value*> <*unit*>

monthlymax(未訳) <*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(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	dailymax(未訳), weeklymax(未訳)		

```
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" {
  # Limit the usage of this resource to a maximum of 2 hours per day,
  # 6 hours per week and 2.5 days per month.
  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 }
  }
}
```

7.75. now(未訳) <*date*>

now(未訳) <*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(未訳),		
Inheritable	Yes	**Scenario Spec.**	No

```
project simple "Some task" "1.0" 2005-06-06 - 2005-06-26 {
  now 2005-06-15
}

resource tux "Tux"

task t "Task" {
  start 2005-06-06
  effort 10d
  allocate tux
  # This task should have be be completed much more on Jun 15, but
  # it's only 20% done.
```

```
  complete 20
}
```

7.76. numberformat(未訳) <*negativeprefix*> <*negativesuffix*> <*thousandseparator*> <*fractionseparator*> <*fractiondigits*>

numberformat(未訳) <*negativeprefix*> <*negativesuffix*> <*thousandseparator*> <*fractionseparator*> <*fractiondigits*>			
Description	These values specify the default format used for all numerical real values. The `negativeprefix` and `negativesuffix` strings enclose negative currency values. The `thousandseparator` can be used to make large numbers more readable. The `fractionseparator` separates the fractional part from the rest. `fractiondigits` specifies how many fractional digits should be shown at a maximum.		
Attributes	**Name**	**Type**	**Description**
	negativeprefix	STRING	
	negativesuffix	STRING	
	thousandseparator	STRING	
	fractionseparator	STRING	
	fractiondigits	INTEGER	
Context	project(未訳),		
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
}
```

7.77. overtime(未訳) <*value*>

overtime(未訳) <*value*>			
Description	This attribute enables bookings to override working hours and vacations.		
Attributes	**Name**	**Type**	**Description**
	value	INTEGER	Number between 0 and 2. See table below.
Context	booking(未訳),		
Inheritable	No	**Scenario Spec.**	No

`overtime 0:` You can only book available working hours. (Default)
`overtime 1:` You can book off-hours as well.
`overtime 2:` You can book normal working hours, off-hours and vacations.

```
project prj "Project" "1.0" 2003-06-05 +1m {
  # 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 { strict }
  }
}

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 t1 2003-06-06 +8h { sloppy 2 }
  booking t1 2003-06-08 +4h,
             2003-06-09 +4h { sloppy 2 }
  # Book interval that extends into off-hours.
  booking t1 2003-06-11-8:00 +10h { overtime 1 }
}
```

7.78. period(未訳)

period(未訳)			
Description	This property is a shortcut for setting the start and end property at the same time. In contrast to using these, it does not change the scheduling direction.		
Context	task(未訳),		
Inheritable	yes	**Scenario Spec.**	No
See also	end(未訳), start(未訳)		

```
project prj "Period Project" "1.0" 2006-09-24 +3m {
 now 2006-10-02
}

task items "Project breakdown" {
  start ${projectstart}

  task plan "Plan work" {
    period 2006-10-01 +2w
  }
}

taskreport "My Tasks" {
 period ${now} +1w
}
```

7.79. period(未訳)

period(未訳)	
Description	This property is a shortcut for setting the start and end property at the same time.

period(未訳)			
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), export(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	no	**Scenario Spec.**	No
See also	end(未訳), start(未訳)		

```
project prj "Period Project" "1.0" 2006-09-24 +3m {
 now 2006-10-02
}

task items "Project breakdown" {
  start ${projectstart}

  task plan "Plan work" {
    period 2006-10-01 +2w
  }
}

taskreport "My Tasks" {
 period ${now} +1w
}
```

7.80. persistent(未訳)

persistent(未訳)			
Description	Specifies that once a resource is picked from the list of alternatives this resource is used for the whole task. This is useful when several alternative resources have been specified. Normally 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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	alternative(未訳)		

```
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 }
}
```

7.81. priority(未訳) <*value*>

priority(未訳) <*value*>			
Description	Specifies a priority between 1 and 1000. A task with higher priority is more likely to get the requested resources. Don't confuse the priority of a tasks with the importance or urgency of a task. It only increases the chances that the tasks gets the requested resources. It does not mean that the task happens earlier, though that is usually the effect you will see. It also does not have any effect on tasks that don't have any resources assigned (e.g. milestones). This attribute is inherited by subtasks if specified prior to the definition of the subtask.		
Attributes	**Name**	**Type**	**Description**
	value	INTEGER	
Context	The TJP File(未訳), task(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project prj "Priority Demo" "1.0" 2005-07-15 - 2005-10-01

resource tux "Tux"

task items "Project breakdown" {
  start 2005-07-15

  task coolStuff "Do some cool stuff" {
    start 2005-08-01
    effort 10d
    priority 800
```

```
    allocate tux
  }

  task otherStuff "Other not so important stuff" {
    start 2005-08-01
    effort 20d
    priority 500
    allocate tux
  }

  task maintenance "Maintenance work" {
    # This is a fallback task. Whenever tux is not doing something
    # else he is allocated to this task.
    duration 2m
    priority 300
    allocate tux
  }
}
```

7.82. precedes(未訳) <*task*> [, <*task*> ...]

precedes(未訳) <*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 `alap`. If both `depends` and `precedes` are used within a task, the last policy counts.		
Attributes	**Name**	**Type**	**Description**
	task	ID	
Optional Attributes	gapduration(未訳), gaplength(未訳)		
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	depends(未訳), scheduling(未訳)		

```
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
}
```

7.83. project(未訳) *<id> <name> <version> <period>*

project(未訳) *<id> <name> <version> <period>*			
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.
	period	DATEINTERVAL	The expected interval of the project. It may be larger than necessary, but it needs to be large enough to fit all tasks.

project(未訳) *<id> <name> <version> <period>*			
Optional Attributes	allowredefinition(未訳), currencyformat(未訳), currency(未訳), dailyworkinghours(未訳), drawemptycontainersastasks(未訳), extend(未訳), include(未訳), journalentry(未訳), now(未訳), numberformat(未訳), scenario(未訳), shorttimeformat(未訳), timeformat(未訳), timezone(未訳), timingresolution(未訳), weekstartsmonday(未訳), weekstartssunday(未訳), workinghours(未訳), yearlyworkingdays(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

taskreport "My Tasks"
```

7.84. projectid(未訳) *<id>*

projectid(未訳) *<id>*

projectid(未訳) <*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(未訳), task(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	project(未訳), projectids(未訳)		

```
project mainID "ProjectIDs example" "1.0" 2006-08-22 +1m


task t1 "Task 1" {
  start 2006-08-22
 # This task has project ID "mainID"
}

projectid prj1
projectids prj2

task t2 "Task 2" {
  start 2006-08-22
 # This task has now project ID "prj1"
}


task t3 "Task 3" {
  start 2006-08-22
  projectid prj2
 # This task has now project ID "prj2"
}
```

7.85. projectids(未訳) <*projectid*> [, <*projectid*> ...]

projectids(未訳) <*projectid*> [, <*projectid*> ...]

projectids(未訳) *<projectid>* [, *<projectid>* ...]			
Description	Declares a list of project IDs.		
Attributes	**Name**	**Type**	**Description**
	projectid	ID	
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	project(未訳), projectid(未訳)		

