MOMO
TECHNICAL DOCUMENTATION

Version: 2.0

International Relations and Security Network (ISN)
ETH Zentrum LEH D4
8092 Zurich
Switzerland

Table of Contents

1 Third Party Components....1

1.1 Application Stack....1

1.2 Other Components....1

2 Application Design....2

2.1 Architecture....2

2.2 Domain Model....5

3 Ancillary Persisted Objects....10

3.1 Description of Ancillary Objects....10

4 Application Concepts....11

4.1 Teams....11

4.2 User Roles....11

5 Application Functions Accessible from the GUI....13

5.1 Timetracker (> Timetracker)....13

5.2 Reports (> Reports)....13

5.3 Requests (> Requests)....13

5.4 Teams (> Teams)....14

5.5 Application (> Application)....14

6 Conventions, Algorithms and Other Functionality....17

6.1 Conventions....17

6.2 Algorithms....17

6.3 Other Functionality....21

7 Other Definitions 23

7.1 Supported languages 23

7.2 Calendar Weeks 23

7.3 Storage of time values 23

8 Application Requirements 24

8.1 Supported Browsers 24

1 Third Party Components

1.1 Application Stack

At the ISN, Momo is run off a LAMP stack:

- Linux Red Hat Enterprise 6.3
- Apache 2.2.15
- MySQL 5.1.61
- PHP 5.3.3

1.2 Other Components

In addition, Momo builds on the following third party components:

- CodeIgniter 2.1.3
- Propel 1.6.6-dev
- Twitter Bootstrap 2.0.3
- jQuery 1.7.2
- jQuery UI 1.8.21
- Date JS 1.0 Alpha-1
- Bootstrap Colorpicker n/a

Momo comes bundled will all components necessary for runtime deployment.

Phing will need to be made available if the model is to be rebuilt – consult the Momo installation guide for instructions on this.

2 Application Design

2.1 Architecture

Momo makes use of the **Model**-**View**-**Controller** pattern with controllers interacting with the model via a logic layer comprised of **Managers** and **Services**.

2.1.1 Controllers

Controllers negotiate the interaction between managers, services and the views (clients). To this end, they expose a series of actions (methods) which clients may execute by means of http requests.

In what follows, controllers conforming to the naming scheme "Manage*XYZ*Controller" typically provide all actions needed to list, create, modify and delete instances of type ":XYZ".

All controllers extend the base controller :Momo_Controller, which mainly provides convenience methods related to authorization and view rendering.

Momo defines the following controllers:

- **:CronController**
  The :CronController exposes actions needed to execute :CronService tasks. As a security measure, the controller will not run if called outside the command line environment (CLI).

- **:EnforcementController**
  The :EnforcementController exposes a subset of :EnforcementService functionality to clients. At present, this is limited to an action allowing clients to unlock Timetracker weeks.

- **:ManageAdjustmentsController**
  The :ManageAdjustmentsController provides actions that allow clients to manage :AdjustmentEntry instances.

- **:ManageAuditTrailController**
  The :ManageAuditTrailController provides actions that allow clients to manage :AuditEvent instances. As :AuditEvent instances are created by the system and are intended to persist indefinitely, the controller only exposes "read" actions to clients.

- **:ManageBookingsController**
  The :ManageBookingsController provides actions that allow clients to manage :OOBooking instances.

- **:ManageBookingTypesController**
  The :ManageBookingTypesController provides actions that allow clients to manage :OOBookingType instances.

- **:ManageEntryTypesController**
  The :ManageEntryTypesController provides actions that allow clients to manage :RegularEntryType instances.

- **:ManageProjectsController**
  The :ManageProjectsController provides actions that allow clients to manage :Project instances.

- **:ManageRequestsController**
  The :ManageProjectsController provides actions that allow clients to manage :OORequest instances.

- **:ManageSettingsController**
  The :ManageSettingsController provides actions that allow clients to manage :Setting instances.

- **:ManageTeamsController**
  The :ManageTeamsController provides that allow clients to manage :Team instances.

- **:ManageUsersController**
  The :ManageUsersController provides actions that allow clients to manage :User instances.

- **:ManageWorkplansController**
  The :ManageWorkplansController provides actions manage clients to manage :Workplan instances.

- **:ReportsController**
  The :ReportsController provides actions that allow clients to generate reports with regard to project time, Timetracker activity and out-of-office periods.

