[redacted]

# Optimize My Day

# SCHEDULER

# TECHNICAL REFERENCE GUIDE


| | |
|---|---|
| Author: | Ernst Verbeek |
| Creation Date: | July 29, 2013 |
| Last Updated: | September 8 , 2017 |

# 1. DOCUMENT CONTROL

## 1.1 Change Record

| Date | Author | Version | Change Reference |
|---|---|---|---|
| 29-7-2013 | Ernst Verbeek | 0.9 | Initial version |
| 13-9-2013 | Frank Klomp | 0.91 | Review |
| 30-9-2013 | Ernst Verbeek | 0.92 | Added content |
| 1-10-2013 | Frank Klomp | 0.93 | Review |
| 1-10-2013 | Ernst Verbeek | 0.94 | Updated content |
| 22-5-2014 | Ernst Verbeek | 7.7 | Updated content |
| 2-11-2015 | Frank Klomp | 8.3.2 | Updated content |
| 15-1-2015 | Thomas Lukas | 8.6.2 | Updated content |
| 2-2-2015 | Thomas Lukas | 8.6.3 | Updated content |
| 27-4-2016 | Ernst Verbeek | 9.0 | Updated content |
| 27-9-2016 | Ernst Verbeek | 9.2 | Updated content |
| 5-10-2016 | Ernst Verbeek | 9.3 | Updated content |
| 17-11-2016 | Ernst Verbeek | 9.4 | Updated content |
| 15-12-2016 | Ernst Verbeek | 9.4 | Updated content |
| 17-2-2017 | Ernst Verbeek | 9.9 | Updated content |
| 8-9-2017 | Ernst Verbeek | 10.2 | Updated content |

## 1.2 Distribution

| Copy | Recipient | Role |
|---|---|---|

## 1.3 Reviewers

| Copy | Recipient | Role |
|---|---|---|

## 1.4 Reference to (Linked) Documents

| Doc No. | Document Name | Contained in | Owner |
|---|---|---|---|

# CONTENT

# 2. INTRODUCTION

This document describes the technical details of the OMD Visual Scheduler. It's intended audience are consultants and integration partners, who need to understand what the scheduler is and how it can be integrated and configured. Also, information on its internal working are given, which should help to understand the Scheduler's behavior.

The Scheduler XML Schema contains detailed information on the data model used by the scheduler. It should therefore be considered an important part of the technical documentation.

For end-users, the document 'OMD Visual Scheduler - User Guide' is available. This document covers working with the Scheduler using the UI application.

# 3. THE VISUAL SCHEDULER

## 3.1 Purpose

The OMD Visual Scheduler (scheduler) is a high performance planning engine. Its main purpose is to read a set of planning data, generate a viable planning from that, and store the results. The data the scheduler works on pertains mainly to tasks (plannable pieces of work) and resources (work performing entities).

The scheduler comes both in the form of an engine which can be called from the command line (batch scheduler), and as a feature rich graphical application, with which the planning can be visualized and manipulated (visual scheduler).

## 3.2 Integration

When integrating the scheduler into a larger system, the main concern is how the scheduler will receive the data to work on, and how the results should be stored. Both XML files and direct SQL connections are supported (or a mix). Usually the best approach is to connect the scheduler directly to a SQL database, so transactions can be used. This means that the database needs to provide a set of views, tables and stored functions that adhere to the scheduler schema.

The Visual Scheduler itself can be integrated into any web based system by using running it as a Java applet. This requires the use of a scheduler web service.

## 3.3 Configuration

Configuring the scheduler is done by means of command line parameters and an XML document containing the settings (called 'preferences'). The command line options mainly control the technical aspects of the scheduler, such as file locations and modes of operation. The preferences control the functional aspects the scheduler such as planning parameters and UI settings.

## 3.4 Dependencies

The scheduler relies heavily on the concepts of geocoding and routing. The scheduler will be supplied including a complete mapping environment for this. The geographical area covered is dependent on the purchased license.

# 4. MODES OF OPERATION

The scheduler can be used in several modes. The mode is determined by the startup parameters and the Java main class.

## 4.1 Batch Scheduler

The Batch Scheduler is used for background processes, during which the input for the Scheduler is turned into an output that can be processed further by the caller. The processing of the input depends on the Scheduler preferences, which define in detail the various cost-related parameters and algorithm settings. Often, the Batch Scheduler is used on a regular basis, once an hour or once a day, depending on the requirements. Implementations of the Scheduler may define several Batch Scheduler configurations for different planning purposes.

## 4.2 Visual Scheduler

The Visual Scheduler is a front-end application that can be used by planners, dispatchers or supervisors. Apart from visualizing the planning, it allows users to trigger planning or request advices for new or currently planned visits. The Visual Scheduler also provides online KPI calculations that can help measuring service performance.

The Visual Scheduler can be run as a Java application, a Java Webstart application, or an applet within a browser.

## 4.3 Scheduler Webserver

The scheduler webserver is needed in scenarios where a Visual Scheduler is used in either Java webstart or Applet mode. Is these cases, the scheduler is usually not allowed to access the file system and/or SQL database directly because of the Java security model. The scheduler webserver acts as a 'proxy' through which the scheduler can read and update the planning information, and execute geocoding or routing requests.

## 4.4 Preference editor

The Preference Editor is a graphical tool for editing a preferences XML file. The preferences are presented in a tabbed interface and grouped by functionality. However, not all preferences are accessible and/or editable with this tool. For example, the workflow can be viewed but not changed. In these cases, the XML file can be edited directly using a text based (XML) editor.

## 4.5 Geocoder

The Geocoder tool can be used to verify or update geographical locations of either resource or task data. In some cases, the geocoding will not be able to resolve addresses due to either incomplete data sets or misspelled information. In these cases, the Geocoder can be used to specify substitution rules, so that whenever that address information is encountered, the scheduler will know which location to use.

# 5. SCHEDULER CONCEPTS

## 5.1 Tasks

Tasks are the units of work that need to be performed. Each task usually defines the following properties: its estimated duration, address of the customer, time window within the task must be performed. Optionally, it can define properties such as: skills required, preferred resource, service hours, etc.

A 'floating task' is a task that has no specific location. Instead, it is assumed to take place at the same location as the previous task. Consequently, there is no travel time. Typical examples of floating tasks are lunch breaks. By modeling

A 'fixed task' is a task which is assigned to a specific resource at a specific time. The scheduler will normally not move this task.

## 5.2 Resources

A resource is work performing entity, usually a person. Each resource usually defines the following properties: name, address. Optionally, it can define properties such as: its skills, work rosters.

## 5.3 Working hours and rosters

A resource will usually work according to a roster (pattern). Each roster contains information on which hours are working hours and at which location the working day starts and ends.

## 5.4 Skills

A skill is a qualification that applies to both tasks and resources. Tasks that require specific skills will only be assigned to resources that have those skills. A skill can be either simple (its there or not) or have a level (a qualification level for example).

If the implementation of the Scheduler requires to handle more than one skill for each task, the skill field of the task can be used to express this constraint as a logical expression. The logical expression can contain logical *and* (&&), logical *or* (||) and logical *negation*(!) predicates as well as parentheses for cascaded logical expressions. Furthermore, comparisons are allowed. They can be used to compare required skill levels by using the level() function.

Examples

The requirement for skills A and B would be expressed as:
"A" && "B"

The requirement for A or B, with the constraint that B should not occur in combination with skill C would be expressed as:
"A" || ("B" && !"C")

Only resources that have a matching skill set will be considered a candidate for the task. Note that all skills need to be quoted to allow any string literal as a skill identifier. There is one limitation for skill identifiers: they may not contain linefeed or end of line characters nor the " character.

The level() method retrieves the level of the resource's skill definition. This allows expressions such as

level("A") > 1.5 or level("A") == level("B").

Supported comparisons are <, >, <=,>=, == (equal) and != (not equal).The level function will cause an exception to be thrown if the resource does not have the provided skill, causing the evaluation of the expression to return false.lt; level("A") constraint.

In case the expression is not recognized by the Scheduler constraint parser, the string will be interpreted as a single unquoted skill identifier. Evaluation of the constraint will then return true if and only if one of the resource's skills matches the unquoted skill identifier.

## 5.5 Constraints

Constraints are the set of rules that are applied to the planning, to ensure that the planning is feasible. Constraints can either be hard and soft, and have an effect on the cost (score or fitness) of an assignment. The overall goal of the scheduler is to compute a plan that doesn't violate any hard constraints, and has the lowest overall cost possible. In practice, it may not always be possible to meet these goals, and a tradeoff is necessary.

## 5.6 Grouping

TBD

## 5.7 Workflow

The workflow during the planning process of a task is defined by a set of status values and transitions between these status values. Most organizations use different names to identify these state values. the Scheduler has a predefined set of status values which correspond to the most simple possible workflow:

| | | |
|---|---|---|
| PL | Plannable | In this status, a task is available for planning. |
| IP | In Planning | Whenever a task is planned by the Scheduler, the task status changes to IP to indicate that the task has now a defined resource and scheduled date/time. |

The table is stored in the preference file and is only editable in an editor. The workflow for a specific implementation should not change frequently. It is possible to specify a color for each status in the workflow. If specified, all tasks in the Gantt charts that have a colored status will be displayed in the defined status color. Tasks with status PL are displayed in the Plannable tab. Tasks with status IP are displayed in the Planned tab.

By setting the *closed* attribute to true, a status is recognized as a final state. These tasks will not be editable anymore.

By setting the *remove* attribute to true, tasks with the state will be removed from the Gantt charts. They will still appear in the list of closed tasks.

The *replicate* attribute indicates that the status is to be replicated between an administrative system and a Scheduler database. The default is true. If set to false, the status will not be overwritten by the replicator.

If the *optimizeWithinTrip* attribute is set to true, the visit may be moved within the current trip. this is very often useful when reassignment to another resource or day is not possible anymore but moving the visit into a better position within the trip is still allowed.

If the *onTheWay* attribute is set to true, the visit is a status that may not be changed anymore because the resource is currently on its way to the location.

The *email* attribute is used to identify a list of e-mail accounts that are to be notified if a task

is changed into the according status. If running in interactive mode, the application will ask for a reason to be added to the message.

Example

```
<Workflow default="PL">
   <Status code="OS" color="ffcc99"/>
   <Status code="RM" color="ffcc99" remove="true"/>
   <Status code="RC" closed="true" color="fafafa"/>
   <Status code="IP" unplan="PL" color="ccccff"/>
   <Status code="PL" plan="IP" color="99ff99"/>
   <Status code="IP" unplan="PL" color="ccccff"/>
</Workflow>
```

## 5.8 Timeline

The closer the planning reaches the timeline, the more important it is to adapt the planning to the current situation. Therefore, the Scheduler extends visits that are currently active but have not been closed yet, if the preference "extend exceeding visits" is set and the timeline has exceeded the planned visit end. The extension is performed whenever a recalculation of the trip occurs. It is possible to combine the recalculation with the refresh process. In that case, the Scheduler automatically adapts the plan after having refreshed the data and immediately commits the changes.

# 6. CONSTRAINTS AND COSTS

## 6.1 Hard and soft constraints

Any real world task planning has a number of constraints, such as working hours, opening hours, skills. A constraint can either be hard or soft.

A hard constraint is a constraint that must be met from the schedulers perspective, otherwise it is not considered a viable option. Examples of these are: a tasks that must be performed by a resource with a certain skill; a task that may only be serviced within the opening hours.

A soft constraint is a constraint that involves a penalty (cost) when they are encountered. For the scheduler, soft constraints are taken into account indirectly, when the total cost of an assignment are evaluated. Since the scheduler will always try to minimize the overall cost of the plan, solutions that do not violate hard and soft constraints will be deemed better.

The scheduler will never violate hard constraints when computing the plan. However, a planner may manually override this restriction, and perform an assignment that is deemed invalid. This should be done with care, since the optimisation routines will often not be able to optimize such routes.

Note

Even when a solution violates a hard constraint, the costs of that trip compute as MAX_LONG.

## 6.2 Cost calculation

The Scheduler uses a cost function when estimating the fitness of a plan. This cost function considers a set of real-world constraints that reflect the quality of a plan. The term *cost* as such does not have a financial meaning (although theoretically possible), but rather a virtual cost (or penalty) that allows to compare one plan with another. In some cases, one might also want to refer to a fitness parameter.

Let p be a plan for the assignment of tasks to resources. The assignments result in trips, consisting of several stops, starting with a departure and an arrival. All other stops are visits to a client's site or other locatable activities.

The total cost of a plan is the sum of the costs for each trip in the plan:

total cost(p) = sum(cost(each trip in p)) + sum(cost(each unplanned task))

The cost of a trip consists of the costs of each stop in the trip plus a cost for possible overtime (in case the trip is longer than the resource's working hours):

cost(trip) = sum(cost(each stop in trip)) + overtime(trip)

Note

The fact that a task is not planned yet is not considered a cost.