```
project mainID "ProjectIDs example" "1.0" 2006-08-22 +1m


task t1 "Task 1" {
  start 2006-08-22
 # This task has project ID "mainID"
}

projectid prj1
projectids prj2

task t2 "Task 2" {
  start 2006-08-22
 # This task has now project ID "prj1"
}


task t3 "Task 3" {
  start 2006-08-22
  projectid prj2
 # This task has now project ID "prj2"
}
```

7.86. projection(未訳)

projection(未訳)

projection(未訳)			
Description	Enables the projection mode for the scenario. All tasks will be scheduled taking the manual bookings into account. The tasks will be extended by scheduling new bookings starting with the current date until the specified effort, length or duration has been reached. In sloppy mode tasks with no bookings will be filled from the original start. In strict mode all tasks will be filled starting with the current date. No bookings will be added prior to the current date.		
Optional Attributes	sloppy(未訳), strict(未訳)		
Context	scenario(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	baseline(未訳), booking(未訳)		

```
project prj "Project" "1.0" 2003-06-05 +1m {
  # 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 { strict }
  }
}

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 t1 2003-06-06 +8h { sloppy 2 }
  booking t1 2003-06-08 +4h,
             2003-06-09 +4h { sloppy 2 }
  # Book interval that extends into off-hours.
  booking t1 2003-06-11-8:00 +10h { overtime 1 }
}
```

7.87. properties(未訳) <*property*> [, <*property*> ...]

properties(未訳) <*property*> [, <*property*> ...]			
Description	This attribute determines which properties will be included in the report.		
Attributes	**Name**	**Type**	**Description**
	property	ID	See table below for possible values.
Context	export(未訳),		
Inheritable	No	**Scenario Spec.**	No

`all`	Include all properties.
`bookings`	Include all bookings for the report interval. Only bookings for non-filtered resources will be included.
`shifts`	Include all shift definitions.
`tasks`	Include all task definitions for non-filtered tasks.
`resources`	Include all resource definitions for non-filtered resources.

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

resource r "Resource"

task t "Task" {
  start 2000-01-01
  effort 10d
  allocate r
}

# Export the project as fully scheduled project.
export "FullProject.tjp" {
  taskattributes all
  hideresource 0
}

# Export only bookings for 1st week as resource supplements
export "Week1Bookings.tji" {
  start 2000-01-01
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

7.88. purge(未訳) <*attributeName*>

purge(未訳) <*attributeName*>			
Description	This attribute can be used to purge inherited flags or allocations. Possible values for `attributeName` are `allocations` or `flags`.		
Attributes	**Name**	**Type**	**Description**
	attributeName	ID	
Context	allocate(未訳), task(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.89. rate(未訳) <*value*>

rate(未訳) <*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(未訳), resource(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	account(未訳), task(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
```

```
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.90. rawhead(未訳) <*html*>

rawhead(未訳) <*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(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmlstatusreport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳),		
Inheritable	No	**Scenario Spec.**	No

rawhead(未訳) <html>	
See also	rawstylesheet(未訳), rawtail(未訳)

7.91. rawstylesheet(未訳) <stylesheet>

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

7.92. rawtail(未訳) <html>

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

7.93. reference(未訳) <*url*>

reference(未訳) <*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(未訳)		
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	extend(未訳)		

```
project ca "Custom Attributes" "1.0" 2003-05-28 - 2003-06-28 {
  extend task {
    reference MyLink "My Link"
    text MyText "My Text" { inherit }
  }
}

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

7.94. resource(未訳) <*id*> <*name*>

resource(未訳) <*id*> <*name*>			
Description	Task use resources to fulfil the specified efforts.		
Attributes	**Name**	**Type**	**Description**
	id	ID	
	name	STRING	

resource(未訳) *<id> <name>*			
Optional Attributes	booking(未訳), efficiency(未訳), flags(未訳), journalentry(未訳), maxeffort(未訳), limits(未訳), rate(未訳), `resource`, shift(未訳), vacation(未訳), workinghours(未訳)		
Context	The TJP File(未訳), `resource`,		
Inheritable	No	**Scenario Spec.**	No
See also	task(未訳)		

```
project resources "Resource Examples" "1.0" 2005-06-06 - 2005-06-26

# A simple resource
resource tux1 "Tux1"

# A team
resource team "A team" {
  # A 2 days of team vacation
  vacation 2005-06-07 - 2006-05-09
  resource tux2 "Tux2"
  resource tux3 "Tux3" {
    # And one extra day
    vacation 2005-06-10
  }
}

task t "An important date" {
  start 2005-06-10
}
```

7.95. resourcereport(未訳) *<file>*

resourcereport(未訳) *<file>*			
Description	This report is intended for the TaskJuggler graphical user interface. 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**
	file	STRING	

resourcereport(未訳) *<file>*			
Optional Attributes	caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rollupresource(未訳), rolluptask(未訳), scenario(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), timeformat(未訳)		
Context			
Inheritable	No	**Scenario Spec.**	No
See also	csvtaskreport(未訳), htmlaccountreport(未訳), htmlresourcereport(未訳), taskreport(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"
resource tuxia "Tuxia"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    task phase1 "Phase 1" {
      effort 5d
      allocate tuxia
    }
    task phase2 "Phase 2" {
      effort 2d
      allocate tux
    }
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

taskreport "Project Breakdown" {
  columns start, end, effort
  # Open only the first level of tasks
  rolluptask treelevel() > 1
}
```

```
resourcereport "Resource Allocations" {
  columns id, effort
  # We only want to see the tasks with real work (without parents),
  # sorted by name
  sorttasks nameup
  hidetask ~isleaf()
}
```

7.96. resourcereport(未訳)

resourcereport(未訳)			
Description	This attribute switches a calendar report from task report mode to resource report mode.		
Context	htmlweeklycalendar(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.97. responsible(未訳) <*resource*>

responsible(未訳) <*resource*>			
Description	The ID of the resource that is responsible for this task. This value is for documentation purposes only. It’s not used by the scheduler.		
Attributes	**Name**	**Type**	**Description**
	resource	ID	
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	resource(未訳)		