- **:SecurityController**
  The :SecurityController provides actions that allow clients to authenticate.

- **:TimetrackerController**
  The :TimetrackerController exposes the actions needed for clients to interact with the Timetracker.

- **:UtilityController**
  The :UtilityController serves as a catch-all for actions that are not clearly assignable to one of the above controllers. As such, it contains actions intended for client use as well as such intended for internal use only. The latter will only execute when called from within the CLI environment.

2.1.2 Managers

Managers concern themselves with operations on (or “management of”) specific entity objects. At a minimum, a manager will provide CRUD methods for working with its set of entity objects. Additionally, a manager may provide ancillary methods useful for working within its domain of responsibility.

All manager methods operate within their own transaction scope as it pertains to Propel operations. Callers may provide your own transaction scope, however.

Momo defines the following managers:

- **:AuditTrailManager**
  Allows for the management of :AuditEvent instances.

- **:BookingTypeManager**
  Allows for the management of :OOBookingType instances.

- **:EntryTypeManager**
  Allows for the management of :RegularEntryType instances.

- **:EntryManager**
  Allows for the management of :Entry instances.

- **:OOManager**
  Allows for the management of :OOBooking and :OORequest instances.

- **:ProjectManager**
  Allows for the management of :Project instances.

- **:SettingsManager**
  Allows for the management of :Setting instances.

- **:TagManager**
  Allows for the management of :Tag instances.

- **:TeamManager**
  Allows for the management of :Team instances.

- **:UserManager**
  Allows for the management of :User instances.

- **:WorkplanManager**
  Allows for the management of :Workplan instances.

2.1.3 Services

A **service** provides a solution package for a given problem domain. Services typically make use of manager and/or framework functionality to perform their tasks.

Momo consists of the following services:

- **:ComputationService**
  Allows for the computation of time and vacation related quantities.

- **:CronService**
  Allows for the time scheduled execution of application tasks.

- **:EmailService**
  Allows for the sending of emails.

- **:EnforcementService**
  Provides functionality underpinning the enforcement of constraints placed on the application's users. An example of this would be the detection of weeks left incomplete in the Timetracker.

- **:ReportingService**
  Allows for the compilation of aggregate data on user activity.

- **:SecurityService**
  Provides means to authenticate and authorize users.

2.2 Domain Model

The Momo domain model is implemented through Propel.

For an overview of the domain model, please consult Appendix A (Entity Relationship Model).

As Propel models all entity relationships as bi-directionally navigable, navigability indicators have been omitted from the diagram.

2.2.1 Description of Entity Objects

The following describes Momo's entity objects. In the interest of brevity, only the most pertinent object properties are discussed. A full description of the entity objects can be obtained from the Propel schema definition (XML).

2.2.1.1 :Workplan

Momo's time tracking activities center on the concept of a work-plan.

A work-plan can be thought of as an electronic version of a day planner. Just like a day planner, a work-plan concerns itself with a particular year and provides means to associate data with the days contained therein.

A work-plan has knowledge of the following parameters governing its function:

- The year it applies to.
- The holidays that occur over the course of the year.
- The weekly work hours that apply for the year.
- The annual vacation days that are awarded for the year (full-time basis).

Work-plans are modeled by the :Workplan class, each instance of which is related to the following entity objects:

- :Day (1 : 365..366)
  Each day governed by a :Workplan instance is represented by a :Day instance.

- :Holiday (1 : N)
  Each holiday in the scope of a :Workplan instance is represented by a :Holiday instance.

2.2.1.2 :Holiday

Each holiday that occurs over the course of a work-plan is represented by a :Holiday instance.

A :Holiday instance has knowledge of the following parameters governing its function:

- The type of holiday it represents.
- The date that the holiday applies to.

There are four types of holidays:

- **Full-day holiday**
  A holiday of this type applies to an entire day.

- **AM Half-day holiday**
  A holiday of this type applies to the AM half of a day.

- **PM Half-day holiday**
  A holiday of this type applies to the PM half of a day.

- **One hour holiday**[1]
  A holiday of this type reduces a day's work time by one hour – it does not refer to a particular part of a day.

Holidays occurring in a given time span lead to a reduction in plan time. The reduction applied is based on the weekly work time configured in the applicable work-plan.