### 6.2.1 Overtime Factor

The overtime costs are calculated by comparing the (end) time of the arrival to the end time of the second last stop. In case they are subsequent in time, no costs are applied. Otherwise, the time difference (in minutes) is multiplied with the *overtime factor* . The overtime factor defaults to 1 and can be influenced in the Preferences dialog.

overtime(trip) = (time(arrival) > time(second last stop)) ? 0 : (time(second last stop) - time(arrival)) * *overtime factor*

### 6.2.2 Maximum Overtime

During assignment of a task, overtime is limited by the *maximum overtime* parameter. If this parameter is exceeded, the task will not be considered as an option. During planning, this parameter is considered a hard restriction.

### 6.2.3 Maximum Workload

The amount of time of the available standard working hours of a resource can be limited to a certain percentage. The MaxWorkload parameter indicates this percentage. The value may be higher than 100 to indicate that overtime is accepted. The workload is based on the sum of all stop durations in a trip. During planning, this parameter is considered a hard restriction.

### 6.2.4 Travel Time and Travel Distance

In general, each stop adds a cost to the plan that is relative to the travel time (in minutes) and distance (in kilometres) to reach the location of the next stop.

cost(stop) =(travel time(stop) * *travelTime* + travel distance(stop) * *travelDistance* + overlap(stop)

where

The duration of an unplanned task is not considered during the cost calculation.

### 6.2.5 Overlap

A possible time overlap with the next stop (including the second last stop that might overlap with the arrival) results in additional costs. The overlap in (minutes) is multiplied with the Overlap factor and is added to the total cost of a trip:

overlap(stop) = (time(stop) + travel time(stop) < time(next stop)) ? 0 : time(stop) + travel time(stop) - time(next stop)

### 6.2.6 Task Too Early and Task Too Late

Different stop types may have different cost parameters. For example, visits are bound to certain time limits, called *earliest* and *latest* . The Scheduler tries to place the task between these boundaries. In case this is not possible, the visit will exceed one of these boundaries which will provoke additional costs. The costs for an excess of the boundaries are calculated as follows:

excess costs(stop) = excess earliest(stop) + excess latest(stop)

where

excess earliest(stop) = (earliest(stop) > start(stop)) ? earliest(stop) - start(stop) * *TaskTooEarly*

: 0

and

excess latest(stop) = (latest(stop) < start(stop)) ? start(stop) - latest(stop) * *TaskTooLate* : 0

### 6.2.7 Best Age

Whenever a task is plannable between a earliest and latest boundary, there is usually a certain point that would be the best point for insertion. Imagine a regular task that is to be scheduled every seven weeks. Each single task will have an earliest and a latest, but getting close to the boundaries is not favorable, since the repetitive series will result in too short or too long time differences between the tasks. The best option to schedule the task would be at a certain point between earliest and latest, for example, in the sixth week.

We therefore define a parameter called *best age* which is the percentage of the latest-earliest range where the task is preferably planned. In our example above, this would be 6/7 = 86%. Any scheduling before that point would result in larger costs, any scheduling beyond this point will also result in additional costs. The absolute cost is called *Fixed* , representing the cost that is applied whenever the task is scheduled at exactly one of the boundaries. If planned at the best age, the cost is zero.

The best starting time for a task is calculated by the following formula:

best start(task) = earliest(task) + (latest(task) - earliest(task)) * *BestAge*/100

Setting *best age* to 0% indicates that the task should be planned as early as possible. Setting *BestAge* to 100% indicates that the task should be planned as late as possible. Costs planning outside of the *earliest/latest* boundaries is reflected by the TaskTooEarly and TaskTooLate parameters. In that case, the *BestAge* and *Fixed* parameters are not applied.

The *BestAge* cost is only considered if the *Fixed* value is not 0.

### 6.2.8 Skills and Skill Levels

Whenever a task requires a certain skill, the Scheduler only considers those resources that have this skill assigned. Resources may have different skill levels attached to the skill itself. For example, one resource is an expert for one particular machine. Another resource is recently trained and is only considered as a trainee. It is expected that the trainee will perform the task in a longer time that the expert.

A mismatch of skills may result in a very high penalty/cost. A skill level mismatch may result in lower but still considerable costs.

skill cost(visit) = skill level cost(visit) + (supports skill(resource, skill(visit))) ? 0 : *skillMismatch*

where

skill level cost(visit) = skill level too low cost(visit) + skill level too high cost(visit)

and

skill level too high cost(visit) = (skill level(resource, skill (visit)) > skill level(visit)) ?
*skillLevelTooHigh* * skill level(resource, skill (visit)) - skill level(visit)) : 0
skill level too low cost(visit) = (skill level(resource, skill (visit)) < skill level(visit)) ?

*skillLevelTooLow* * skill level(resource, skill (visit)) - skill level(visit)) : 0

This parameter is only applicable when the *use Skills* checkbox is selected.

### 6.2.9 Preferred Resources and Backup Resources

Service tasks may have preferred resources assigned. This is to increase the customer loyalty towards a constant service representative and to limit costs during the travel to unknown sites. In case the preferred resource is not available or the costs to let this resource perform the task is too high, the backup resource may perform the task. In case the backup resource is not available or the costs are too high, another resource needs to perform the task. The *preferred mismatch cost* and the *backup mismatch cost* factors indicate the costs that need to be calculated when a task is not performed by a requested resource.

preferred resource cost(visit) = (resource(visit) = preferred resource(task(visit))) ? 0 : *preferred mismatch cost* + backup resource cost(visit)

and

backup resource cost(visit) = (resource(visit) = backup resource(task(visit))) ? 0 : *backup mismatch cost*

### 6.2.10 Opening Hours

Costs may occur whenever a task is planned outside of the customer's opening hours. Starting the service outside opening hours is prohibitive. Although in some cases the resource may start the task within the opening hours, it may be possible to extend the service beyond the opening hours. The *endsOutsideOpeningHours* factor and *startsOutsideOpeningHours* factor reflect the costs that are applied whenever visiting a customer outside the opening hours of that day.

opening hours cost(visit) = (starts during opening hours(start(visit), day(visit))) ? opening hours excess cost(visit) : *startsOutsideOpeningHours*

where

opening hours excess cost(visit) = (ends during opening hours(end(visit), day(visit))) ? 0 : (end(visit) - end(opening hour(visit)) * *endOutsideOpeningHours*

This parameter is only valid if the *use Opening Hours* checkbox is selected.

### 6.2.11 Planned Shift Duration

This is particularly useful with opening hours to plan visits rather in the morning than in the afternoon since the costs for a visit later in the afternoon results in a long shift duration. The planned shift duration is calculated as the amount of minutes from the departure of a trip to the last visit for that day. Each minute of the planned shift duration is multiplied with the *PlannedShiftDuration* .

### 6.2.12 Excessive Weight and Volume

Each resource has a default maximum weight and volume it can carry (in a car or a truck). Any weight or volume surplus results in additional costs as defined by the *ExcessiveWright* and *ExcessiveVolume* parameters.

The parameter is only applicable if the *use Weight* and *use Volume* preferences are selected.

# 7. SCHEDULER PLANNING ALGORITHMS

## 7.1 General

The Scheduler offers a set of different planning algorithms. It depends on the planning scenario and the customer's requirements to find the most appropriate planning algorithm or algorithm combination.

Most algorithm can both be used in visual and batch mode. However, the behaviour will vary slightly depending on usage. For example, in Visual mode a selection of tasks, days and resources can be made, which certain algorithm will take into account. In batch mode, such selections are not possible. However, in batch mode certain parameters giving more control over other planning aspects are available. Therefore, the decision on which algorithms to use and in which scenarios requires careful consideration.

Note
The algorithms below are actually a bit more complex as described, but it should give an idea of the general approach.

### 7.1.1 Insertion points

Most algorithms rely on the principle of insertion points. An insertion point is a viable option for inserting a task into a certain trip at a certain cost. Often, the cheapest insertion points are searched for, meaning the point where the overall cost of the planning will increase the least.

Computing insertion points is done by several algorithms, either directly or through the use of another algorithm. There are certain preferences which influence this process.

### 7.1.2 Trip Optimization

Optimization is the process of rescheduling tasks so the overall cost of the planning will decrease. Optimization can occur within trips (changing the order), or across trips (tasks may be moved to other trips and/or resources).

There are two techniques used for optimization. The first use TSP solvers, and are mainly concerned with optimizing distances. The other class uses k-opting, where using heuristics tasks are selected and inserted into other locations, hopefully leading to better results.

Both techniques always make sure that the result will meet the constraints. This may lead to contradiction. For example, a TSP will minimize the travelled distances. However, opening hours for example may make it impossible to visit two locations shortly after another. Because of this, the optimization algorithms may become 'stuck' in a local optimum. In those cases, the planner may need to intervene.

### 7.1.3 Algorithms used in Visual mode

The following algorithms are available within the scheduler. Some algorithm are available only in batch mode, others only in visual mode, and some in both modes. The algorithms behavior may vary depending on the mode in which they are used. For example, in visual mode some algorithms take the currently selected resources, tasks and days into account, while in batch mode these are ignored. Also, in batch mode certain parameter can be specified that are not available in visual mode.

This table explains which algorithms are used when accessing functions (right mouse click) from the UI.

#### 7.1.3.1 Unplanned tasks

| | |
|---|---|
| Advice | PlanAdvice (selected tasks) |

| | |
|---|---|
| Cheapest | PlanCheapest (selected tasks) |
| Plan Day | PlanForChartable(selected tasks) |
| Plan Resource | ALGORITHM_CONFIRM_INSERT = true:<br>PlanAdvice (selected tasks, selected resources)<br>ALGORITHM_CONFIRM_INSERT = false:<br>PlanForChartable (selected tasks) |
| Fixed Appointment | Uses no algorithm, but adds the selected tasks (1) to the selected resource (1) as a fixed appointment. |
| Nearest First | NearestFurthest(selected tasks, all resources) |
| SwoonMix | Swoonmix(selected tasks, selected resources) |
| Chinese Postman | ChinesePostman(selected tasks) |
| Location Planning | LocationPlanning(all trips) |
| Fast Lane | FastLane(selected tasks) |
| VRPTW | VRPTW(all tasks) |
| Cluster -> With k-means partitioning | KMeans(selected tasks, selected territory) |
| Cluster -> PC7 | PC7(selected tasks/task groups) |
| Cluster -> Service Stop Calculator | ServiceStopCalculator(entire plan) |

#### 7.1.3.2 Resource / Resource container

| | |
|---|---|
| Recalculate | NormalizeTrip(all resources in container) |
| Optimize | OptimizeTrips(all trips for selected resources) |
| Unplan | Unplan(all trips for selected resources) |
| Unplan Peaks | UnplanPeaks(all trips for selected resources) |
| Fill Trip | FillTrips(all trips for selected resources) |

#### 7.1.3.3 Day

| | |
|---|---|
| Recalculate | NormalizeTrip(all trips for that day) |
| Unplan | Unplan(all trips that day) |
| Unplan Peaks | UnplanPeaks(all trips that day) |
| Optimze -> Optimize | OptimizeTrips(all trips that day) |
| Optimize -> Across Trips | Optimize2Opt(all trips for this day) |
| Optimize -> Replan All | Replan(all trips for this day) |
| Optimize -> Starting this day | Replan(all days starting from this day) |

## 7.2 Algorithms

### 7.2.1 PlanAdvice

This algorithm calculates the cheapest insertion points for a task. It considers the costs for adding this task to all plan day trips. The list of options is sorted by costs and is limited to a maximum number of options. Whenever an option is found that is better (cheaper) than the current maximum value, this option is added to the list, removing the most expensive one. If no options are found, the list of options remains unchanged. Finally, the remaining options are presented to the user in a dialog where a choice for one of the options can be made. In that same dialog, the user can opt to ignore certain constraints, so possibly more options are returned.

In visual mode, the territory of the task is checked against the selected territory. If these don't match, the task will be skipped. In batch mode, tasks can be filtered on several criteria.

Optional batch mode parameters:

| Name | Description |
|---|---|
| sort=latest | Sorts all the tasks on "Latest", so tasks that need to finish sooner are planned first. |
| max=<response time in minutes> | Checks if the task needs to be serviced within x minutes from now. If not, the task will be skipped. |
| required=true | Checks if the task has a required resource. If not, the task will be skipped. |
| contractType=<type> | Checks if the task has <type> contract type. If not, the task will be skipped. |

### 7.2.2 PlanCheapest

This algorithm loops through the tasks and obtains for each task the cheapest insertion point using the PlanAdvice algorithm. If an insertion point was found, the algorithm will ask the user for confirmation, unless the 'ALGORITHM_CONFIRM_INSERT' preference is not set, or the 'silent' command line option is used. If the option was accepted or no confirmation required, the insertion point is applied and the task is added to the planning.

### 7.2.3 PlanForChartable

This algorithm works similar to PlanCheapest, only it is applied to either resources or days. Also, when no suitable insertion point was found, and we are running in visual mode, the algorithm will ask if a retry should be attempted ignoring any hard constraints. If a match is found with these settings, it will be applied automatically and the task is added to the planning.

### 7.2.4 NearestFurthest

This algorithm loops through all resources. First, it will assign either the nearest or furthest tasks (in terms of location) to the resource. Next, all tasks that are closest to that task are added, until there are no more tasks to plan or the resource is full. This algorithm uses the PlanForChartable algorithm for inserting the tasks at the cheapest position in the planning.