```
project prj "Responsible Demo" "1.0" 2005-07-15 - 2005-08-01

resource tux "Tux"
resource ubertux "Uber Tux"
```

```
task someJob "Some Job" {
  start 2005-07-15
  effort 1w
  allocate tux
  responsible ubertux
}

taskreport "Job List" {
  columns effort, resources, responsible
}
```

7.98. rollupaccount(未訳) <*logicalexpression*>

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

7.99. rollupresource(未訳) <*logicalexpression*>

rollupresource(未訳) <*logicalexpression*>			
Description	Do not show sub-resources of resources that match the specified logical expression.		
Attributes	**Name**	**Type**	**Description**
	logicalexpression	LOGICALEXPRESSION	

rollupresource(未訳) <*logicalexpression*>			
Context	csvresourcereport(未訳), export(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), icalreport(未訳), resourcereport(未訳), taskreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	rollupaccount(未訳), rolluptask(未訳)		

7.100. rolluptask(未訳) <*logicalexpression*>

rolluptask(未訳) <*logicalexpression*>			
Description	Do not show sub-tasks of tasks that match the specified logical expression.		
Attributes	**Name**	**Type**	**Description**
	logicalexpression	LOGICALEXPRESSION	
Context	csvtaskreport(未訳), export(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), icalreport(未訳), resourcereport(未訳), taskreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	rollupaccount(未訳), rollupresource(未訳)		

7.101. scenario(未訳) <*id*> <*name*>

scenario(未訳) <*id*> <*name*>

scenario(未訳) <*id*> <*name*>			
Description	Specifies the different project scenarios. A scenario that is nested into another one inherits all inheritable values from the enclosing scenario. There can only be one top-level scenario. It is usually called `plan` scenario. By default this scenario is pre-defined but can be overwritten with any other scenario. In this documenation each attribute is listed as scenario specific or not. A scenario specific attribute can be overwritten in a child scenario thereby creating a new, slightly different variant of the parent scenario. This can be helpful to do plan/actual comparisons if what-if-anlysises. By using bookings and enabling the projection mode you can capture the progress of your project and constantly get updated project plans for the future work.		
Attributes	**Name**	**Type**	**Description**
	id	ID	
	name	STRING	
Optional Attributes	baseline(未訳), disabled(未訳), enabled(未訳), maxpaths(未訳), minslackrate(未訳), projection(未訳), `scenario`		
Context	project(未訳), `scenario`,		
Inheritable	No	**Scenario Spec.**	No
See also	scenarios(未訳)		

```
project prj "Example" "1.0" 2005-05-29 - 2005-07-01 {
  scenario plan "Planned Scenario" {
    scenario actual "Actual Scenario"
    scenario test "Test Scenario" {
      disabled
    }
  }
}

task t "Task" {
  start 2005-05-29
  actual:start 2005-06-03
  test:start 2005-06-07
}
```

7.102. scenario(未訳) <*scenarioid*>

scenario(未訳) <*scenarioid*>			
Description	ID of the scenario that should be included in the report.		
Attributes	**Name**	**Type**	**Description**
	scenarioid	ID	The ID of the scenario.
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	scenario(未訳)		

7.103. scenarios(未訳) <*scenarioid*> [, <*scenarioid*> ...]

scenarios(未訳) <*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(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), icalreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	scenario(未訳)		

7.104. scheduled(未訳)

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

7.105. scheduling(未訳) <*type*>

<table>
<tr><th colspan="4">scheduling(未訳) <<i>type</i>></th></tr>
<tr><td>Description</td><td colspan="3">Specifies the scheduling policy for the task. A task can be scheduled from start to end (As Soon As Possible, <code>asap</code>) or from end to start (As Late As Possible, <code>alap</code>).
A task can be scheduled from start to end (ASAP mode) when it has a hard (start) or soft (depends) criteria for the start time. A task can be scheduled from end to start (ALAP mode) when it has a hard (end) or soft (precedes) criteria for the end time.

Some task attributes set the scheduling policy implicitly. This attribute can be used to explicitly set the scheduling policy of the task to a certain direction. To avoid it being overwritten again by an implicit attribute this attribute should always be the last attribute of the task.

A random mixture of ASAP and ALAP tasks can have unexpected side effects on the scheduling of the project. It increases significantly the scheduling complexity and results in much longer scheduling times. Especially in projects with many hundreds of tasks the scheduling time of a project with a mixture of ASAP and ALAP times can be 2 to 10 times longer. When the projects contains chains of ALAP and ASAP tasks the tasks further down the dependency chain will be served much later than other non-chained task even when they have a much higher priority. This can result in situations where high priority tasks do not get their resources even though the parallel competing tasks have a much lower priority.

As a general rule, try to avoid ALAP tasks whenever possible. Have a close eye on tasks that have been switched implicitly to ALAP mode because the <code>end</code> attribute comes after the <code>start</code> attribute.</td></tr>
<tr><td rowspan="2">Attributes</td><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>type</td><td>ID</td><td>Possible values are <code>alap</code> or <code>asap</code>.</td></tr>
<tr><td>Context</td><td colspan="3">task(未訳),</td></tr>
<tr><td>Inheritable</td><td>Yes</td><td>Scenario Spec.</td><td>No</td></tr>
<tr><td>See also</td><td colspan="3">depends(未訳), end(未訳), precedes(未訳), start(未訳), period(未訳)</td></tr>
</table>

```
project prj "Scheduling Example" "1.0" 2005-07-23 - 2005-09-01

task items "Project breakdown" {
  task t1 "Task 1" {
    start 2005-07-25
    end 2005-08-01
    # Implicite ALAP task
  }
  task t2 "Task 2" {
    end 2005-08-01
    start 2005-07-25
    # Implicite ASAP task
  }
  task t3 "Task 3" {
    start 2005-07-25
    end 2005-08-01
    scheduling asap
    # Explicite ASAP task
  }
  task t4 "Task 4" {
    end 2005-08-01
    start 2005-07-25
    scheduling alap
    # Explicite ALAP task
  }
}
```

7.106. separator(未訳) <*sep*>

separator(未訳) <*sep*>			
Description	Specifies the separator used in CSV reports between the values. The default value is a semicolon.		
Attributes	**Name**	**Type**	**Description**
	sep	STRING	
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.107. select(未訳) <*mode*>

select(未訳) <*mode*>			
Description	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. 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.		
Attributes	**Name**	**Type**	**Description**
	mode	ID	See table below for possible values.
Context	allocate(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	persistent(未訳)		

`maxloaded`	Pick the available resource that has been used the most so far.
`minloaded`	Pick the available resource that has been used the least so far.
`minallocated`	Pick the resource that has the smallest allocation factor. The allocation factor is calculated from the v
`order`	Pick the first available resource from the list.
`random`	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 tuxux 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}
}
```