2.2.1.3 :Day

As mentioned, each day occurring over the course of a work-plan (year) is represented by a :Day instance.

A :Day instance carries the following information:

- The date that it represents.
- The weekday name that it applies to the date it represents.
- The week number that the date it represents falls on on (as per ISO 8601).

2.2.1.4 :User

Each Momo user account is represented by an instance of :User, which is related to a series of other entity objects, namely to:

:Team instances (N:N)

There are several types of team membership available to users:

- A user can be a *primary member* of a team
- **or** a user can be a a *secondary member* of a team
- **and/or** a user can be a designated *team leader* of a team.

With respect to a single team, primary and secondary memberships are mutually exclusive.

Additionally, a user can hold at most one primary membership – the number of secondary memberships is not restricted.

The designation as *team leader* is independent of the notion of membership proper: A user can be designated *team leader* of a team and at the same time hold *primary membership, secondary membership* or *no membership* for that team.

:User entities maintain references to whatever :Team entities are assigned to them, with the type of relation identified by means of the flags "primary", "secondary" and "leader" (these flags are placed on the junction table entry reflecting the relationship).

:Project instances (N:N)

Projects may be directly assigned to users. Whenever this is the case, :User instances maintain references to the :Project instances assigned to them.

[1] "One-hour holidays" are ETH specific. Organizations that do not reduce work-time prior to full-day holidays may ignore them.

:Entry instances (1:N)

When users enter data into the system, these are modeled by means of :Entry instances.

:Tag (1:N)

The system may attach user-specific data to a given :Day instance. Each datum is modeled by means of an appropriate :Tag instance.

:AuditEvent (1:N)

User actions written to the audit trail are modeled by means of :AuditEvent instances.

:OOBooking (1:N)

Prolonged absences from the office are termed “out-of-office periods”. A single such period is modeled by means of an :OOBooking instance.

2.2.1.5 :Team

A :Team instance models a Momo workgroup and maintains references to the following entity objects:

:User instances (N:N)

For a description of this relation, see section 2.2.1.1

:Project instances (N:N)

Teams may be associated with an arbitrary number of projects. :Team instances maintain references to the :Project instances assigned to them.

:Team instance (N:1)

Teams support simple hierarchization. If applicable, a :Team instance will maintain a reference to its parent :Team instance.

2.2.1.6 :Project

Momo models projects by means of :Project instances. As projects may be assigned to both users and teams, a given :Project instance is related to both the :User and :Team instances it is assigned to (N:N, in both cases).

2.2.1.7 :Entry

As users enter time related data into the system, they give rise to :Entry instances. Each such instance maintains a reference to the :Day instance (N:1) and the :User instance (N:1) it applies to.

:Entry comes in a number of flavors (subclasses):

:RegularEntry

Entering time periods into the Timetracker results in :RegularEntry instances. Each :RegularEntry instance refers to an :RegularEntryType instance (N:1), with the latter providing information as to the entry’s meaning.

:ProjectEntry

Users may annotate a day's time tracking activities by indicating how much of the recorded time they spent on projects assigned to them.

These annotations are modeled by :ProjectEntry, where each :ProjectEntry instance refers to the :Project instance that it applies to (N:1).

As a user annotates her day's project information, the corresponding :ProjectEntry instances will refer to her :User instance (:ProjectEntry descends from :Entry). :ProjectEntry instances for a project that is assigned to the user's primary team will also refer to the :Team instance of the user's primary team.

:OOEntry

When creating an out-of-office booking for a user, the days that the booking extends over are related to :OOEntry instances. In the context of a booking, :OOEntry instances describe the extent of the out-of-office absence along with information as to the configuration of the out-of-office booking vis-à-vis holidays, weekends and possible off-days.

Note that the :OOEntry instances comprising a booking are related to an appropriate :OOBooking instance (N:1).

:AdjustmentEntry

"Adjustments" provide a transparent scheme by which users' time and vacation balances can be modified.

Adjustments to users' work-time or vacation balances are modeled by :AdjustmentEntry, each instance of which represents a date-sensitive offset a user's time or vacation balance.

2.2.1.8 :RegularEntryType

Instances of this class are used to categorize :RegularEntry instances (1:N).

2.2.1.9 :OOBooking

Momo tracks users' prolonged absences from the office (out-of-office periods) by means of :OOBooking instances.