### 7.2.5 Swoonmix

This algorithm is an extension of the NearestFurthest algorithm. It works on partitions. A partition is a grouping of tasks, based on either the territory or computed clusters. First, any fixed appointments are planned. Next, the most difficult task for the first partition is selected and assigned. Usually, the furthest task to the nearest resource is chosen. After the first assignment, the nearest task to the previous one is evaluated. If the best option for this task is within the current trip, i.e. the trip to which the furthest task has been added, then the task

is added to the trip and so forth until the trip is filled. If one of the nearest tasks has a better option within another trip, the filling of the current trip is stopped. In the next loop, the partition is again assigned the furthest task and so forth until all plan days are filled or all tasks for this resource are planned. The algorithm performs this for each partition individually and therefore also works well for one-to-many partition/resources. Depending on the preferences, either pre-defined territories or calculated clusters are used as partitions.

### 7.2.6 LocationPlanning

In some cases, it is useful to plan maintenance tasks nearby a location that is currently serviced. This avoids travel time and groups tasks geographically. The Location Planning algorithm evaluates each trip and checks if there are unplanned tasks nearby any of the stops of the trip. If possible, the task is added to the trip at the cheapest insertion point. This process is continued until no nearby unplanned tasks are available anymore.

The Location Planning algorithm allows a definition of a radius in meters to define tasks that are considered as *nearby*. A value of 0 means that only tasks on the same location are considered during this algorithm. A value of -1 indicates that any unplanned task is considered as a nearby task. It is also possible to use the *preferred resource* as a criteria for the planning. Furthermore, the algorithm may or may not continue planning nearby tasks on the next day if the current day is full or overloaded.

Note that there is also the option to specify the task group as a criterion for location planning. This is useful in situations where the task group is, for example, the location of the task. Hence, all tasks that are part of a group and on the same location are considered for location planning. Note that the location task will not be considered if the technician does not have the appropriate skill.

### 7.2.7 FastLane

This algorithm immediately adds all unplanned tasks that are close to the *latest* time limit to the planning. The numeric criteria is defined in the preferences (tab Algorithms, section Fast Lane). The preference allows the definition of the number of minutes that is subtracted from the *latest* time limit. Any task within this range is considered valid for this algorithm. In case a value of 0 is used, the algorithm will only consider those tasks that have passed the *latest* time limit.

Note that for this algorithm *latest* needs to be present.

### 7.2.8 OptimizeTrips

This algorithm loops through all trips, and optimizes the order of the visits within the trip. Also, a normalization is performed. The algorithm uses either a TSP or k-opt algorithm, depending on the preference *OPTIMIZE_TRIP_WITH_TSP*. The optimization will only change the order of stops in a trip, it will not move stops to other trips (use Replan for that).

The TSP optimizer works by optimizing the order of visits based on its location, minimizing the distance of the trip. The k-opt algorithm works by looping through all stops in the trip, and trying to reinsert it into a better position. Both algorithms take the planning constraints into account. Therefore, the result is often a trade-off between optimal distance and honoring constraints such as opening hours.

When performing optimizations, it will often help to relax the constraints temporarily. This helps the algorithm in finding a new local optimum, even when the results temporarily get worse in terms of cost. Especially MaxW*orkload* and *MaxOvertime* have a big impact on the results.

### 7.2.9 Optimize2Opt

This algorithm attempts exchanges of stops between trips, and checks if the overall costs as a result decrease. The candidate trips are selected based on distance between each other, or

rather the resources that service them. It is possible to limit the number of trips taken into account with the *OPTIMIZE_ACROSS_NEIGHBOURS_LIMIT* preference.

### 7.2.10 Replan

This algorithm looks at all currently planned tasks, and tries to find a cheaper position for it somewhere else in the planning. Tasks may be moved to other resources and/or other plan days. Internally, it uses PlanAdvice.

### 7.2.11 NormalizeTrips

This algorithm loops through all trips and recalculates the times and durations of the stops. It will attempt to remove any overlap between stops. When the scheduled date times are in the past, the stops are moved to the time line. When resources are travelling or stops are in execution, the estimated durations will be updated with the actual durations.

Optional batch mode parameters:

| Name | Description |
|---|---|
| today=true | If set, the normalization will only run for today's trips. |
| startToday=true | Deprecated |

### 7.2.12 Unplan

This algorithm simply unplans any planned tasks. It is mainly used with batch planning, when the planning needs to be reconstructed entirely.

Optional batch mode parameters:

| Name | Description |
|---|---|
| startToday=true | Only tasks planned on or after today must be unplanned. |

### 7.2.13 UnplanPeaks

This algorithm loops through trips, and checks if the trip is overloaded. A trip is overloaded if it has excessive workload or overtime. If the trip is overloaded, it will unplan the last visit until no overload exists, or the trip is empty.

### 7.2.14 FillTrips

This algorithm builds trips by adding tasks that are closest to the resource or other stops in that trip. This results in trips that are at first very efficient, but degrade gradually, when tasks that are located far away are taken into account.

This algorithm can run both in batch and visual mode. In visual mode, the user can choose to sort on latest instead of distance, and limit the distance.

Optional batch mode parameters:

| Name | Description |
|---|---|
| sortByNearest=true | When set, all tasks will be processed in order of distance to the current trip. |

### 7.2.15 BatchScript

The BatchScript algorithm is a container for defining and executing multiple algorithms sequentially. The definition of which algorithms to run and their parameters is given in the form of an XML file.