7.108. shift(未訳) <*id*> <*name*>

shift(未訳) <*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. Shifts can also be used to limit work on certain tasks to the hours of the shift or to switch part time schedules of a resource.		
Attributes	**Name**	**Type**	**Description**
	id	ID	
	name	STRING	
Optional Attributes	`shift`, workinghours(未訳)		
Context	The TJP File(未訳), `shift`,		
Inheritable	No	**Scenario Spec.**	No
See also	shift(未訳)		

```
project prj "Example" "1.0" 2000-01-01 - 2000-03-31

shift s1 "Shift1" {
  # Special working hours Monday to Wednesday. Use program defaults
  # for other days.
  workinghours mon 10:00 - 12:00, 13:00-15:00
  workinghours tue 9:00-14:00
  workinghours wed off

  shift s2 "Shift2" {
    # Like s1 but with different times on Monday
    workinghours mon 10:00 - 17:00
  }
}
```

```
resource r1 "Resource1" {
  shift s1 2000-01-01 - 2000-01-10
  shift s2 2000-01-11 - 2000-01-20
}

shift s3 "Part-time schedule 1" {
  # The resource works on mondays, wednesdays and fridays.
  # The days that the resource doesn't work must be mentioned
  # explicitely otherwise the defaults values are used (usually
  # full-time employment).
  workinghours mon,fri 9:00 - 12:00, 13:00-18:00
  workinghours wed 9:00 - 12:00
  workinghours tue, thu off
}

shift s4 "Part-time schedule 2" {
  # The resource changes his schedule to work on tuesday and
  # thursdays. The days that the resource doesn't work must be
  # mentioned explicitely, otherwise the default values are used
  # (usually full-time employment).
  workinghours tue, thu 9:00 - 12:00, 13:00-18:00
  workinghours mon, wed, fri off
}

# Now determine when these schedules are applicable
resource r2 "Resource2" {
  # r2 works three days a week from January to June
  shift s3 2005-01-01 - 2005-01-15
  # r2 switches to two days a week
  shift s4 2005-01-15 - 2006-01-01
}

task t1 "Task1" {
  start 2000-01-01
  length 200h
  # During the specified interval only work at the shift s2 working
  # hours.
  shift s2 2000-01-09 - 2000-01-17
}
```

7.109. shift(未訳) <*shiftid*> [<*dateinterval*>]

shift(未訳) <*shiftid*> [<*dateinterval*>]

shift(未訳) *<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.
	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(未訳), resource(未訳), task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	shift(未訳)		

```
project prj "Example" "1.0" 2000-01-01 - 2000-03-31

shift s1 "Shift1" {
  # Special working hours Monday to Wednesday. Use program defaults
  # for other days.
  workinghours mon 10:00 - 12:00, 13:00-15:00
  workinghours tue 9:00-14:00
  workinghours wed off

  shift s2 "Shift2" {
    # Like s1 but with different times on Monday
    workinghours mon 10:00 - 17:00
  }
}

resource r1 "Resource1" {
  shift s1 2000-01-01 - 2000-01-10
  shift s2 2000-01-11 - 2000-01-20
}

shift s3 "Part-time schedule 1" {
  # The resource works on mondays, wednesdays and fridays.
  # The days that the resource doesn't work must be mentioned
  # explicitely otherwise the defaults values are used (usually
```

```
  # full-time employment).
  workinghours mon,fri 9:00 - 12:00, 13:00-18:00
  workinghours wed 9:00 - 12:00
  workinghours tue, thu off
}

shift s4 "Part-time schedule 2" {
  # The resource changes his schedule to work on tuesday and
  # thursdays. The days that the resource doesn't work must be
  # mentioned explicitely, otherwise the default values are used
  # (usually full-time employment).
  workinghours tue, thu 9:00 - 12:00, 13:00-18:00
  workinghours mon, wed, fri off
}

# Now determine when these schedules are applicable
resource r2 "Resource2" {
  # r2 works three days a week from January to June
  shift s3 2005-01-01 - 2005-01-15
  # r2 switches to two days a week
  shift s4 2005-01-15 - 2006-01-01
}

task t1 "Task1" {
  start 2000-01-01
  length 200h
  # During the specified interval only work at the shift s2 working
  # hours.
  shift s2 2000-01-09 - 2000-01-17
}
```

7.110. shorttimeformat(未訳) <*format*>

shorttimeformat(未訳) <*format*>			
Description	Specifies time format for time short specifications. This is normal just the hour and minutes.		
Attributes	**Name**	**Type**	**Description**
	format	STRING	
Context	csvresourcereport(未訳), csvtaskreport(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), project(未訳), resourcereport(未訳), taskreport(未訳),		

shorttimeformat(未訳) *<format>*			
Inheritable	Yes	**Scenario Spec.**	No
See also	timeformat(未訳)		

7.111. showprojectids(未訳)

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

7.112. sloppy(未訳) *<value>*

sloppy(未訳) *<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(未訳),		
Inheritable	No	**Scenario Spec.**	No

`sloppy 0:` Period may not contain any off-duty hours, vacation or other task assignments. (Default)
`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.

```
project prj "Project" "1.0" 2003-06-05 +1m {
  # 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 { strict }
  }
}

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 t1 2003-06-06 +8h { sloppy 2 }
  booking t1 2003-06-08 +4h,
            2003-06-09 +4h { sloppy 2 }
  # Book interval that extends into off-hours.
  booking t1 2003-06-11-8:00 +10h { overtime 1 }
}
```

7.113. sloppy(未訳)

sloppy(未訳)			
Description	Puts the scenario in sloppy bookings mode. This is the default. This mode makes only sense when you also use projection mode for this scenario. In sloppy mode all task that don't have any bookings provided will be filled with bookings according to the original schedule. In strict mode, no bookings will be filled in for any task prior to the current (or now) date.		
Context	projection(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	booking(未訳), projection(未訳), strict(未訳)		

7.114. sortaccounts(未訳) <*criteria*> [, <*criteria*> ...]

sortaccounts(未訳) <*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(未訳), htmlaccountreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.115. sortresources(未訳) <*criteria*> [, <*criteria*> ...]

sortresources(未訳) <*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 resources, the next criteria will be used to sort the resources 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

sortresources(未訳) <`criteria`> [, <`criteria`> ...]			
Context	csvresourcereport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.116. sorttasks(未訳) <*criteria*> [, <*criteria*> ...]

sorttasks(未訳) <`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 tasks, the next criteria will be used to sort the tasks within this group.		
Attributes	**Name**	**Type**	**Description**

sorttasks(未訳) <`criteria`> [, <`criteria`> ...]			
	criteria	SORTINGCRITERIA	Possible values are fullnamedown, fullnameup, iddown, idup, indexdown, indexup, namedown, nameup, prioritydown, priorityup, responsibledown, responsibleup, sequencedown, sequenceup, tree In addition the following values are supported as well. They specify scenario specific values so they can be prefixed with the ID of the scenario and a colon (e.g. `plan:startup`). If no scenario is specified, the default scenario is used. Possible values are completeddown, completedup, criticalnessdown, criticalnessup, enddown, endup, pathcriticalnessdown, pathcriticalnessup, startdown, startup, statusdown, statusup
Context	csvtaskreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project test "Test Project" "$Id" 2000-01-01 - 2000-03-01

flags flag1, flag2, flag3, flag4

rate 100.0
```

```
resource r1 "FooResource 1"
resource r2 "FooResource 2"
resource r3 "FooResource 3"
resource r4 "FooResource 4"

account a1 "FooAccount 1" cost {
  account a3 "FooAccount 3"
}

account a2 "FooAccount 2" revenue {
  account a4 "FooAccount 4"
}

task t1 "FooTask1" {
  account a4
  task t1_1 "FooTask1_1" {
    flags flag2
    start 2000-01-01
    effort 20d
    allocate r1
    allocate r2
  }
  flags flag3
}