An :OOBooking instance is related to a series of other entity objects:

:OOEntry instances (1:N)

See section 2.2.1.7.

:OOBookingType instances (N:1)

Each instance of :OOBooking identifies the type of out-of-office period by means of a relation to an instance of :OOBookingType.

:OORequest instances (1:1)

If the out-of-office booking arose due to a user request, that request is modeled by means of a related instance of :OORequest.

:User instances (1:N)

Each :OOBooking is related to the :User instance it pertains to.

2.2.1.10 :OORequest

An out-of-office booking can arise due user requests. Whenever this is the case, the request information is represented by an instance of :Request, which, as mentioned above, is related to the booking that it applies to. :OORequests carry information about the request status and a user-specified request message.

2.2.1.11 :OOBookingType

As explained above, an out-of-office booking is associated with a type. Each such type is represented by an instance of :OOBookingType. The most important attributes of the type are those that specify its name, its nature (paid or an unpaid) and its allowed booking increments (by day and/or by half-day).

2.2.1.12 :AuditEvent

Momo maintains an audit trail for a number of user actions. Each entry in the audit trail is represented by an instance of :AuditEvent which is related to the :User instance it applies to (N:1).

2.2.1.13 :Tag

Tags allow user-specific information to be attached to a given :Day instance. A :Tag instance is related to the day it applies to and carries information regarding its type and expiration date. Tags created without an expiration date (null) will persist indefinitely.

3 Ancillary Persisted Objects

These objects are also modeled through Propel, but are not part of the domain model proper.

3.1 Description of Ancillary Objects

:Setting

Application parameters that are administrable by means of the function **Application > Change Settings** are modeled as :Setting instances. At present, only a subset of the parameters modeled this way are in fact exposed in the GUI.

:ApplicationScopeValue

If a Momo variable/quantity is to have application scope it is represented as an instance of :ApplicationScopeValue.

4 Application Concepts

This chapter expands on the concepts and entities underpinning Momo.

4.1 Teams

Teams form Momo's organizational backbone and much of its functionality is team-oriented in one way or another.

Teams may be structured as follows:

- **Multiple Team Membership**
  Users may be members of multiple teams. This, however, contingent on there being at most one team designated as the team of "primary membership".

- **Multiple Team Leaders**
  A team may have multiple users designated as team leaders and a user may function as team leader for multiple teams.

- **Hierarchization of Teams**
  Teams may be simply hierarchized. Users report to the team leader of the highest order team they are a member of.

With regard to multiple team memberships, it is to be noted that outside of those directly assigned, a user has access only to the projects assigned to the team for which she is a primary member.

4.2 User Roles

Momo supports four user roles.

In order of increasing permission scope, these are:

- User
- Team Leader
- Manager
- Administrator

The roles are cumulative in nature, e.g.: the role "Administrator" encompasses all the permissions of the roles beneath it.

4.2.1 Role "User"

This is the basic role assigned to the majority of the application's users.

The role may:

- access the Timetracker.
- generate reports (restricted to information applicable to the user).
- place out-of-office requests.

4.2.2 Role “Team Leader”

For a user to be assignable as team leader she must, at a minimum, be assigned this role.

The role may:

- perform all operations of the role “User”.
- access team management functions.

4.2.3 Role “Management”, Role „Administrator“

The role “Management” is intended for management level users and is at present identical in permission scope to the role “Administrator”.

The roles may:

- perform all operations of the role “Team Leader”.
- manage users.
- manage teams.
- manage work-plans.
- manage projects.
- manage time log entry types.
- manage out-of-office booking types.
- inspect the audit trail.
- change application settings.

5 Application Functions Accessible from the GUI

The notational convention **> Reports > Project Time** indicates that the function **Project Time** is accessed as an entry of the top menu item **Reports**. Similarly, **> Timetracker** indicates, that the function **Timetracker** is accessed directly from the top menu entry **Timetracker**.

5.1 Timetracker (> Timetracker)

The Timetracker is the centerpiece of the application and for most users it is the primary means by which they interact with it. The interface is organized by week and users can switch between weeks by clicking on the provided date-picker.

Each weekday is presented in a separate section. Clicking on any given day expands the section for that day while simultaneously collapsing that of whatever day is already open – this with the caveat that days may only be switched if there are no unsaved entries pending in the currently open day.

Apart from these elements, the user is provided with summary information related to vacation, overtime, Timetracker status and team membership.