The XML file has the following structure:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <Batch name="NiemSurf">
3.    <Algorithm name="Unplan"/>
4.    <Algorithm name="SwoonMix">
5.      <Property name="sort" value="latest"/>
6.    </Algorithm>
7.    <Algorithm name="OptimizeTrips"/>
8.    <Algorithm name="Replan">
9.      <Property name="try3Opt" value="true"/>
10.   </Algorithm>
11. </Batch>
```

The name of the script file to run can be specified using the *script* command line parameter. When this parameter is not present, and the scheduler is run in visual mode, the user can choose from a list of available files. In HTTP mode, this list comes from the server, in non-HTTP mode it comes from the working directory. If running in batch mode through HTTP, the *script* parameter must be used. If running in batch mode in non-HTTP mode, the default file batch.xml will be used.

#### 7.2.16 VRPTW

Tbd.

#### 7.2.17 TripSwapper

Tbd.

#### 7.2.18 Balancer

Tbd.

#### 7.2.19 Divider

Tbd.

#### 7.2.20 ChinesePostman

Tbd.

#### 7.2.21 PC7

Tbd.

#### 7.2.22 ServiceStopCalculator

Tbd.

#### 7.2.23 RequiredResource

Tbd.

#### 7.2.24 Fixed

Tbd.

#### 7.2.25 Response

Tbd.

#### 7.2.26 BatchReload

Tbd.

#### 7.2.27 BatchRefresh

Tbd.

### 7.2.28 GroupEstimate

Tbd.

### 7.2.29 SplitTasks

Tbd.

### 7.2.30 ActualResourcePosition

Tbd.

# 8. INTEGRATION

The following sections describe the schemas that need to be used when exchanging data in XML and SQL mode. Before implementing an interface to a customer's ERP system, it is important to read this section carefully. Make sure right from the beginning that the ERP system is capable of providing all required data. If not, the Scheduler might be unable to perform the required task correctly. Any false address or invalid calendar data may corrupt the planning and may turn the result of the planning into unusable data. It is therefore important to validate any provided data before sending it to the planning engine.

## 8.1 Scheduler XML Schema

The schedulers data model is defined in the XML schema 'plan.xsd'. All interface methods to/from the scheduler adhere to this model. The schema is an important part of the documentation of the scheduler, and should therefore be studied before attempting to implement the interface.

## 8.2 Interface modes

The scheduler supports several methods of interfacing to databases or other systems. Which method to choose depends on the scenario in which the scheduler is to be used. The scheduler allows separate interface modes for reading and writing, though this is not often used.

### 8.2.1 File based interfaces

File based interfaces use files for loading and saving the data. This mode is often used in stand-alone scenario's, where a the scheduler is used as a dedicated tool. For high volume or transaction based scenario's, the SQL interface is more appropriate.

#### 8.2.1.1 XML

Planning data can be provided in an XML file. The Scheduler schema describes the detailed definitions of the data.

Note that the XML file can be opened manually with the Visual Scheduler through the File->Open menu or with the urlin command line option (both with Visual Scheduler and Batch Scheduler).

#### 8.2.1.2 Microsoft Excel (XLS)

The Excel adapter allows to read and write Microsoft Excel documents with planning data. The format of the Excel document must conform to the Scheduler schema definition, i.e. all column headers must identify a specific attribute. Furthermore, date and time fields such as earliest and latest must have the Excel date format YYYY-MM-DD hh:mm. Each worksheet corresponds to one of the Scheduler elements.

Note that the XML file can be opened manually with the Visual Scheduler through the File -> Open menu or with the urlin command line option (both with Visual Scheduler and Batch Scheduler).

### 8.2.2 SQL based interfaces

SQL based interfaces use an RDBMS for loading and saving data. This mode is the better choice in high volume and/or transaction based environments. To implement a SQL interface, the database must provide a set of tables or views and (optionally) stored functions.

Currently, both Microsoft SQL Server 2003 and later and Oracle 10 or later are supported. Other database vendors may work but require additional testing. Make sure that the appropriate JDBC driver is present in the class path.

All SQL parameters need to be defined in the preferences file.

#### 8.2.2.1 Reading data

There are several tables that the database should implement. If using an abstract layer on top of an existing database model, one may implement views to provide data in the required format. We therefore refer to views in this documentation. The views that provide tasks and resources are mandatory, all other views are optional. The DDL template can be used as a starting point for the views.

Make sure that all plannable tasks and tasks that are currently planned will appear in the task view. Even if a task is closed today, it should appear in the view to inform the planner about the change. Tasks that are closed before today are not relevant for the planner. You may choose to provide unplannable tasks if they are in a wait state. This will inform the planner about upcoming tasks.

#### 8.2.2.2 Writing data

Writing data back to the database is accomplished either trough the views or through a set of stored functions. When the stored functions are not provided, the scheduler will automatically use the views. The function template can be used as a starting point for the functions.

Make sure that the functions return an error code if they were not able to perform a lock on the required row. This is to avoid a hanging process if several users are working with the same data.

#### 8.2.2.3 Transactions

The scheduler can optionally use transactions when writing data to the database. When transactions are used, all data is kept in memory until they are explicitly committed. If an error occurs during the commit, all transactions are rolled back.

When transactions are not used, all changes will be written directly to the database. This can either be through the views or through the stored functions.

#### 8.2.2.4 Authentication

By default, the configured username / password are used to authenticate with the database. In some cases, more fine grained authentication is needed, based on the current logged in user. In those cases an initialization function on the database can be called, with the application users name and password.

#### 8.2.2.5 Connection checking

The scheduler will create the required database connections at startup, and re-use them for all database operations. In some case, the connection may be lost, usually because of network equipment dropping the connection, or processes killing the connection. In environments where this is known to occur, active connection checking can be switched on.

In this mode, each time before the connection is about to be used, a simple select statement is executed to validate the connection. Should the connection have been dropped, the scheduler will automatically attempt to reconnect to the database. Should all reconnect attempts fail, the current database operation will fail. When a new database operation is started, the scheduler will again attempt to re-establish the connection.
Note that if a connection cannot be established during startup, the scheduler will terminate.

## 8.3 Multi-user aspects

The scheduler does not support concurrent write access to the same data by multiple users. In file based mode, this may lead to inaccessible or corrupt data. In SQL mode, this may lead to users overwriting each others data. When deploying the scheduler in such configurations, special measure need to be taken when designing the interfaces, and user must be instructed on how to prevent problems.

Also, the scheduler will not automatically see changes to the data made by other users until a reload or refresh is performed.

# 9. USING THE WEBSERVER

The Scheduler Webserver allows the use of the scheduler application through Webstart or an Applet. The webserver acts as a proxy to the actual data and for the mapping environment, ensuring that the Single Origin Policy is followed. The REST interface provides methods for retrieving, inserting, updating or deleting planning information. It also provides a proxy for mapping related functions. When using the default mapping environment, it must be installed on or be reachable from the webserver.

When the webserver starts, it mounts the 'web' directory in the product directory. This allows the webserver to serve any static content such as JNLP files, without the need for a separate webserver such as Apache or IIS.

## 9.1 Authentication

If running the web server against a SQL database, authentication will be based on the initializationSf method provided with the SQL interface. The username and password of the configured SQL user will be used. By default, the web server uses the BASIC authentication method.

## 9.2 GET Requests

Most requests return XML streams with elements that correspond to the Scheduler XML schema. Callers can use the Plan noun initially and navigate through the plan by specifying the child noun until they have reached the element that they want to retrieve.

- http://localhost:<port>/Plan/Territories

  Retrieves a list of all territories that are accessible to the requesting user.

- http://localhost:<port>/Plan

  Retrieves the entire plan for all territories that are accessible to the requesting user.

- http://localhost:<port>/Plan?territory1=01&territory2=02

  Retrieves the entire plan for a specific list of territories.

- http://localhost:<port>/Plan/Tasks?lastUpdate=2007-12-04+14:12

  Retrieves all tasks that have been updated since the specified date and time.

- http://localhost:<port>/Plan/Breaks?lastUpdate=2007-12-04+14:12

  Retrieves all breaks that have been updated since the specified date and time.

- http://localhost:<port>/Plan/Resource/484

  Retrieves the details of the resource with the unique identifier 484

- http://localhost:<port>/Plan/Resource/19505/Trip/2007-12-04

This request returns a list of stops of the specified trip.

- http://localhost:<port>/Plan/Resource/19505/Trip/2007-12-04/image

  This request returns an rendered image of the specified trip.

- http://localhost:<port>/Plan/Trips

  This request returns a list of all trips.

- http://localhost/Plan/Resource/1980/Trip/2005-01-18?stylesheet=<XSL document URL>

  By adding a stylesheet parameter to the URI, XSL transformations can be requested. The parameter must specify a XSL document URL, which is applied to the XML result before returning the HTTP response. It is assumed that the XSL generates HTML documents.

- http://localhost:<port>/Plan/Resource/(resourceId)/Trip/(date)?resourceId=19505&date=2007-12-04

  It is possible to use parameters for replacement of REST-identifiers. This is particularly useful when used in combination with HTML forms.

- http://localhost:<port>/Schemas

  This method returns a list of all schema names that are accessible to the Geocoder.

- http://localhost:<port>/Schema/S4/Table/Resource

  This method returns a list of resource columns that are visible to the Geocoder.

- http://localhost:<port>/Schema/S4/Table/Resource/Columns

  This method returns a list of all resources that are visible to the Geocoder.

## 9.3 POST, UPDATE and DELETE Requests

- http://localhost:<port>/Plan/Task/00029931200100000000 with an XML upload of the task that is to be inserted/updated/deleted.
- http://localhost:<port>/Plan/Resource/484/Break/1233 with an XML upload of the break that is to be inserted/updated/deleted.

# 10. RUNNING THE SCHEDULER

The scheduler is written in Java and requires a Java runtime environment to run. The It is distributed as a single JAR file called 'dist1.jar'. As of version 7.4 build 483, this JAR is code signed.

Note

When running the scheduler in a web environment, all JARs used must be code signed to avoid security warnings.

## 10.1 Java requirements

As of version 7.4, the scheduler requires a Java runtime 1.6 or newer.

## 10.2 Mapping Environment

The mapping environment needs to be up and running and configured before the scheduler can be started. Failure in doing so will result in an exception. Also, the mapping environment should be configured correct

The standard mapping environment comes with its own installers and documentation.

## 10.3 Running as an application

Running the scheduler as a normal Java application involves starting java from the command line. Because of the required command line parameters, it is often useful to create a batch file for this.

Basic command:

*java -classpath <path> <options> <main class> <command line parameters>*

| | |
|---|---|
| path | Path to dist1.jar and possibly other JAR files, such as jxl.jar and possibly custom look and feel. |
| options | Java JVM options if required (normally not the case) |
| main class | Name of the class to execute (depends on the mode of operation):<br>• batch/visual scheduler: com.crmalliance.s4.s4<br>• webserver: com.crmalliance.s4.web.server.WebServer<br>• geocoder: com.crmalliance.s4.Geocoder |
| command line parameters | Scheduler command line parameters |

Note

The directory from which the scheduler was started is called the working directory.

## 10.4 Command line parameters

The following list contains all parameters that can be used on the command line. The mode column depicts in which mode the parameter is applicable. The following modes are defined:

- Visual; scheduler runs with as an UI application.
- Batch: scheduler runs as a batch program.
- HTTP: scheduler runs as an HTTP client

- All: all modes

| Parameter | Description | Modes | Default |
|---|---|---|---|
| language=<locale><br><br>Eg: language= nl_BE | Set the language for all output and UI. The locale is determined in the following order:<br>- client machine<br>- preference "language"<br>- language parameter | All | |
| locale=<locale><br><br>Eg: locale=nl_BE | See Language | | |
| urlwork=<url><br><br>eg: urlwork="file://d/engine",<br>urlwork-="http://server:8080" | Set the working directory of the engine. URL must point to either a file path on the file system or an URL in HTTP mode. | All | current directory |
| batch | Run the scheduler in batch mode. | Batch | |
| service | Signal that the scheduler runs as a service. When set, the scheduler will not issue a System.exit() when the maximum nr of warnings is exceeded . When set, "embedded" is automatically set as well. | All | |
| embedded | Signal that the scheduler runs within a container. When set, the scheduler will not issue a System.exit() when it terminates. | All | |
| eXpert | Run scheduler in expert mode. In expert mode more options are available from the 'tools' menu, and a tab with working days is added. | Visual | |
| http, http=true | Signal that the scheduler runs in a webstart environment or as an applet. In this mode most data is retrieved using HTTP, including mapping services. | HTTP | |
| replicate=[all \| tasks \| breaks-tasks \| nothing] | obsolete. | All | nothing |
| nosplash, nosplash=true | Suppress the splash window at startup. | Visual | |
| silent, silent=true | Suppress engine output at startup. | All | |

| Parameter | Description | Modes | Default |
|---|---|---|---|
| urlin=<url><br><br>eg. urlin=file://d/plan.xml | Specify the path to the data input file when using file mode interface. | Batch | |
| urlout=<url><br><br>eg. urlout=file://d/plan.xml | Specify the path to the data output file when using file mode interface. | Batch | |
| urlcache=<url> | Specify path to mapping service cache file. Note that in HTTP mode the file cannot be saved. | All | tds_cache.xml in working directory. |
| urlsplash= <url><br><br>eg. urlsplash=http://server/about.html | Specify url to HTML page with about information. | Visual | about.html in working directory. |
| dayoffset=<nr of days> | Number of days to skip for planning starting from today. | Batch | -1 |
| nomap, nomap=true | Specify whether map is to be made unavailable to the client. | Visual | false |
| readonly, readonly=true | Specify whether the planning is to made unchangeable by the client. | All | false |
| noprefs, noprefs=true | Specify whether the preferences editor is to be made unavailable to the client. | Visual | false |
| wait=<sec> | Nr of seconds to pause before starting the engine. | All | 0 |
| JSESSIONID=<sessionid> | Session id to use when starting the scheduler from an existing session. The username and password are taken from the session for authenticating against the server. | HTTP | <n.a.> |
| changeToBasicAuthentication = [true \| false] | Switch from form-based to basic authentication. See changeToBasicAuthenticationContext. | HTTP | <n.a.> |
| changeToBasicAuthenticationContext = <context name> | Context to match against when authenticating from session id. The code base is retrieved, and calls to <code base>/session are made to retrieve the username and password. | HTTP | <n.a.> |
| username | Username for authenticating against the server. | HTTP | <n.a.> |

| Parameter | Description | Modes | Default |
|---|---|---|---|
| password | Password for authenticating against the server. | HTTP | <n.a.> |
| script=<url><br><br>script = file://d/batch.xml | Specify the name of the batch script file to use. | Batch | <n.a.> |
| googleMapsProxy=[true \| false] | Specify whether to proxy all Google Maps calls through the server proxy, or access Google Maps directly from the local machine. | HTTP | <n.a.> |
| mapsSuiteProxy = [true \| false] | Specify whether to proxy all Infoware calls through the server proxy, or access Infoware directly from the local machine. | HTTP | <n.a.> |
| laf=<class name> | Specify the Swing LookAndFeel class when using a custom look and feel. The class must be on the class path. | Visual | <n.a.> |
| territories = [select \| all] | Specify whether the user must select the territories to be loaded. | All | all |
| noload | Specify that the data should NOT be loaded by the engine, usually when running in Service mode. | All | <n.a.> |
| A.1.1 nosave = [true \| false] | A.1.2 Hide the save button from the tool bar. Since the recommended way of saving changes in a concurrent HTTP-environment is the refresh-button, the save-button might be obsolete. | A.1.3 H T T P | A.1.4 false |
| A.1.5 nocache = [true \| false] | A.1.6 Inidicates if the geocode cache is to be used. | A.1.7 A l l | A.1.8 false |

## 10.5 Running as a Java Applet or Webstart application

Running the scheduler as a Java webstart application or Java applet involves the use of a JNLP file. This JNLP file can either be loaded explicitly in a browser to start the Java application, or implicitly to embed an applet in a HTML page. The <argument> element in the JNLP can be used to specify any parameters that would normally have been provided on the command line.

http://www.oracle.com/technetwork/java/javase/javawebstart/index.html

http://docs.oracle.com/javase/tutorial/deployment/deploymentInDepth/jnlp.html

# 11. CUSTOMIZATION

## 11.1 Custom task data

The Scheduler allows the customization of task data. By splitting the remarks field into various sections, information can be divided into separate fields. These fields occur in a separate tab of the task dialog. Editable fields are updateable by the user. In that case, updates will adapt the section in the remarks field accordingly. Administrative systems may reuse this information to trigger other activities. Note that not all administrative systems allow updates of custom information.

Custom information is specified in the preference file, gathered in a CustomSplit tag. Each section must contain a unique name, a description, a start and end position, a data type and a flag to indicate whether the field is updateable by the user. The confirm attribute indicates that the user is signalled with an information icon that there is additional information to be viewed. Furthermore, attributes width and height can be specified to determine the preferred size of the field in the form and the displayInRemarks flag is used to specify if the section is to be displayed in the standard remarks field. Default for this flag is true, all sections are displayed in the remarks field by default.

The skip attribute allows customers to remove a leading portion of a specific remarks section. The attribute value must be an integer, specifying the number of leading characters that need to be skipped.

Example

```
<CustomSplit>
   <Section code="Symptom" description="The symptom information." start="0" end="80"
editable="false" width="300" height="23" displayInRemarks="false" type="xs:string"/>
   <Section code="Call Center" description="The information that has been provided by the
Call Center Agent." start="80" end="160" editable="false" type="xs:string" confirm="true"/>
   <Section code="Planner" description="The information that has been added by the
Planner." start="160" end="2000" editable="true" type="xs:string" skip="20"/>
</CustomSplit>
```

## 11.2 Gantt information format

If existent, the Scheduler uses a standard XSL stylesheet to format the layout of the visit info in the Gantt chart. If no stylesheet is found, a standard layout will be used. The stylesheet must be called gantt.xsl and must be placed into the product directory.

Below is an example of a stylesheet that displays detailed task information:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:s4="http://www.oracle.com/XSL/Transform/java/com.crmalliance.common.GenString">
<xsl:output method="text" encoding="ISO-8859-1"/>
<xsl:decimal-format name="numbers" decimal-separator=","
grouping-separator="."/>
<xsl:template match="/Task"><xsl:value-of select="s4:fromXMLString(@name)"/>
<xsl:value-of select="s4:newLine()"/>
<xsl:value-of select="@postalCode"/>
<xsl:value-of select="s4:newLine()"/>
```

```
<xsl:value-of select="@city"/>
<xsl:value-of select="s4:newLine()"/>
<xsl:value-of select="s4:substring(@id, 3, 9)"/>
<xsl:text> </xsl:text>
<xsl:value-of select="s4:substring(@id, 9, 12)"/>
<xsl:value-of select="s4:newLine()"/>
<xsl:value-of select="@skill"/>
<xsl:value-of select="s4:newLine()"/>
<xsl:value-of select="@contractType"/>
</xsl:template>
</xsl:stylesheet>
```