task t2 "FooTask2" {
  flags flag1
  start 2000-01-01
  duration 1d
  startcredit 10000.0
  account a4
}

task t3 "FooTask3" {
  flags flag4
  milestone
  start 2000-01-01
}

htmltaskreport "Report_task.html" {
  columns hierarchindex, name { title "Task Name" }, daily, effort
  sorttasks tree, startup, nameup
}

htmlresourcereport "Report_resource.html" {
}

htmlaccountreport "Report_account.html" {
  columns name, weekly
  hideaccount 0
}
```

7.117. start(未訳) <*date*>

start(未訳) <*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**
	date	DATE	
Context	csvaccountreport(未訳), csvresourcereport(未訳), csvtaskreport(未訳), export(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	end(未訳), period(未訳)		

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

resource r "Resource"

task t "Task" {
  start 2000-01-01
  effort 10d
  allocate r
}

# Export the project as fully scheduled project.
export "FullProject.tjp" {
  taskattributes all
  hideresource 0
}

# Export only bookings for 1st week as resource supplements
export "Week1Bookings.tji" {
  start 2000-01-01
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

7.118. start(未訳) <*date*>

start(未訳) <*date*>			
Description	The start date of the task. When specified for the top-level (default) scenario this attribute also implicitly sets the scheduling policy of the task to `asap`.		
Attributes	**Name**	**Type**	**Description**
	date	DATE	
Context	task(未訳),		
Inheritable	Yes	**Scenario Spec.**	Yes
See also	end(未訳), period(未訳), maxstart(未訳), minstart(未訳), scheduling(未訳), startbuffer(未訳)		

```
project duration "Duration Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task t "Enclosing" {
  start 2005-06-06
  task durationTask "Duration Task" {
    # This task is 10 calendar days long.
    duration 10d
  }

  task intervalTask "Interval Task" {
    # This task is similar to the durationTask. Instead of a start
    # date and a duration it has a fixed start and end date.
    end 2005-06-17
  }

  task lengthTask "Length Task" {
    # This task 10 working days long. So about 12 calendar days.
    length 10d
  }

  task effortTask "Effort Task" {
    effort 10d
    allocate tux
  }
}
```

7.119. startbuffer(未訳) <*percent*>

startbuffer(未訳) <*percent*>			
Description	Specifies how much slack time you expect to have at the beginning of the task. This information has no impact on the scheduling of the project. It is for documentation purposes only.		
Attributes	**Name**	**Type**	**Description**
	percent	REAL	Percent slack of the overall effort, duration or length of the task.
Context	task(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	duration(未訳), endbuffer(未訳), effort(未訳), length(未訳)		

```
project simple "Simple Project" "$Id" 2000-01-01 - 2000-01-20

resource tux1 "Tux1"

task t1 "Task1" {
  start 2000-01-01
  length 10d
  # 20% of the working time of this task are marked as buffer at the
  # beginning.
  startbuffer 20
  # An additional 10% of the working time of this task are marked as
  # buffer at the end.
  endbuffer 10.0
  allocate tux1
}

# Generate a report that lists the start end end dates for the
# buffers.
htmltaskreport "Buffer.html" {
  columns no, name, start, startbufferend, endbufferstart, end,
  startbuffer, endbuffer, duration, effort, daily
  hideresource 0
}
```

7.120. startcredit(未訳) <*amount*>

startcredit(未訳) <*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(未訳),		
Inheritable	No	**Scenario Spec.**	Yes
See also	endcredit(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26 {
  currency "USD"
}

account project_cost "Project Costs" cost
account payments "Customer Payments" revenue {
  credit 2005-06-08 "Customer down payment" 500.0
}

resource tux "Tux" {
  rate 300
}

task items "Project breakdown" {
  start 2005-06-06
  # The default account for all tasks
  account project_cost

  task plan "Plan work" {
    # Some upfront material cost
    startcredit 500.0
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
```

```
    account payments
    # Customer pays at end of acceptance
    endcredit 2000.0
  }
}

htmlaccountreport "PAndL.html" {
  timeformat "%d-%M-%y"
  accumulate
  columns index, name, weekly
}
```

7.121. statusnote(未訳) <*text*>

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

7.122. strict(未訳)

strict(未訳)			
Description	Puts the scenario in strict bookings mode. This mode makes only sense when you also use projection mode for this scenario. In strict mode, no bookings will be filled in for any task prior to the current (or now) date. In sloppy mode all task that don't have any bookings provided will be filled with bookings according to the original schedule. TaskJuggler will generate a warning for each task where the booked effort excedes the specified effort.		
Context	projection(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	booking(未訳), sloppy(未訳), projection(未訳)		

7.123. subtitle(未訳) <*text*>

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

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
```

```
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.124. subtitleurl(未訳) *<url>*

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

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
```

```
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.125. supplement(未訳) <*type*>

supplement(未訳) <*type*>			
Description	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. 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.		
Attributes	**Name**	**Type**	**Description**
	type	ID	Possible values are `resource` or `task`.
Context	The TJP File(未訳), task(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	resource(未訳), task(未訳)		

```
project test "Test Project" "$Id" 2000-01-01 - 2000-01-04

flags important

resource joe "Joe"

task top "Top Task" {
  start 2000-01-01

  task sub "Sub Task" {
  }
  supplement task sub {
    length 1d
  }
}

supplement resource joe {
  vacation 2000-02-10 - 2000-02-20
}

supplement task top {
  flags important
}
```

7.126. task(未訳) <*id*> <*name*>

task(未訳) <*id*> <*name*>			
Description	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.		
Attributes	**Name**	**Type**	**Description**
	id	ID	
	name	STRING	
Optional Attributes	account(未訳), allocate(未訳), complete(未訳), depends(未訳), duration(未訳), effort(未訳), endbuffer(未訳), endcredit(未訳), end(未訳), flags(未訳), journalentry(未訳), length(未訳), maxend(未訳), maxstart(未訳), milestone(未訳), minend(未訳), minstart(未訳), note(未訳), period(未訳), precedes(未訳), priority(未訳), projectid(未訳), purge(未訳), reference(未訳), responsible(未訳), scheduled(未訳), scheduling(未訳), shift(未訳), startbuffer(未訳), startcredit(未訳), start(未訳), statusnote(未訳), supplement(未訳), `task`		
Context	The TJP File(未訳), `task`,		
Inheritable	No	**Scenario Spec.**	No
See also	resource(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
```

```
    depends !implementation
  }
}

taskreport "My Tasks"
```

7.127. taskattributes(未訳) <*attribute*> [, <*attribute*> ...]

taskattributes(未訳) <*attribute*> [, <*attribute*> ...]			
Description	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. `complete`, `depends`, `flags`, `maxend`, `maxstart`, `minend`, `minstart`, `note`, `priority`, `responsible` By specifying the ID of a user-defined attribute, these can be included as well. A special case the is `all` 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.		
Attributes	**Name**	**Type**	**Description**
	attribute	ID	
Context	export(未訳),		
Inheritable	No	**Scenario Spec.**	No

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

resource r "Resource"

task t "Task" {
  start 2000-01-01
  effort 10d
  allocate r
}
```

```
# Export the project as fully scheduled project.
export "FullProject.tjp" {
  taskattributes all
  hideresource 0
}

# Export only bookings for 1st week as resource supplements
export "Week1Bookings.tji" {
  start 2000-01-01
  end 2000-01-08
  properties bookings
  hideresource 0
}
```