5.2 Reports (> Reports)

Momo provides the following reporting functions:

> Reports > Project Time

The function reports on accrued project time from either a project, team or user perspective.

> Reports > Timetracker

The function allows a user's Timetracker activities to be output in summary view.

> Reports > Out-of-office

The function provides a graphical overview of users' out-of-office periods.

5.3 Requests (> Requests)

Users indicate upcoming out-of-office periods by placing appropriate requests with their superior. ("Superior" is always taken to be the team leader of the highest hierarchized team that the user is a member of.)

Upon creation, a request has status "open" and the team leader in charge is notified of its existence. Once a request has been accessed (read) by the team leader in charge its status changes to "pending". Finally, upon approval or rejection, the request's status is updated to either "approved" or "denied".

The possible request states have the following implications:

- "open" - request may be not be edited by users
- "pending" - request may not be edited by users
- "denied" - request may be edited by users (for resubmit)
- "approved" - request may not be edited by users

5.4 Teams (> Teams)

Team leaders perform team related management tasks by means of the following functions:

> Teams > Team Overview

By means of the team overview, team leaders are able to quickly assess various metrics and statuses for the team members they are in charge of.

For each team member, the overview lists the accrued overtime, the vacation days remaining, spent or booked, as well as the designated off days (for part-timers). In addition, the overview provides information about, and quick-links to, pending requests and/or Timetracker days with status "incomplete".

The function also provides a series of actions that can be performed for a given team member.

These are:

- Unlock incomplete days
- Unlock specific days
- Send message

> Teams > Manage "Out of Office"

Manage "Out of Office" allows team leaders to manipulate the out-of-office entries of users they are responsible for. The function presents out-of-office entries on a per-user basis and allows the scope of returned entries to be limited to a suitable time range.

5.5 Application (> Application)

The group encompasses functions related to overall application management.

5.5.1 Manage Users (> Application > Manage Users)

Manage Users allows the management of user accounts.

A user's work-time and vacation related properties are:

- workload - what fraction of a full workload the user is employed for
- off days - the days in a workweek where the user is off (part-timers)
- entry date - the date the user's employment begins (contract start)
- exit date - the date the user's employment ends (contract end)
- type - whether the user is of type "staff" or "student"
- birthdate - the user's birthdate

Accounts of type "staff" are awarded vacation credit and are subject to incomplete week detection, while accounts of type "student" are not awarded vacation credit and are not subject to the detection of weeks left incomplete[2].

Several properties of a user record are immutable:

- login
- type

[2] The type "student" is ISN specific and most likely of no use to outside users.

- workload

Allowing the properties “type” and “workload” to be mutable would have come at the expense of a non-trivial increase in application complexity. As this is a fairly rare use-case, it was decided to assign users a new account whenever these properties change.

5.5.2 Manage Teams (> Application > Manage Teams)

Manage Teams allows for the retrieval and management of teams.

Teams may be simply hierarchized (the implications of this were explained earlier). Furthermore, teams must be assigned team leaders (at minimum one) and they may be assigned users with primary or secondary membership.

5.5.3 Manage Projects (> Application > Manage Projects)

Manage Projects allows for the retrieval and management of projects.

Project management entails the assignment of teams and/or users to projects.

5.5.4 Manage Workplans (> Application > Manage Workplans)

Manage Workplans allows for the retrieval and management of workplans.

Absent a work plan for a given year, users will not be able to make Timetracker entries for it.

Accordingly, it must be ensured that a properly configured work plan is in place prior to the start of a new year – ideally this is done well in advance of the critical deadline.

Momo assigns the status “in use” and “active” to a workplan.

- A workplan is considered “in use” if there are :Entry instances that refer to it.
- A workplan is considered “active” if the current date lies in the workplan‘s scope **or** there are :RegularEntry instances that refer to it.

Once in place, work plans may be modified subject to these constraints:

- A workplan may be deleted, as long as it is not “in use”.
- A workplan may be edited, as long as it is not “active”.

The parameterization of a work plan concerns key data points relevant to time tracking:

- weekly work hours - for full time work
- annual vacation credit - in days, relative to full-time work
- full-day holidays - days that are full day holidays
- half-day holidays[3] - days that are half-day holidays
- one-hour holidays - days that carry one-hour holidays

The “annual vacation credit” property is subdivided into three tiers, with each tier corresponding to a certain employee age band[4]:

- **Tier 1** corresponds to the age band up to 19 years.