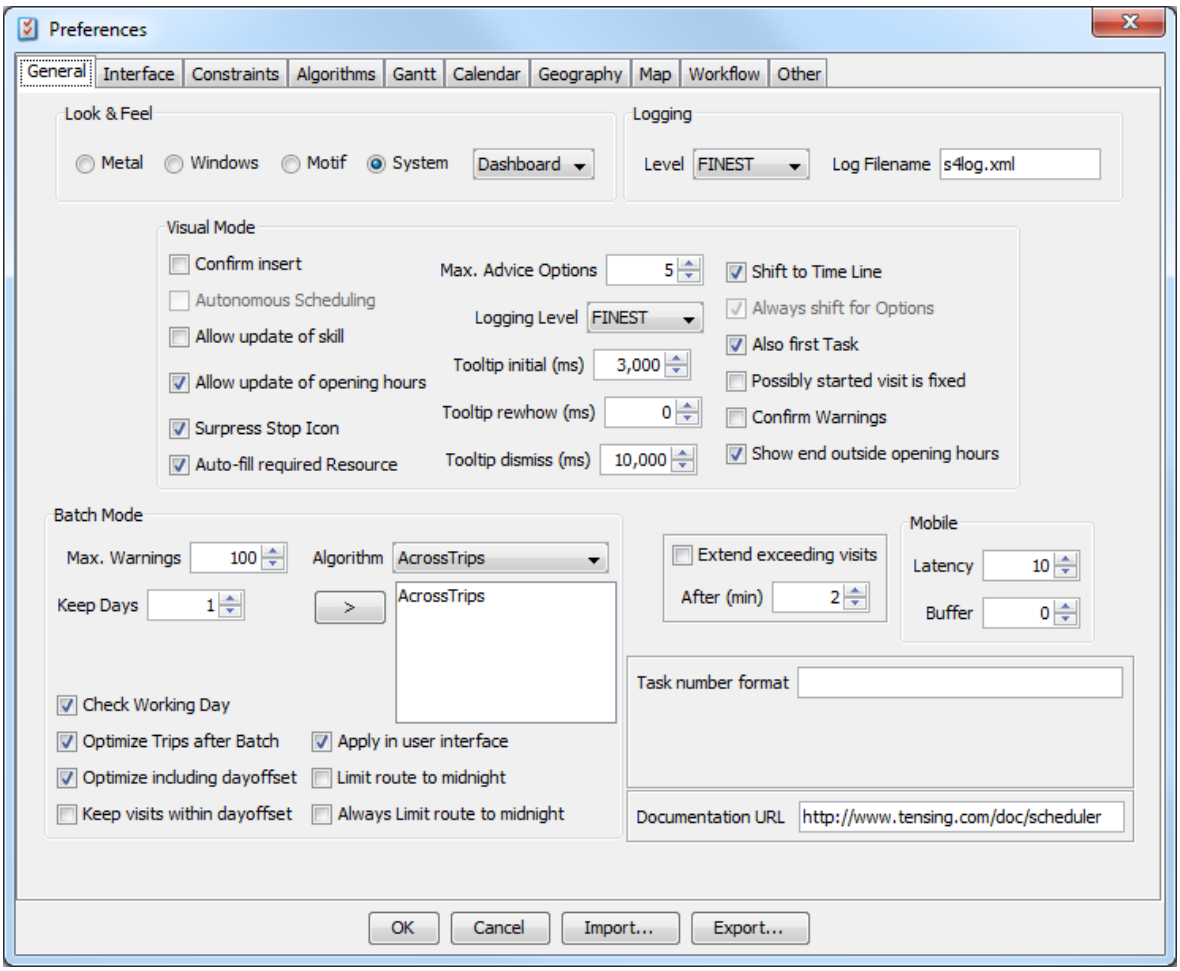
To insert a task attribute, use the @ notation. Available task attributes are: name, territory, department, street, postalCode, city, country, skill, estimatedDuration, earliest, latest, scheduled, scheduledFor, requiredResource, preferredResource, backupResource, fixed, status, actualDateTime, actualDuration, openingHoursMon, openingHoursTue, openingHoursWed, openingHoursThu, openingHoursFri, openingHoursSat, openingHoursSun, remarks, weight, geocode, volume, lastUpdate, messageConfirmed, contractType, requiresParts, holidayCalendarId, planAutomatically, assignedDateTime, minimumResources, maximumResources.

## 11.3 Look and Feel

TBD

# 12. PREFERENCES

## 12.1 General




### 12.1.1 Look & Feel

- The radio buttons select the theme to use.
- The dropdown list must be set to Dashboard.

### 12.1.2 Logging

| | |
|---|---|
| Level | Logging level for log file. Default: INFO |
| Log file | Filename for logging. Logging information is stored in XML format. |

### 12.1.3 Visual Mode

| | |
|---|---|
| Confirm insert | When using Swoonmix: if switched on, whenever a task cannot be planned the advice dialiog is shown. If switched off, Swoonmix will quit.<br>When using 'Plan resource': if switched on, PlanAdvice algorithm is used. Otherwise, PlanForChartable algorithm is used. |

| | |
|---|---|
| Autonomous scheduling | obsolete |
| Allow update of skill | If set, skills can be changed in the task form. |
| Allow update of opening hours | obsolete |
| A.1.1 Suppress stop icon | If set, the pin icon will not be displayed in the tasks. |
| Auto-fill required resource | Is used in the Task form to fill the resource if the task is yet unplanned. |
| Max. advice options | Maximum number of advices retrieved by the PlanAdvice algorithm. |
| Logging level | Logging level for log tab. |
| Tooltip initial (ms) | Number of ms before a tooltip is displayed. |
| Tooltip reshow (ms) | Number of ms before a tooltip is redisplayed. |
| Tooltip dismiss (ms) | Number of ms before a tooltip is hidden. |
| Shift to time line | If set, a tasks that is moved manually to a time in the past will be moved to the current time. |
| Always shift for options | If set, tasks will automatically be moved to the time line when computing insertion points. |
| Also first task | If set, tasks will automatically be moved to the time line when normalizing. |
| Possibly started visit is fixed | A.1.2 If set, a task that according to status and/or the planning information should have started, will be treated as fixed, meaning that it will not be movable anymore. |
| Confirm warnings | If set, warnings displayed in the warnings tab can be confirmed using a checkbox. The value is not persisted. |
| Show end outside opening hours | If set, a warning will be given in the task info when the task ends outside of the opening hours. |

### 12.1.4 Batch Mode

| | |
|---|---|
| Max. warnings | Number of warnings after which the batch will be terminated. |
| Keep days | Number of days to keep after running algorithms. All other days will be unplanned.<br>-1: keep everything from dayOffset on<br>0: keep nothing<br>n: keep n days from dayOffset on |
| Check working day | If set, executes the batch only if today is a working day, i.e. not a public holiday and possibly not a weekend. |
| Optimize trips after batch | Optimize all trips after completing the batch, provided they are not unplanned. |
| Optimize including day offset | Include all trips within the dayOffset when optimizing. |
| Keep visits within day offset | Keep all days within the dayOffset |

| | |
|---|---|
| Apply in user interface | When set, all algorithms configured will run when using the automatic planning button. If not set, the Swoonmix algorithm will run. |
| Limit route to midnight | Limit all stops that are planned beyond midnight to 23:59 on the routes starting day. |
| Always limit route to midnight | Normally, limit to midnight will only work when 1) limit route to midnight is set, and 2) 'accept beyond latest day' is not set. This option overrules the second condition. |

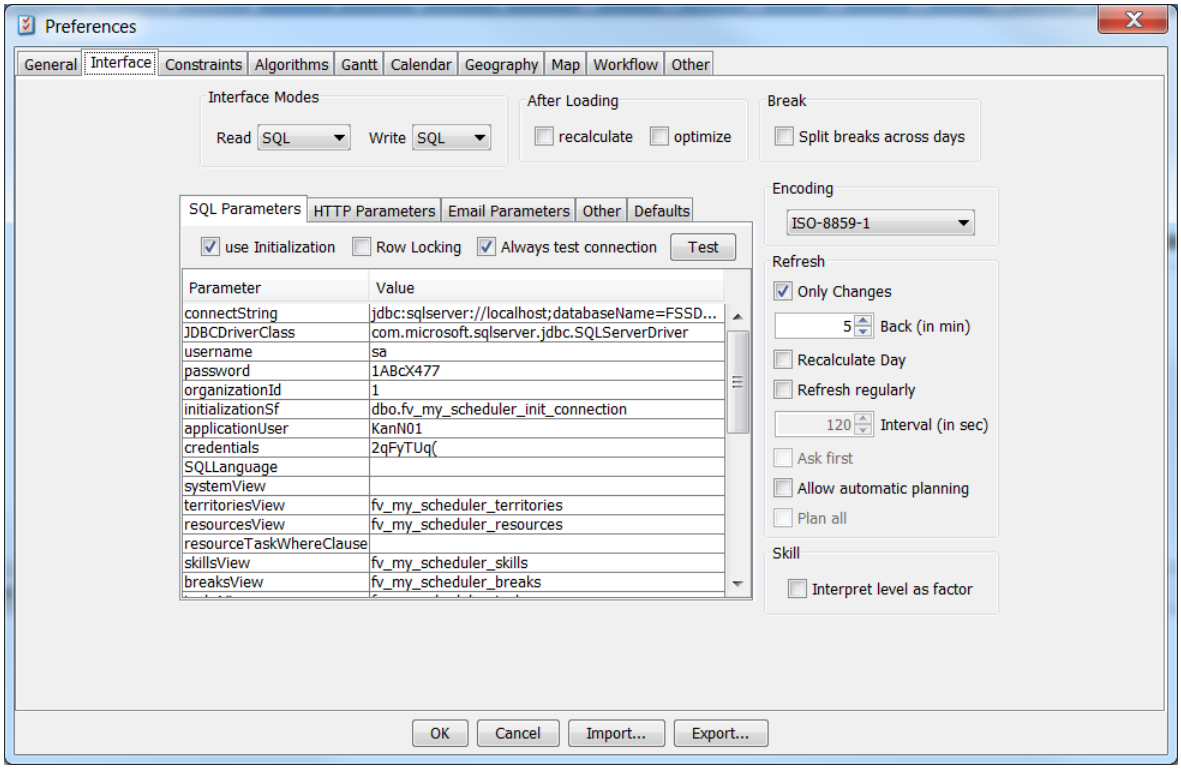
### 12.1.5 Mobile

| | |
|---|---|
| Latency | Nr of minutes to compensate for delays in communication from the field. |
| Buffer | obsolete |

### 12.1.6 Other

| | |
|---|---|
| Task number format | Formats the task ID string according to the format pattern. A X in the simple format removes the character at that position. A underscore (_) character copies the appropriate character, whereas any other character will be inserted at the specified position. For example, a input text of 0033459600100000000 and a simple format of XXX#####-###XXXXXXXX will result in 334596-1. |
| Documentation URL | URL pointing to the online documentation. |
| Extend exceeding visits | If set, the end time of visits may be adjusted to the time line when they are running late. |
| After (min) | The number of minutes after which to adjust the time line for exceeding visits. |

## 12.2 Interface




### 12.2.1 Interface modes

| | |
|---|---|
| Read | Interface mode for reading data. |
| Write | Interface mode for writing data. |

### 12.2.2 After loading

| | |
|---|---|
| Recalculate | If set, the routes will be normalized after being loaded. |
| Optimize | If set, the routes will be optimized after being loaded. |

### 12.2.3 Break

| | |
|---|---|
| Split breaks across days | If set, breaks that span multiple days will be split into multiple breaks. |

### 12.2.4 Encoding

The encoding used by XML documents.

### 12.2.5 Refresh

| | |
|---|---|
| Only changes | If set, the lastUpdate field is used to determine what information has changed since the previous update. |
| Back (in min) | Number of minutes to subtract from the lastUpdate time stamp, to compensate for latency, delays, etc. |

| | |
|---|---|
| Recalculate day | If set, the day or trips for the current territory are normalized after a refresh. |
| Refresh regularly | If set, a timer will be used to automatically refresh. |
| Interval (in sec) | When using automatic refresh, the interval in sec. |
| Ask first | If set, when using automatic refresh the user will be prompted to refresh. |
| Allow automatic planning | If set, unplanned tasks will automatically be planned after a refresh, using the current batch settings. |
| Plan all | Obsolete |

### 12.2.7 SQL Parameters - settings

| | | | |
|---|---|---|---|
| A.1.1 | Use initialization | A.1.2 | If set, for each session the initializationSf stored function will be called. This will enable the ERP system to identify the user making the call. |
| A.1.3 | Row locking | A.1.4 | If set, the scheduler will use an optimistic locking scheme for tasks and breaks. The stored function updaters will then use a version parameter, which is to be checked against the database. |
| A.1.5 | Always test connection | A.1.6 | If set, each time a connection to the database is obtained, a check will be performed to verify the connection is valid. |