7.128. taskprefix(未訳) <*prefix*>

taskprefix(未訳) <*prefix*>			
Description	All tasks in the included file are added as sub-tasks of the task specified by `taskprefix`. The `taskprefix` must be a valid absolute ID of an already defined task.		
Attributes	**Name**	**Type**	**Description**
	prefix	ID	
Context	include(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	task(未訳)		

```
project yourId "Your Project" "1.0" 2005-04-05 - 2005-05-01

task main "Main task" {
}

include "Include2.tji" { taskprefix main }
```

7.129. taskreport(未訳) <*file*>

taskreport(未訳) <*file*>			
Description	This report is intended for the TaskJuggler graphical user interface. 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**
	file	STRING	
Optional Attributes	caption(未訳), columns(未訳), end(未訳), headline(未訳), hideresource(未訳), hidetask(未訳), loadunit(未訳), period(未訳), rollupresource(未訳), rolluptask(未訳), scenario(未訳), shorttimeformat(未訳), showprojectids(未訳), sortresources(未訳), sorttasks(未訳), start(未訳), taskroot(未訳), timeformat(未訳)		
Context			
Inheritable	No	**Scenario Spec.**	No
See also	csvtaskreport(未訳), htmlaccountreport(未訳), htmlresourcereport(未訳), resourcereport(未訳)		

```
project simple "Simple Project" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"
resource tuxia "Tuxia"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    task phase1 "Phase 1" {
      effort 5d
      allocate tuxia
    }
    task phase2 "Phase 2" {
      effort 2d
      allocate tux
    }
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }
```

```
  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

taskreport "Project Breakdown" {
  columns start, end, effort
  # Open only the first level of tasks
  rolluptask treelevel() > 1
}

resourcereport "Resource Allocations" {
  columns id, effort
  # We only want to see the tasks with real work (without parents),
  # sorted by name
  sorttasks nameup
  hidetask ~isleaf()
}
```

7.130. taskroot(未訳) <*root*>

taskroot(未訳) <*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	csvtaskreport(未訳), export(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), taskreport(未訳), xmlreport(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project prj "Taskroot Example" "1.0" 2005-07-22 - 2005-08-26

task items "Project breakdown" {
  start 2005-07-22
```

```
  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    task phase1 "Phase 1" {
      length 5d
      depends !!plan
    }
    task phase2 "Phase 2" {
      length 3d
      depends !phase1
    }
    task phase3 "Phase 3" {
      length 4d
      depends !phase2
    }
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

taskreport "My Tasks" {
  taskroot items.implementation
}
```

7.131. timezone(未訳) <*zone*>

timezone(未訳) <*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 `tzselect` 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	

timezone(未訳) *<zone>*			
Context	project(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project tz "Timezone" "1.0" 2005-06-06-0:00-UTC - 2005-06-07-0:00-UTC {
  timezone "Europe/Athens"
}

task item "Project" {
  start 2005-06-06-12:00
}
```

7.132. timeformat(未訳) *<format>*

timeformat(未訳) *<format>*			
Description	Determines how time specifications in reports look like.		
Attributes	**Name**	**Type**	**Description**
	format	STRING	See table below for possible values.
Context	csvresourcereport(未訳), csvtaskreport(未訳), htmlaccountreport(未訳), htmlmonthlycalendar(未訳), htmlresourcereport(未訳), htmltaskreport(未訳), htmlweeklycalendar(未訳), project(未訳), resourcereport(未訳), taskreport(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	shorttimeformat(未訳)		

Ordinary characters placed in the format string are copied 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 countries %d/%m/
%e	Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space. (SU)
%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 ISO week number (
%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 a blank. (See also %
%l	The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank. (See also %
%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 current locale. Noon
%P	Like %p but in lowercase: 'am' or 'pm' or %a corresponding string for the current locale. (GNU)
%r	The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to '%I:%M:%S %p'. (SU)
%R	The time in 24-hour notation (%H:%M). (SU) For a version including the seconds, see %T below.
%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 first Sunday as the fi
%V	The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the f
%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 the first Monday as 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 "%a, %d %%b %Y
%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.

7.133. timingresolution(未訳) <*value*> <*unit*>

timingresolution(未訳) <*value*> <*unit*>			
Description	Sets the minimum timing resolution. The smaller the value, the longer the scheduling process lasts and the more memory the application needs. The default and maximum value is 1 hour. The smallest value is 5 min. 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. The timing resolution should be set prior to any value that represents a time value like now or workinghours.		
Attributes	**Name**	**Type**	**Description**
	value	INTEGER	
	unit	UNIT	
Context	project(未訳),		
Inheritable	No	**Scenario Spec.**	No

7.134. title(未訳) <*text*>

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

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"
```

```
task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}

# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.135. titleurl(未訳) <*url*>

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

```
project prj  "Project" "1.0" 2005-01-01 - 2005-03-01

resource r "Resource"

task t "Task" {
  task s "SubTask" {
    start 2005-01-01
    effort 5d
    allocate r
  }
}

# Just a very basic report with some standard columns
htmltaskreport "SimpleReport.html" {
  columns hierarchindex, name, start, end, weekly
}

# Report with custom colum title
htmltaskreport "CustomTitle.html" {
  columns hierarchindex, name { title "Work Item" }, effort
}

# Report with custom colum title and subtitle
htmltaskreport "CustomSubTitle.html" {
  columns hierarchindex, name,
          monthly { title " " subtitle "$${month} $${year}" }
  loadunit days
}

# Report with efforts only for leaf tasks
htmltaskreport "LeafEfforts.html" {
  columns hierarchindex, name,
          effort { hidecelltext ~isLeaf() }
}
```

```
# Report with link in title of calendar
htmltaskreport "LinkURL.html" {
  columns hierarchindex, name,
          monthly { subtitleurl "Monthly-Detail-$${month}.html" }
}

# Report with link to page with furter task details
htmltaskreport "LinkToTaskDetails.html" {
  columns hierarchindex,
          name { cellurl "TaskDetails-$${taskid}.html"
                 hidecellurl ~isLeaf() }, start, end
}

# Report with index and task name combined in one single column
htmltaskreport "CombinedColumn.html" {
  columns name { celltext "$${hierarchno} $${0}"}, start, end, weekly
}
```