[3] Half-day holidays are always placed in the PM.
[4] This is a feature of Swiss labor laws / ETH regulations.
Users in jurisdictions where this does not apply may set all tiers to the same value.

- **Tier 2** corresponds to the age band from 20 to 49 years.
- **Tier 3** corresponds to the age band from 50 onwards.

Once a work plan has been parameterized, the system calculates the resulting annual work time. The result is displayed in an overview (subdivided into months and calendar weeks) and needs to be cross-checked to detect possible errors or omissions in the parameterization.

5.5.5 Manage Entry Types (> Application > Manage Entry Types)

Manage Entry Types allows for the retrieval and management of :RegularEntryType instances.

The definition of an entry type encompasses the following properties:

- **Type**
  The designation of the entry type as it appears in the Timetracker.

- **Work-time Credit**
  Determines a Timetracker entry of this type is awarded work time credit.

- **Enabled**
  Whether the entry is accessible in the Timetracker.

Entry types that are referenced by :RegularEntry instances are considered “in use” and subject to limitations as to the extent of changes permitted:

- They may not be deleted from the system
- They may not have their “work-time credit” parameter changed.

5.5.6 Audit Trail (> Application > Audit Trail)

Audit Trail allows review of Momo’s audit trail.

Audited user actions are listed in chronological order, with detailed information given as to the particulars of each action.

5.5.7 Settings (> Application > Settings)

The function allows management to set aspects governing application behavior:

- **Timetracker lockout**
  The grace time before the Timetracker locks days against further edits.

- **Timetracker relock**
  The grace time before explicitly unlocked days revert to the locked state.

- **Out-of-Office Booking Default Color**
  The default color to use when defining new out-of-office booking types.

- **Site Admin Email**
  The site administrator email address; the site administrator being the person responsible for running Momo from a technical perspective.

6 Conventions, Algorithms and Other Functionality

6.1 Conventions

6.1.1 Vacation Credit

Normalization of Vacation Credit

All vacation credit is expressed in terms of 100% days.

Prorating of Vacation Credit

Vacation days are prorated to a user's employment period and workload.

Transitioning Between Vacation Tiers

The transition among the vacation tiers is handled as follows:

- **Transition from 19 to 20 years of age**
  If a person transitions from 19 to 20 years of age in a given year, the entire year is treated as if the person were 19 years of age.
- **Transition from 49 to 50 years of age**
  If a person transitions from 49 to 50 years of age in a given year, the entire year is treated as if the person were 50 years of age.

6.1.2 Holidays

Prorating of Holidays

Full-day and half-day holidays are prorated to a user's workload. An exception to this are one-hour holidays which are always applied in full, irrespective of the user's workload.

6.1.3 Selective Interpretation of "Delete" Operation

Performing a delete operation on certain entities can lead to a loss of valuable information. For such entities, the delete operation, as reflected in the frontend, is in fact an archive operation.

The entities concerned are:

- :User
- :Team
- :Project

6.2 Algorithms

6.2.1 Incomplete-Week Detection

The system automatically detects Timetracker weeks left incomplete. In the event incomplete weeks are detected for a given user, the user and their team leader(s) are notified of this by email.

For users of type "staff", a week is a candidate for being flagged incomplete, if we detect a workday within that:

- has no regular entry
- **and** is not marked as a “full day” off day
- **and** has no “full day” out-of-office entry of type “paid, autocredit”
- **and** has no “full day” out-of-office entry of type “unpaid”
- **and** is not a “full-day” holiday

If the above checks yield days that do not pass muster, and the user works a part-time workload, the suspect days are checked for possible completeness due to a combination of half-day attributes:

A suspect day is deemed complete, if it is marked as a “half-day (am/pm)” off day and is:

- a (pm/am) holiday
- **or** has a (pm/am) "paid, autocredit" oo entry
- **or** has a (pm/am) "unpaid" oo entry

A suspect day is also deemed complete, if it is a half-day holiday (these always fall on the pm half of the day) and has

- an (am) “paid, autocredit” oo entry
- **or** an (am) “unpaid” oo entry

In addition, for a user of type “staff” the sum of the days marked as “off-days” may not exceed the nominal number of off-days as computed on the basis of the user’s workload. (This restriction is in place so as to force users to place a Timetracker entry when compensating overtime.)