### 12.2.8 SQL Parameters

| | |
|---|---|
| connectString | The JDBC connect string, identifying the database URL, for example, |
| JDBCDriverClass | The name of the JDBC driver used during the connection with the database. Make sure that the class is accessible in the system's CLASSPATH. |
| username | The name of the database account the SQL interface should connect to. |
| password | The password of the database account. |
| organizationId | The id of the organization, used in conjunction with ERP systems that differentiate several organizations in the same database instance. |
| useInitialization | If this flag is set, the Scheduler applies initializationSf in conjunction with the applicationUser and credentials parameters right after the connection is succesfully opened. If this flag is not set, the initialization function is not applied. |
| initializationSf | A stored function that initializes the connection. Used in scenarios where the user name is not equal to the application user name or in the case where some initialization needs to be performed before retrieving data from the interface views. The Scheduler calls this function with the parameters applicationUser and credentials. |

| | |
|---|---|
| applicationUser | The application user name, which may vary from the database username. In case no application user name is used, leave this field blank. |
| credentials | The credentials for the application user name. Not required if the back-office system does not make use of application users and credentials/responsibilities. |
| SQLLanguage | The SQL language, set with the SET LANGUAGE command (not supported by all databases). If the parameter is not defined, the connection language will be the default language. |
| territoriesView | The name of the view containing a name, description list of the territories that are available to the user. If this view is specified, the user is asked to choose a selection from the list of available territories before actually loading the data. The selection causes an additional clause to be added to the resource and task queries |
| resourcesView | The name of the resources view. |
| skillsView | The name of the skills view. |
| breaksView | The name of the breaks view. |
| tasksView | The name of the tasks view. |
| holidayCalendarsView | A list of dates that form together a set of public holidays are grouped in a HolidayCalendar object. If a resource is assigned to a HolidayCalendar, the public holidays of that calendar are not considered a working day. If a resource does not have a HolidayCalendar assigned, the standard public holidays as defined in the preferences will be taken into account. Different resources may be assigned to different holiday calendars. |
| publicHolidaysView | A day in which the resources that are assigned to a calendar containing the public holiday, are unavailable. |
| workPatternView | A table containing working hours for each day of the week. Working patterns can be assigned to resources to define their standard working hours. |
| workingHoursView | A definition of the working hours of a day. It is possible to specify a floating break, i.e. a pause that needs to be assured between certain hours. |
| resourceWorkPatternView | This element is used to represent a many-to-many relationship between resources and work patterns. |
| taskUpdateSf | The name of the stored function that allows to update of a task for both date/time and resource assignment. If the Scheduler is used with a non-SQL write mode, this field is not required. If using direct SQL mode, i.e. without stored functions to update planning data, this field must be empty. |
| breakInsert | The name of the stored function to insert a new break. |
| breakUpdate | The name of the stored function to update an existing break. |
| breakDelete | The name of the stored function to remove a break. |

| | |
|---|---|
| breakTypeTable | A table containing the types of breaks. If defined, the types are offered for selection when inserting or updating a break. |
| breakChargeToTable | A table containing the accounts that breaks are chargeable to. If defined, the accounts are offered for selection when inserting or updating a break. |
| schedulerResultTable | The name of the table where planning result is written to (only applicable of interface write mode is SQL) |
| UseSchedulerResultTable | Indicates whether to use the scheduler result table or whether the result should use the taskUpdateSf stored function. |
| insertPlanDaySf | If this flag is set, the plan days table is used to inform the back-office system about the relevant plan days. This parameter is deprecated. |
| deleteAllPlanDaysSf | This stored function is used to delete all plan days for the organization before inserting the current plan days. This parameter is deprecated. |
| planDaysTable | The name of the plan days table, hosting the plan days that the scheduler is prepared to plan. If not defined, it is the responsibility if the tasksView to retrieve the correct tasks for the Scheduler. In some implementations, it might be useful to first let the Scheduler create a temporary table of plan days, which can then be used in the breaksView and the tasksView to retrieve only the portion that is relevant for the period that is defined by the plan days in this table. This parameter is deprecated. |
| connectRetry | The number of times a connection attempt should be repeated before stopping. If 0 is specified, the application will not attempt to reconnect. If -1 is specified, the application will try to reconnect perpetually. |
| connectRetryWait | The number of milliseconds the application should wait before attempting to connect again. |
| StandardFetchSize | This number specifies the fetch buffer. Change this value with caution. It may have an impact on speed an memory. |

### 12.2.9 HTTP Parameters

| | |
|---|---|
| HTTPProviderURL | The base URL for the scheduler webservice. |
| HTTPSecurityPrincipal | The username when using basic authentication |
| HTTPSecurityCredentials | The password when using basic authentication |
| HTTPCompressCheckPlanCodes | If set, the update stack will take change of plan codes into account. |
| HTTPCompressCheckStatus | If set, the update stack will take change of status codes into account. |
| HTTPZipPlan | If set, the scheduler will use ZIP compression for the plan XML when possible. |
| HTTPStopReportURL | The parameterized pattern for requesting a stop report. |

| | |
|---|---|
| HTTPRouteReportURL | The parameterized pattern for requesting a route report. |

### 12.2.10 Email parameters

| | |
|---|---|
| MailServer | The mail server to use for sending emails. |
| MailProtocol | The protocol used to send emails. |
| MailPort | The port for the mail server. |
| MailAccount | The account name for authorizing with the mail server. |
| MailPassword | The password for authorizing with the mail server. |
| MailDomain | The domain which the mail server uses. |

### 12.2.11 Other

Obsolete.

### 12.2.12 Defaults

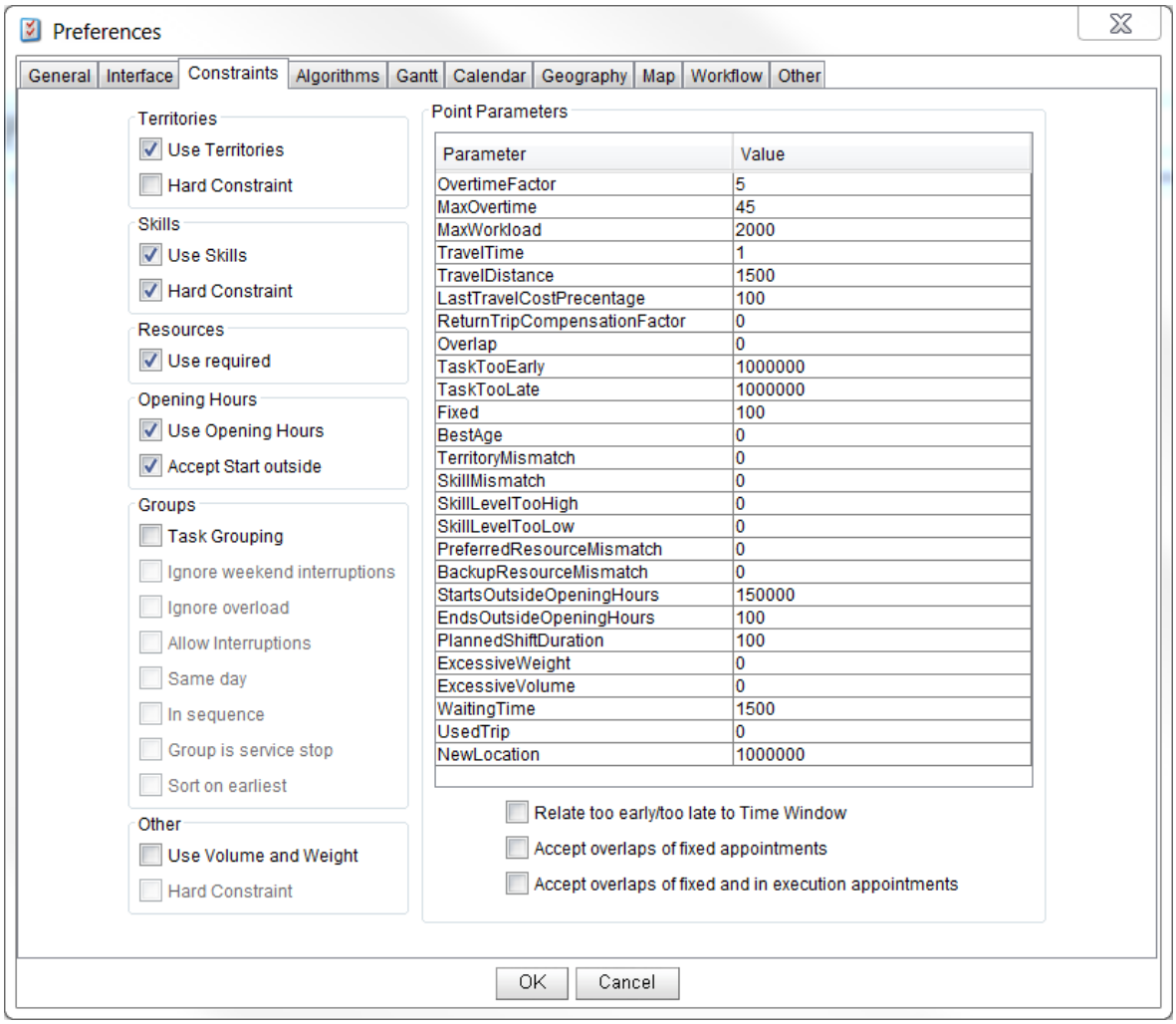
#### 12.2.12.1 Tasks

| | |
|---|---|
| Est. duration (min) | Default estimated duration in minutes. |
| Weight | Default weight. |
| Volume | Default volume. |

#### 12.2.12.2 Resources

| | |
|---|---|
| Max. weight | Max weight. |
| Max. volume | Max. volume. |

## 12.3 Constraints




### 12.3.1 Territories

| | |
|---|---|
| Use territories | If set, all tasks and resources will be assigned to the given territories. |
| Hard Constraint | If set, this is a hard constraint. |

### 12.3.2 Skills

| | |
|---|---|
| Use territories | If set, the use of skills is enabled. |
| Hard Constraint | If set, this is a hard constraint. |

### 12.3.3 Resources

| | |
|---|---|
| Use required | If set, required resources will be used. |

### 12.3.4 Opening Hours

| | |
|---|---|
| Use opening hours | If set, opening hours will be used. Note that all tasks must specify valid opening hours, otherwise all days will be considered as closed. |
| Accept start outside | If set, starting a task outside opening hours is acceptable, at a additional cost. |

### 12.3.5 Groups

| | |
|---|---|
| Task grouping | If set, task grouping is enabled (through taskGroupId). All tasks will be assigned to a task group, even when it is the only task for the group. |
| Allow interruptions | If set, it is acceptable if tasks in a taskgroup are mixed with other tasks. |
| Ignore weekend interruptions | If set, Friday to Monday gaps in planned tasks are acceptable. |
| Ignore overload | If set, possible overload will be ignored when planning tasks within a task group. This option makes sure all tasks of the group will be planned on the same day. |
| In sequence | If set, all tasks of a group must be planned in the order given by the taskgroupId. |
| Sort on earliest | If set, planning tasks that are part of a taskgroup will be based on the earliest time. If not set, it will be based on closest distance. This option enables forcing the order in which sub tasks are planned. |

### 12.3.6 Other

| | |
|---|---|
| Use volume and weight | If set, the volume and weight will be calculated per trip and compared to the resources maxima. |
| Hard constraint | If set, excessive weight and/or volume are a hard constraint. |
| PartsRequiredFixResource | If set, a task that has "requiresParts" set the following behaviour can be observed; when the task is planned for the first time, the required resource will automatically be changed to the trips' resource. Consequent moves to other resources are thus prohibited, only moves in time are allowed. This may be usefull in situations where the ERP automatically orders parts for such tasks in response to a move attachment. |

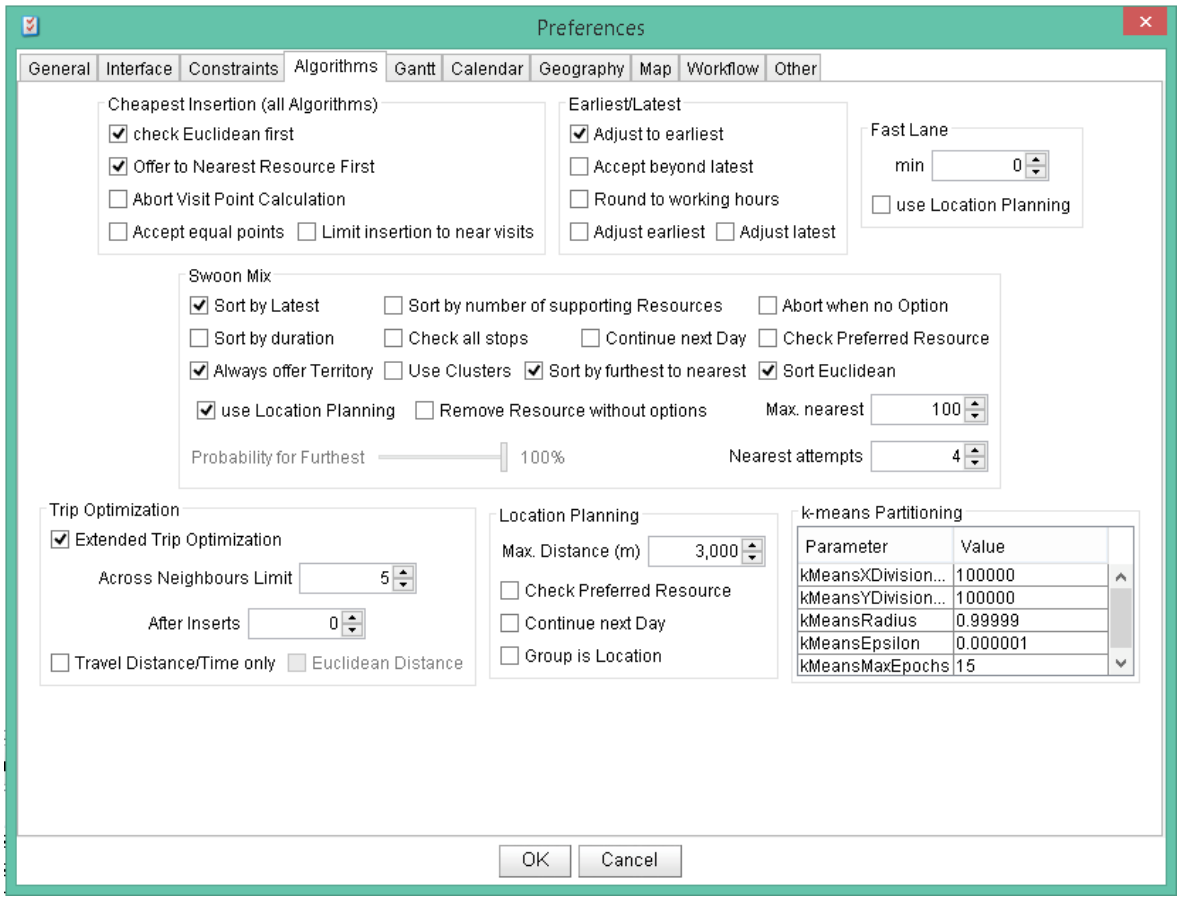
### 12.3.7 Point parameters

| | |
|---|---|
| OvertimeFactor | Multiplier for every minute a resource works beyond its nominal working hours. |