7.136. vacation(未訳) <*name*> <*interval*>

vacation(未訳) <*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(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	vacation(未訳)		

```
project prj "Vacation Examples" "1.0" 2005-07-22 - 2006-01-01

# Labor Day
vacation "Labor Day" 2005-09-05
# 2 days Christmas break (27th not included!)
vacation "Christmas" 2005-12-25 - 2005-12-27

resource team "A team" {
```

```
  # 2 days of team vacation
  vacation 2005-10-07 +2d
  resource tux2 "Tux2"
  resource tux3 "Tux3" {
    # And one extra day
    vacation 2005-08-10
  }
}

# The vacation property is also usefull when new employees start
# working in the course of a project or if someone quits.
resource tuxia "Tuxia" {
  # Tuxia is a new employee as of August 1st 2005
  vacation 1971-01-01 - 2005-08-01
}

resource tuxus "Tuxus" {
  # Tuxus quits his job on September 1st 2005
  vacation 2005-09-01 - 2030-01-01
}

task t "An important date" {
  start 2005-07-22
}
```

7.137. vacation(未訳) <*interval*>

vacation(未訳) <*interval*>			
Description	Specify a vacation period for the resource. It can also be used to block out the time before a resource joint or after it left. For employees changing their work schedule from full-time to part-time, or vice versa, please refer to the 'Shift' property.		
Attributes	**Name**	**Type**	**Description**
	interval	DATEINTERVAL	
Context	resource(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	vacation(未訳), shift(未訳)		

```
project prj "Vacation Examples" "1.0" 2005-07-22 - 2006-01-01
```

```
# Labor Day
vacation "Labor Day" 2005-09-05
# 2 days Christmas break (27th not included!)
vacation "Christmas" 2005-12-25 - 2005-12-27

resource team "A team" {
  # 2 days of team vacation
  vacation 2005-10-07 +2d
  resource tux2 "Tux2"
  resource tux3 "Tux3" {
    # And one extra day
    vacation 2005-08-10
  }
}

# The vacation property is also usefull when new employees start
# working in the course of a project or if someone quits.
resource tuxia "Tuxia" {
  # Tuxia is a new employee as of August 1st 2005
  vacation 1971-01-01 - 2005-08-01
}

resource tuxus "Tuxus" {
  # Tuxus quits his job on September 1st 2005
  vacation 2005-09-01 - 2030-01-01
}

task t "An important date" {
  start 2005-07-22
}
```

7.138. version(未訳) <*number*>

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

```
project simple "XML Report Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

# This is the format that e. g. tjx2gantt can read
xmlreport "Version1.tjx" {
  version 1
}

# This is the format that taskjuggler can read and write
xmlreport "Version2.tjx" {
  version 2
}
```

7.139. weekdays(未訳) <*weekday*> [, <*weekday*> ...]

weekdays(未訳) <*weekday*> [, <*weekday*> ...]			
Description	This attribute specifies a list of weekdays that are shown in the report.		
Attributes	**Name**	**Type**	**Description**
	weekday	WEEKDAY	
Context	htmlmonthlycalendar(未訳), htmlweeklycalendar(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	workinghours(未訳)		

7.140. weeklymax(未訳) <*value*> <*unit*>

weeklymax(未訳) <*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(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	dailymax(未訳), monthlymax(未訳)		

```
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" {
  # Limit the usage of this resource to a maximum of 2 hours per day,
  # 6 hours per week and 2.5 days per month.
  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 }
  }
```

```
}
```

7.141. weekstartsmonday(未訳)

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

7.142. weekstartssunday(未訳)

weekstartssunday(未訳)			
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(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	weekstartsmonday(未訳)		

7.143. workinghours(未訳) <*weekday*> [, <*weekday*> ...] <*interval*> [, <*interval*> ...]

workinghours(未訳) <*weekday*> [, <*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	

workinghours(未訳) <*weekday*> [, <*weekday*> ...] <*interval*> [, <*interval*> ...]			
	interval	TIMEINTERVAL	
Context	project(未訳), resource(未訳), shift(未訳),		
Inheritable	Yes	**Scenario Spec.**	No
See also	dailyworkinghours(未訳), yearlyworkingdays(未訳)		

```
project prj "Example Project" "1.0" 2000-01-01 - 2000-03-09 {
  # The following attributes are all optional. They illustrate the
  # default values. These attributes are only needed if you want to
  # specify different values than those listed below.
  dailyworkinghours 8
  yearlyworkingdays 260.714
  timingresolution 60min
  timeformat "%Y-%m-%d %H:%M"
  shorttimeformat "%H:%M"
  currencyformat "(" ")" "," "." 0
  weekstartsmonday
    workinghours mon - fri 9:00 - 12:00, 13:00 - 18:00
    workinghours sat, sun off
  scenario plan "Plan" {
  }
}

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

7.144. xmlreport(未訳) <*file*>

xmlreport(未訳) <*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. The file name should have a `.tjx` extension. Version 2 files are gzip compressed XML Files. The DTD for the version 2 file format can be found on the TaskJuggler Web Site (http://www.taskjuggler.org/show_dtd.php).		
Attributes	**Name**	**Type**	**Description**
	file	STRING	

xmlreport(未訳) *<file>*			
Optional Attributes	hideresource(未訳), hidetask(未訳), hideaccount(未訳), rollupresource(未訳), rolluptask(未訳), rollupaccount(未訳), scenarios(未訳), taskroot(未訳), version(未訳)		
Context	The TJP File(未訳),		
Inheritable	No	**Scenario Spec.**	No

```
project simple "XML Report Example" "1.0" 2005-06-06 - 2005-06-26

resource tux "Tux"

task items "Project breakdown" {
  start 2005-06-06

  task plan "Plan work" {
    length 3d
  }

  task implementation "Implement work" {
    effort 5d
    allocate tux
    depends !plan
  }

  task acceptance "Customer acceptance" {
    duration 5d
    depends !implementation
  }
}

# This is the format that e. g. tjx2gantt can read
xmlreport "Version1.tjx" {
  version 1
}

# This is the format that taskjuggler can read and write
xmlreport "Version2.tjx" {
  version 2
}
```

7.145. yearlyworkingdays(未訳) <*days*>

yearlyworkingdays(未訳) <*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. When public holidays and vacations are disregarded, this value should be equal to the number of working days per week times 52.1428 (the average number of weeks per year). E. g. for a culture with 5 working days it is 260.714 (the default), for 6 working days it is 312.8568 and for 7 working days it is 365.		
Attributes	**Name**	**Type**	**Description**
	days	REAL	
Context	project(未訳),		
Inheritable	No	**Scenario Spec.**	No
See also	dailyworkinghours(未訳), loadunit(未訳), vacation(未訳), workinghours(未訳)		

```
project prj "Example Project" "1.0" 2000-01-01 - 2000-03-09 {
  # The following attributes are all optional. They illustrate the
  # default values. These attributes are only needed if you want to
  # specify different values than those listed below.
  dailyworkinghours 8
  yearlyworkingdays 260.714
  timingresolution 60min
  timeformat "%Y-%m-%d %H:%M"
  shorttimeformat "%H:%M"
  currencyformat "(" ")" "," "." 0
  weekstartsmonday
    workinghours mon - fri 9:00 - 12:00, 13:00 - 18:00
    workinghours sat, sun off
  scenario plan "Plan" {
  }
}