Users of type “student” are not subject to incomplete week detection

6.2.2 Overtime and Vacation Lapses

Momo lapses overtime and/or vacation balances carried into a new year at predetermined dates. The dates governing these operations need to be configured at the time the application is installed. Please note that it is not possible to change them once the application is in use.

The lapse algorithms allow time and vacation related entries to be made *ex post factum* with respect to an occurred lapse date. In such a case, the user’s vacation or overtime balance – as the case may be – will be updated to reflect the late entry.

Both for overtime and vacation lapses, affected users are sent notification emails three, two and one month(s) in advance of the respective lapse date.

Lapse dates are configured by means of :Setting instances. Note that these are not exposed in the GUI (> Application > Change Settings).

While the vacation lapse occurs on the exact day specified, the overtime lapse date is interpreted as specifying the week following which the overtime lapses.

6.2.3 “On-the-Fly” Computation of Time and Vacation Balances

As a general principle, Momo computes time and vacation related balances “on-the-fly” by summing over the appropriate Timetracker entries.

6.2.3.1 Time Balances

Interpreting Momo time balances requires understanding of the following definitions:

Total Work-time Credit

A user's total recorded time that is allotted work-time credit for a given period. This figure is given by summing the :RegularEntry instances carrying work-time credit as well as :OOEntry instances that are paid and awarded automatic work-time credit.

Total Time Credit

Equal to the sum of the "total work-time credit" and any further recorded time not awarded work-time credit – the figure represents the total time reported by a user for a given period.

Work-time Adjustments

Sum of all adjustments made to a user's work-time balance in a given period.

Plan time

The nominal work-time for a given period (adjusted for holidays, workload, etc.).

6.2.3.2 Vacation Balances

In the same vein, the interpretation of Momo vacation balances involves the following terms:

Vacation Days Credited

The number of vacation days credited to a user in a given period.

Vacation Days Consumed

The number of vacation days consumed by a user in an given period.

Vacation Days Adjusted

The sum of all adjustments made to the user's vacation day balance in a given period.

Vacation Days Booked

The number of vacation days entered into the system for a given user that lie in the future and have therefore not yet been consumed.

6.2.4 Reporting on “Project Time”

For a user to have access to a project, the corresponding :Project instance needs to be assigned to either the user’s primary team or to the user directly. As project assignment is not subject to any restrictions, it is possible for a user to have access to a project both by virtue of direct assignment and primary team membership.

This circumstance leads to possible ambiguity when reporting project time by project.

The solution is as follows:

When generating project reports by project, Momo will preferentially allot project time records to teams. Where this is not possible (i.e., when a :ProjectEntry has no reference to a :Team), project time is allotted to the user that owns the record.

6.2.5 Automatic Lockout of Timetracker

Momo will automatically lock days against user edits if they lie a certain number of days in the past. The figure that determines this grace time is represented by means of a :Setting instance. The setting is exposed in the GUI (> Application > Change Settings) and may be changed at any time.

Team leaders may unlock days for their team members. The unlocked days will revert to their locked state after a predefined number of days. That figure too, is represented by a :Setting instance and may be changed via the GUI.

6.2.6 Computation of Out-of-Office Allotments

As outlined earlier, an out-of-office booking consists of an :OOBooking instance associated with a series of :OOEntry instances.

Momo can dynamically reconfigure an out-of-office booking if circumstances warrant. This is achieved by laying down two types of :OOEntry instances for every booking: The first type, designated a “marker type”, outlines the extent of the booking. Based on this, Momo will lay down “allotment types” in accordance with the algorithmic constraints of the associated booking type.

For an out-of-office booking of type “Vacation” extending from Monday to Sunday, it is obvious that the booking does not apply to the weekend itself. Accordingly, while the “marker” entries extend over the entire week, the “allotment” entries will only extend from Monday through Friday. (The avoidance of weekends being part of the algorithmic constraints mentioned above.)

Ultimately, for a given out-of-office booking, the :OOEntry instances of “allotment type” represent the days in the booking period to which the absence formally applies. Doing so allows us to think of out-of-office bookings in colloquial terms when communicating them to Momo. In above example, we would say that the user is on vacation “for a week”, even though technically, they are on vacation for only part of the week.