| | |
|---|---|
| MaxOvertime | Maximum acceptable overtime in minutes. Note that overtime is the time beyond the end time of the resource's work schedule. |
| MaxWorkload | Maximum acceptable workload in percentage. Note that workload is computed as the time the resource is actually performing work or travelling, not taking waiting time into account. |
| TravelTime | Multiplier for every minute a resource is travelling. Note that a relation between TravelTime and TravelDistance exists. |
| TravelDistance | Multiplier for every meter a resource travels. Assuming a resource travels at approx 50km/h, the ratio between TravelTime and TravelDistance is 60:50.000. Therefore, TravelTime should be set to approx 1000x TravelCost, or TravelDistance should be ignored (0). |
| LastTravelCostPercentage | The percentage of time and distance when travelling from the last task that should be taken into account. This option can be used when it is acceptable that the resource travels home in his/her own time. |
| ReturnTripCompensationFactor | Obsolete |
| Overlap | A.1.1 Multiplier for every minute that two tasks overlap. Note that overtime is computed as overlap from the scheduled end time. Therefore, there is a relation with OverTimeFactor and ReturnTripCompensationFactor. |
| TaskToEarly | Multiplier for every minute a tasks starts before its earliest time. |
| TaskToLate | Multiplier for every minute a tasks starts after its latest time. |
| Fixed | Fixed cost related to the BestAge setting. Set to 0 to disable BestAge computing. |
| BesteAge | Percentage which indicates where to plan the task relative to its earliest/latest. 0 means that the task should be planned as early as possible, 100 as late as possible. The Fixed costs are distributed linear. |
| TerritoryMismatch | Penalty for assigning a tasks to a resource in another territory than the one specified. |
| SkillsMismatch | Penalty for assigning a tasks to a resource without the required skill(s). |
| SkillLevelTooHigh | Penalty for assigning a task to a resource that has a higher skill level than required. Note that this only applies when using skill levels. |
| SkillLevelTooLow | Penalty for assigning a task to a resource that has a lower skill level than required. Note that this only applies when using skill levels. |
| PreferredResourceMismatch | Penalty for assigning a task to different resource than the preferred. Note that this penalty also applies when |

| | |
|---|---|
| | the task is assigned to either the required or backup resource. |
| BackupResourceMismatch | Penalty for assigning a task to different resource than the backup. Note that this penalty also applies when the task is assigned to either the required or preferred resource. |
| StartsOutsideOpeningHours | Multiplier for every minute a task is planned outside the closest opening hours. |
| EndsOutsideOpeningHours | Multiplier for every minute a tasks projected end time is outside of the closest opening hours. |
| PlannedShiftDuration | Multiplier for every minute of the length of the shift for the resource. The shift is computed as the time the resource actually starts until the time he/she actually finishes. |
| ExcessiveWeight | Multiplier for every kg excessive weight. |
| ExcessiveVolume | Multiplier for every m3 excessive volume. |
| WaitingTime | Multiplier for every minute the resource is waiting, that is not travelling or working. |
| UsedTrip | Penalty for every assignment to a trip that already has 2 or more visits planned. |
| Relate too early/too late to time window | If set, the time a task may be overdue is computed as a percentage relative to the time window between earliest/latest. If not set, the time is computed absolutely from/to the earliest/latest. |
| Accept overlap of fixed appointments | If set, it is acceptable when two fixed tasks overlap. If not set, overlap of fixed appointments is considered a violation of a hard constraint. |
| Accept overlaps of fixed and in execution appointsments | If set, it is acceptable when a fixed appointment overlaps with a previous appointment, if that appointment is currently in execution. If not set, overlap of a fixed /non-fixed appointments is considered a violation of a hard constraint. |

## 12.4 Algorithms




### 12.4.1 Cheapest Insertion (all algorithms)

| | |
|---|---|
| Check Euclidean first | If set, compuation of insertion points (costs) is based on Euclidean distances. |
| Offer to nearest resource first | If set, first resources that are considered closest to the task are considered. The number of resources is determined by 'Across Neighbours Limit' in Trip Optimization settings. When an option was found with one of these resources, no other resources will be tried, unless 'Always offer territory' is set. |
| Abort visit point calculation | Obsolete |
| Accept Equal points | If set, options with similar cost values are acceptable, so the user may select one manually. |
| Limit insertion to near visits | If set, when considering insertion into an existing trip only the N nearest visits in that trip are considered. N is determined by the 'Across neighbors limit' setting in 'Trip Optimization'. |

### 12.4.2 Earliest/latest

| | |
|---|---|
| Adjust to earliest | If set, when a task is planned too early, it will be moved to the time specified by its earliest. |

| | |
|---|---|
| Round to working hours | If set, all visit times will be rounded to minutes. |
| Accept beyond latest | If set, it is acceptable that tasks are planned too late, at an additional cost. |
| Adjust earliest | When scheduling a task before its ealiest execution date, the earliest can be adjusted to match the scheduled date, if set. |
| Adjust latest | When scheduling a task after its latest execution date, the latest can be adjusted to match the scheduled date, if set. |

### 12.4.3 Swoon Mix

| | |
|---|---|
| Sort by latest | If set, tasks will be processed in order of latest time. |
| Sort by duration | If set, tasks will be processed in order of duration (longest first). |
| Sort by number of supporting resources | If set, tasks will be processed in order of the number of resources that are able to perform the task (least resources first). |
| Sort by furthest to nearest | If set, tasks will be processed in order of the distance to the resources (furthest tasks first). This will only be the case when all other sorting options are switched off. |
| Always offer territory | If set, all resources that are able to perform the task are considered, even when a required, preferred or backup resource are given. |
| Check all stops | If set, all visits in the trip are taken into account when selecting the nearest task candidates for inserting into the trip. Otherwise, only the last inserted task is taken into account. |
| Use clusters | If set, candidate resources are determined by computing clusters that link tasks to resources based on distance. Otherwise, the territory is used to select candidate resources. |
| Continue next day | If set, when inserting a task fails on the current day, the algorithm continues to plan that task on a different day. If not, the task will be reevaluated later possibly for another resource. |
| Abort when no option | If set, the algorithm halts when a task could not be planned and/or the user selects cancel in the advice dialog. |
| Check preferred resource | If set, the preferred resource is always evaluated. Other resources may be evaluated as well. |
| Sort Euclidean | If set, the order in which other tasks are evaluated is determined by Euclidean distance instead of route calculation. |
| Use location planning | Obsolete (must be switched off) |

| | |
|---|---|
| Remove resources without options | If set, a resource is only evaluated if it is guestimated that he/she may have enough time to perform the tasks. This option should be switched off. |
| Probability for furthest | Normally, the Swoonmix algorithm starts with the task that is furthest from the nearest resource. With this setting, a random element is introduced so the algorithm will occasionally use the nearest tasks first. If set anything but either 0 or 100%, the results will be non-deterministic. |
| Max nearest | The maximum nr of nearest insertions the algorithm will try. When this number is reached a new furthest seed will be used. |
| Nearest attempts | The nr of nearest candidate tasks that are considered when inserting a task into a trip in nearest mode. The task that is nearest to any of the other stops in the trip is selected. |

### 12.4.4 Trip optimization

| | |
|---|---|
| Extended trip optimization | If set, all planned visits in a trip are unplanned and reinserted into the optimal position in the same trip. Otherwise, 1-opt is used. |
| A.1.1 Travel distance/time only | Performs a TSP algorithm based on distances before running 1-opt optimization. |
| Euclidean distance | If set, the TSP algorithm will be based on Euclidean distances instead of computed routes. |
| Across neighbors limit | Maximum number of nearest tasks in a trip to consider. Note that this setting is not used by the optimization itself.<br>A.1.2 When |
| After inserts | If set to a value N $>$ 0, a trip will automatically be optimized after N inserts into that trip. |

### 12.4.5 Location planning

Tbd

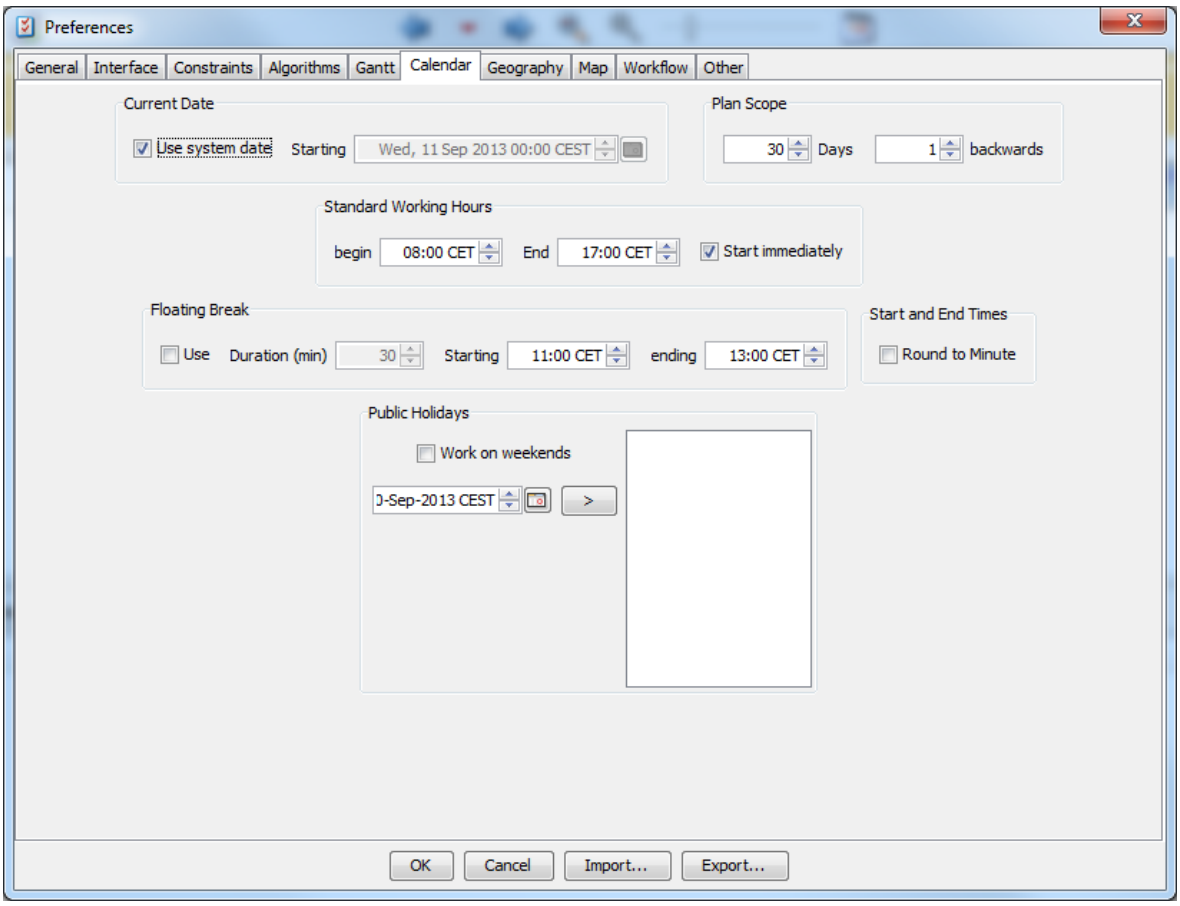
### 12.4.6 Fast lane

| | |
|---|---|
| Min | When using Fastlane, the number of minutes within the time window for the task must start (response time). |
| Use location planning | If set, the Location planning is used to actually plan the tasks. |

### 12.4.7 k-means partitioning

Tbd

## 12.5 Calendar




### 12.5.1 Current date

| | |
|---|---|
| Use system date | If set, the scheduler will use the system date for the time line. |
| Starting | When not using system date, this date/time will be used to simulate the time line. This is useful when working with data from the past. This time is interpreted in the displayed time zone. |

### 12.5.2 Plan scope

| | |
|---|---|
| Days | The number of days starting today that are read into memory. |
| Backwards | The number of days before today that are read into memory. |

### 12.5.3 Standard working hours

| | |
|---|---|
| begin | The default start time of the work schedule for any resource that does not have an assigned schedule. This time is interpreted in the displayed time zone. |

| | |
|---|---|
| end | The default start time of the work schedule for any resource that does not have an assigned schedule. This time is interpreted in the displayed time zone. |
| Start immediately | If set, the resource is assumed to start working at the customers site at the start time. |

### 12.5.4 Floating break

| | |
|---|---|
| Use | If set, floating breaks will be created for every resource when loading the plan. |
| Duration (min) | The duration of the floating break. |
| Starting | The earliest the floating break may start. This time is interpreted in the displayed time zone. |
| Ending | The latest the floating break may end. This time is interpreted in the displayed time zone. |