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

Chapter 8. The Example: Accounting Software(日本語訳なし)

```
/*
 * 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 - 2002-04-28 {
  # 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 the Euro.
  currency "EUR"

  # We want to compare the baseline scenario to one with a slightly
  # delayed start.
  scenario plan "Plan" {
    # Mark all paths as critical that have less than 10% slack time.
    minslackrate 10.0
    scenario delayed "Delayed"
  }
}

# This is not a real copyright for this file. It's just used as an example.
copyright "© 2002 Crappy Software, Inc."

# 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

# Register Good Friday as a global holiday for all resources.
vacation "Good Friday" 2002-03-29

# This is one way to form teams
macro allocate_developers [
  allocate dev1
  allocate dev2 { limits { dailymax 4h } }
  allocate dev3
]

flags team
```

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

  flags team
}
resource misc "The Others" {
  resource test "Peter Murphy" { limits { dailymax 6.4h } 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 "Documentation" cost
account rev "Payments" revenue

# Now we specify the work packages. The whole project is described as
# a task that contains subtasks. These subtasks 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 typing.
task AcSo "Accounting Software" {

  # All work-related costs will be booked to this account unless the
  # subtasks specify something different.
  account dev

  task spec "Specification" {
    # The effort to finish this task is 20 man-days.
    effort 20d
    # Now we use the macro declared above to allocate the resources
    # for this task. Because they can work in parallel, they may finish this
    # task earlier than in 20 working-days.
    ${allocate_developers}
    # Each task without subtasks must have a start or an end
    # criterion and a duration. For this task we use a reference to a
    # milestone defined further below as the start criterion. So this task
    # can not start before the specified milestone has been reached.
    # References to other tasks may be relative. Each exclamation mark (!)
    # means 'in the scope of the enclosing task'. To descent into a task, the
    # fullstop (.) 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 task (and all its subtasks) to 1000, the top
  # priority. The higher the priority, the more likely the task will
  # get the requested resources.
  priority 1000

  # All subtasks depend on the specification task.
  depends !spec

  task database "Database coupling" {
    effort 20d
    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, because it should have been
    # finished already. To document this, we specify that the task
    # 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
    allocate dev1, dev2
  }
}

task test "Software testing" {

  task alpha "Alpha Test" {
    # Efforts can not only be specified as man-days, but also as
    # man-weeks, man-hours, etc. By default, TaskJuggler assumes
    # that a man-week is 5 man-days or 40 man-hours. These values
    # can be changed, of course.
    effort 1w
    # This task depends on a task in the scope of the enclosing
    # task's enclosing task. So we need two exclamation marks (!!)
    # to get there.
    depends !!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 "Project start" {
    # A task that has no duration is a milestone. It only needs a
    # start or end criterion. 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 planned run to the
    # delayed run of the project.
    delayed:start 2002-01-20
    # At the beginning 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 "Beta version" {
    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 values 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 specified completely. 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.

# A traditional Gantt Chart for the TaskJugglerUI
taskreport "Gantt Chart" {
  headline "Project Gantt Chart"
  columns hierarchindex, name, start, end, effort, duration, chart
  # 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"
  loadunit days
  hideresource 1
}

# A list of tasks showing the resources assigned to each task.
taskreport "Task Usage" {
  headline "Task Usage Report"
  columns hierarchindex, name, start, end, effort { title "Work" }, duration,
    cost, revenue
  timeformat "%Y-%m-%d"
  loadunit days
  hideresource ~isLeaf()
  sortresources nameup
}

# A list of all tasks with the percentage completed for each task
taskreport "Tracking Gantt" {
  headline "Tracking Gantt Chart"
  columns hierarchindex, name, start, end, effort { title "Work" }, duration,
    completed, chart
  timeformat "%a %Y-%m-%d"
  loadunit days
  hideresource 1
}

# A graph showing resource allocation. It identifies whether each
# resource is under- or over-allocated for.
resourcereport "Resource Graph" {
  headline "Resource Allocation Graph"
  columns no, name, rate, utilization, freeload, chart
  loadunit days
  hidetask 1
}
```

```
# A list of all project resources, both human and material resources,
# together with the associated costs.
resourcereport "Resource Sheet" {
  headline "Resource Sheet"
  columns no, name, efficiency, id, maxeffort, rate
  loadunit days
  hidetask 1
}

# A list of resources and each task associated with each resource.
resourcereport "Resource Usage" {
  headline "Resource Usage Report"
  columns no, name, utilization, freeload, cost, chart
  loadunit days
  hidetask 0
}

# This report looks like a regular calendar that shows the tasks by
# their dates.
htmlweeklycalendar "Calendar.html" {
  # Only show work days in the calendar.
  weekdays mon - fri
}

# This report is a status report for the current week. It also
# provides an outlook for the next week.
htmlstatusreport "Status-Report.html" {
}

# A P&L report for the project.
htmlaccountreport "Accounting.html" {
  # 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 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
  # for each account.
  accumulate
  scenarios plan, delayed
}

xmlreport "XML-Report.tjx" {
  version 2
  hidetask 0
  hideresource 0
  scenarios plan, delayed
}

icalreport "Calendar.ics"
```

Chapter 9. TaskJuggler 1.xから2.xへ移行する (未訳)

9.1. 互換性を得ること (未訳)

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 of the wonderful new version is its failure to read your old files. With TaskJuggler we like to spare you such situations as much as possible. But TaskJuggler 1.x was written to solve the problems that we encountered. By releasing it to the general public we learned that TaskJuggler is also very useful to many other people. Some contacted us to tell us that it would be even more useful to them, if TaskJuggler could 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 would no longer be supported.

As TaskJuggler uses plain text file as its main data format, you will always be able to read in your old files. But in some cases, you need to change certain syntax constructs to the new 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 the new syntax. The following sections discuss the conceptual changes and what statements need to be changed.

9.1.1. 記法の変更 (未訳)

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 are used and the table elements are customizable now. Old stylesheets will no longer work, since the attribute names have changed. An HTML report contains CSS attribute class specifications 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 dates, if they have their subtasks 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 a new XML format has been added. The old format is still supported. TaskJuggler can read both old and new format XML files but will use the new XML format for output.

9.1.2. スケジューラの変更 (未訳)

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 tasks 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, 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 10. 質問と回答 (未訳)

10.1. 一般的な質問 (未訳)

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

A: Qt is a very powerful 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.

10.2. コンパイルとインストール (未訳)

Q: Can TaskJuggler be compiled and used on Windows?

A: Probably yes, but we have never tried it. It should compile but may require some minor tweaks of the source. You should have good knowledge of C++ and Qt when you try this. Please let us know if you were successful.

10.3. 使用法 (未訳)

Nothing here yet.

Chapter 11. 著作権 (未訳)

Chapter 12. Trademarks(未訳)

Linux is a registered trademark of Linus Torvalds.

KDE and the K Desktop environment are registered trademarks of KDE e. V.

TaskJuggler is a trademark of Chris Schlaeer.

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