As mentioned, the “marker-allotment” mechanism allows Momo to dynamically reconfigure out-of-office bookings if circumstances so warrant. An example would be a vacation booking for a part-time employee: such a booking can conceivably extend across “off-days” for which no allotment is made for reasons readily apparent. If the user subsequently changes the placement of an “off-day”, the booking needs to be updated to reflect the new situation. Momo achieves this by reapplying the allotment algorithm to the booking, the topology of which is given by the “marker type” entries.

Before we elaborate on the allotment algorithms, we define a booking’s **configuration** to be the combination of two attributes: One, the attribute “paid”, which is provided by the :OOBookingType

and two, the attribute “auto assign worktime credit” which is provided by the :OOBooking instance itself. (Note that “auto assign worktime credit” is always false for unpaid booking types.)

Allotment algorithms are defined per configuration as follows:

Paid=Yes, ‘Auto Assign Worktime Credit’=Yes

1. Weekends, full-day holidays and full-day off days are not assigned an allotment.
2. Remaining days are assigned full-day/half-day allotments as indicated by the markers.

   However:

   a. Full-day markers result in half-day allotments for days that are half-day holidays or half-day off days.
   b. Half-day markers are applied as indicated, provided they do not conflict with half-day off days or half-day holidays.

Incidentally, this is the way the system provided out-of-office type “Vacation” is configured.

Paid=Yes, ‘Auto Assign Worktime Credit’=No

1. Full-day off days are not assigned an allotment.
2. Remaining days are assigned full-day/half-day allotments as indicated by the markers.

   However:

   a. Full-day markers result in half-day allotments for days that are half-day off days.
   b. Half-day markers are applied as indicated, provided they do not conflict with half-day off days.

Paid=No

1. Weekends and full-day off days are not assigned an allotment.
2. Remaining days are assigned full-day/half-day allotments as indicated by the markers.

   However:

   a. Full-day markers result in half-day allotments for days that are half-day off days.
   b. Half-day markers are applied as indicated, provided they do not conflict with half-day off days.

6.3 Other Functionality

6.3.1 Excessive Overtime Detection

Administrators may configure a threshold indicating the maximum amount of overtime that may accrue for a user (> Application > Change Settings). If this threshold is exceeded, the affected user and their team leader(s) are notified of this by email.

6.3.2 Team Management

Team leaders manage users assigned to teams for which they are assigned as “team leader”.

For primary members, the management tasks comprise the following:

- Management of “Out-of-Office” bookings and requests.
- Monitoring of team members by means of the “Team Summary”.
- Unlocking Timetracker days as requested by team members.
- Monitoring of “Incomplete Week” notifications for team members.
- Monitoring of “Excessive Overtime” notifications for team members.
- Monitoring of “Out-of-Office” requests for team members.

For secondary members, a team leader will not receive any of the notifications listed above – other than that, secondary members can be managed to the same extent as primary ones.

6.3.3 Team Hierarchization

Normally, management responsibilities for a user fall on the team leader of the team for which the user carries primary membership; it is she who receives the various system notifications regarding the user’s activities.

However, if a user holds a secondary membership to a team up the hierarchy chain, management responsibilities for that user will fall to the team leader of the team with the higher hierarchization.

The precise mechanism can be summarized as follows:

For any given user holding a primary team membership, management responsibilities fall to the team leader of the highest hierarchized team that is both a parent of the user’s primary team and for which the user holds a secondary membership.

7 Other Definitions

7.1 Supported languages

Momo is implemented in English only.

7.2 Calendar Weeks

Whenever the system communicates calendar weeks, it does so in accordance with ISO-8601.

7.3 Storage of time values

Momo stores all time values relative to the server time zone.

7.4 User Types

Momo defines two types of users:

Type “Staff”

- May be assigned workloads in increments of 10% ranging from 10% to 100%.
- Is awarded vacation credit
- Is subject to incomplete week detection

Type “Student”

- May only be assigned a workload of 36.89%
- Is not awarded vacation credit
- Is not subject to incomplete week detection.

8 Application Requirements

8.1 Supported Browsers

Momo has been confirmed to work with the following browsers:

- Mozilla Firefox - version 16.0 onwards
- Google Chrome - version 17.0 onwards
- Opera - version 12.0 onwards

Internet Explorer is not supported.

Appendix A – Entity Relationship Model