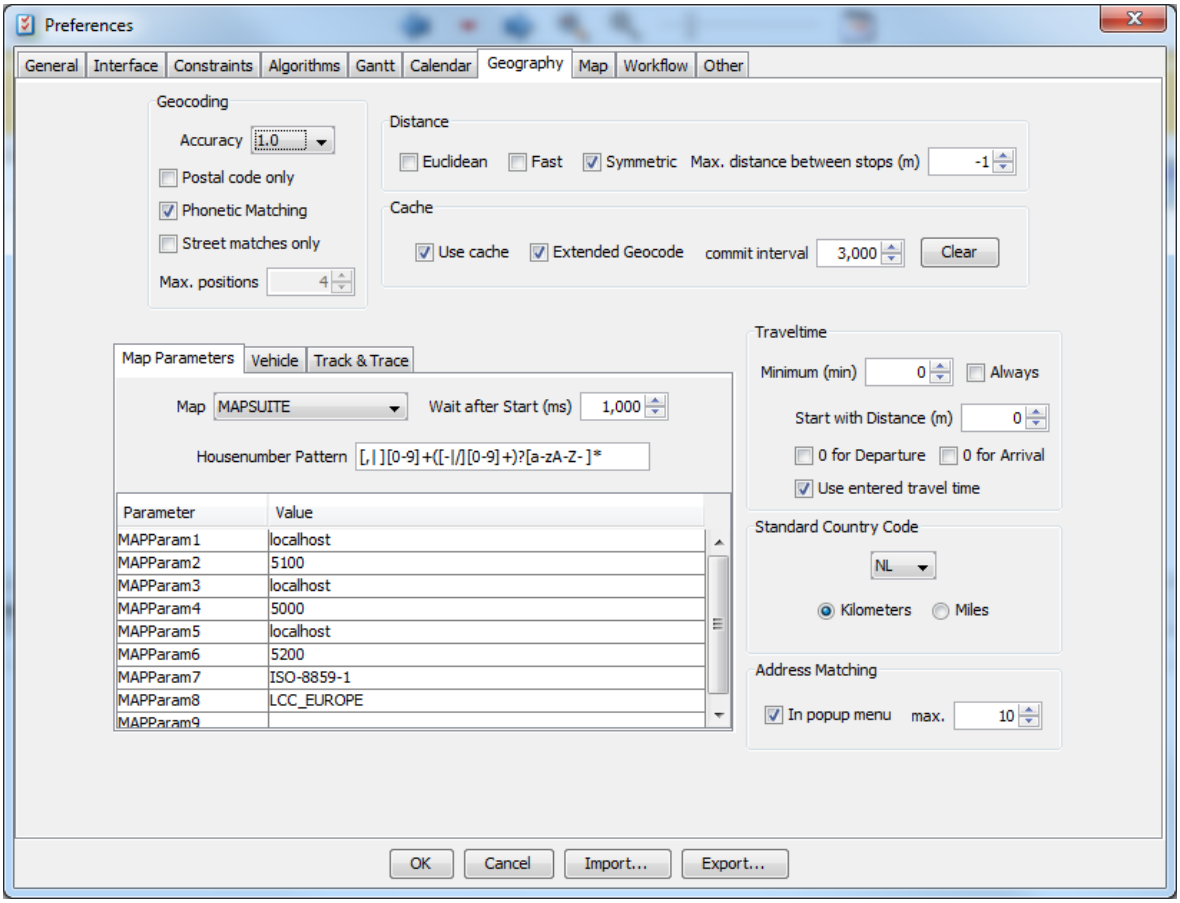
### 12.5.5 Start and end times

| | |
|---|---|
| Round to minute | If set, all visit times will be rounded to the minute. |

### 12.5.6 Public holidays

| | |
|---|---|
| Work on weekends | If set, Saturday and Sunday are considered plannable days. |

## 12.6 Geography




### 12.6.1 Geocoding

| | |
|---|---|
| Accuracy | The resolution for computing geocodes.1 means meter accuracy, .1 means 10m accuracy, etc. |
| Postal code only | If set, geocoding is based solely on postal code. |
| Phonetic matching | If set, soundex is used to determine the best match. |
| Street matches only | If set, geocoding is based solely on street names. |
| Max. positions | The number of digits from the postcode to use, if geocoding on postcode only. |

### 12.6.2 Distance

| | |
|---|---|
| Euclidean | If set, all distance/time calculations are based on Euclidean distances. This will result in very low accuracy. |
| Fast | obsolete (must be switched off) |
| Symmetric | If set, time/distance a->b is considered identical to b->a. |
| Max. distance between stops (m) | If set to anything > -1, the maximum distance a task may be from the previous task when inserting into the |

| | |
|---|---|
| | trip. Note that the distance to/from departure and arrival are not evaluated. |

### 12.6.3 Cache

| | |
|---|---|
| Use cache | If set, all time/distance are cached on the local system. |
| Extended geocode | If set, geocodes are stored including lat/lon and quality indicators. Must be set on. |
| commit interval | Interval in ms after which the time/distance cache will automatically be saved. |
| Clear | Clears all entries from the local cache. This may be necessary when changing geography settings. |

### 12.6.4 Traveltime

| | |
|---|---|
| Minimum (min) | Number of minutes to use a minimum travel time between two locations, except if the locations have the same coordinate. |
| Always | If set, always add the minimum time to any computed travel time. |
| Start with distance | The minimum radius below which the minimum time is not applied. Can be used when several locations are within walking distance. |
| 0 for departure | If set, the travel time/distance to the first task is considered 0. |
| 0 for arrival | If set, the travel time/distance from the last task is considered 0. |
| Use entered travel time | Use actual travel time recorded for visited stops. |

### 12.6.5 Standard country code

| | |
|---|---|
| <list> | The default country code for addresses that do not specify the country code. |

### 12.6.6 Address matching

| | |
|---|---|
| In popup menu | If set, addresses for tasks are geocoded on the fly when using the right mouse button on a task. |
| Max | Max number of matches. |

### 12.6.7 Map Parameters

#### 12.6.7.1 MAPSUITE

| | |
|---|---|
| MapParam1 | The hostname for the geocoding service. |
| MapParam2 | The port for the geocoding service |

| | |
|---|---|
| MapParam3 | The hostname for the routing services. |
| MapParam4 | The port for the routing service |
| MapParam5 | The hostname for the mapping services. |
| MapParam6 | The port for the mapping service |
| MapParam7 | The XML encoding used by the services. |
| MapParam8 | The projection used. |

#### 12.6.7.2 GOOGLE

| | |
|---|---|
| MapParam1 | Delay (ms) between consecutive requests. |
| MapParam2 | Sensor true/false (obsolete as of vs 9.4). |
| MapParam3 | Map type. |
| MapParam4 | Base URL for the mapping services. |
| MapParam5 | URL Signing secret for Static Maps API. |
| MapParam6 | Key for routing and geocoding API. |
| MapParam7 | If true, enables extra debugging info for FX Navigator. |
| MapParam8 | Key for Javascript API. |
| MapParam9 | If true, enables FX Navigator. Otherwise, the standard navigator is used. |

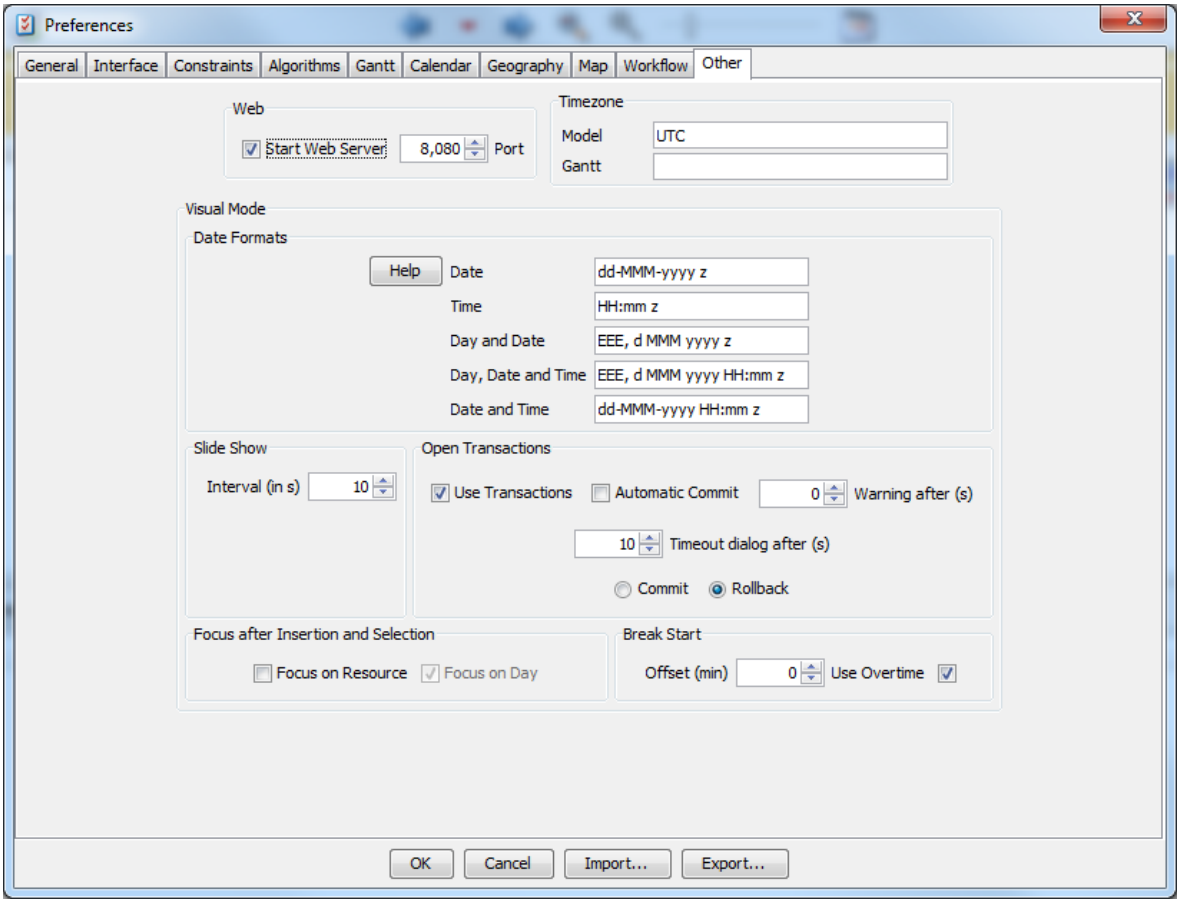
##### 12.6.7.2.1 Using Google Maps

When using the Google Maps mapping provider, please be aware of the following:

There are two API keys required. MapParam6 contains the key for geocoding and routing requests. MapParam8 contains the key for mapping requests.

The Navigator window is implemented using JavaFX and HTML. This is different from the other mapping providers, which use Java Swing. When running in HTTP mode, the navigators main HTML view will be loaded from <server>/omd-resources/javafx/fxmap.html, where <server> is the host part of the 'HTTP Provider URL' preference. Note that OMD can provide the OMD resources package as a WAR file for deployment in any Java servlet container.

## 12.7 Other




### 12.7.1 Web

| | |
|---|---|
| Start web server | If set, the webserver will be started. |
| Port | The TCP port number to start the webserver on. |

### 12.7.2 Time zone

| | |
|---|---|
| Model | The time zone for the data provided to the scheduler. If left blank, the local time zone is used. |
| Gantt | The time zone used throughout the GUI for displaying date/times. If left blank, the local time zone is used. |

### 12.7.3 Date formats

Specifies the various formats for displaying date/time information throughout the GUI. Note that the 'z' or 'Z' parameter specifies time zone information.

### 12.7.4 Open transactions

| | |
|---|---|
| Use transactions | If set, commits to the database will be done using transactions. |

| | |
|---|---|
| Automatic commit | If set, all changes will be committed to the database automatically. Otherwise, all changes are stored locally until the user saves the changes explicitly. |
| Open transaction time out | The number of seconds after which the user is prompted to save any changes, if not using automatic commits. |
| Timeout dialog after | The number of seconds after which the transaction dialog times out, if the user takes no action. The following action is determined by the Commit / Rollback radio buttons. |

### 12.7.5 Focus after insertion and Selection

| | |
|---|---|
| Focus on resource | If set, the GUI switches to the resources if a task was assigned to it. |
| Focus on day | If set, the Calendar automatically shifts to the day on which the assigned has been made. |

### 12.7.6 Break start

| | |
|---|---|
| Offset (min) | The number of minutes from the start of the working hours to add in the create break dialog. |
| Use overtime | If set, the time specified as max overtime is automatically added to the working time in the create break dialog. |

## 12.9 Other settings

| | |
|---|---|
| skipBeyondKeepDays | If set, only tasks that are scheduled before dayOffset + keepDays are serialized to XML. |
| acceptZeroGeocodes | A.1.1 Zero geocodes are empty geocodes for addresses which were not resolved by the map system. This can be useful in certain environments where users plan tasks outside the supported map area (for example, an address in Dubai although the map only supports BeNeLux).<br>A.1.2 A new preference called acceptZeroGeocodes has been added to allow zero geocodes. The preference needs to be set explicitely to support zero geocodes, the default value is false. |