November 2022

# Vertec-Software Release 6.6

Zürich +41 43 444 60 00
Hamburg +49 40 30 37 36 70
Wien +43 1 22 600 90
mail@vertec.com www.vertec.com

# Table of Contents

# 1 Highlights of Version 6.6 and an Introduction

Version 6.6 delivers many new **Highlights**. The most exciting ones are here in a brief overview:



## 1.1 Backwards Compatibility

Features where backwards compatibility plays a role are described in detail in the corresponding article in the **Backwards Compatibility** section:



## 1.2 To be considered BEFORE upgrading Resource Planning

Customers who already work with Vertec Resource Planning must make certain settings or clean up data BEFORE the upgrade.

Please refer to chapter 2.18 for backward compatibility regarding the migration of Resource planning data.

## 1.3 The Renaming of the Reporting Systems

The difference between "Office reports" and "Extended Office reports" always leads to confusion. New customers in particular are more likely to click on the "Office report" first when searching in the KB to find information on Word reports, for example.

Vertec now only promotes the extended Office reports. They are performant, cloud-enabled and thus available in all full-featured apps.

To make this clear in the terminology as well, we will rename the Vertec reporting systems in the following way:

- The **Office Report** (that is the Expression based Word reports, both Office and Vertec generated, and also the COM based Excel reports) is renamed to Legacy Report. We will also customize the "Report Templates" folder so that it no longer appears in the "New" menu (even for existing customers), which is the *ReportWord* class.
- **Extended Office Report** is renamed to Office Report. This also eliminates the inelegant (and also non-English) abbreviation "EOB".

Vertec will not ship new legacy reports, nor will the systems behind them be maintained. Furthermore, Support will be discontinued at some point. Not soon, and certainly not on 6.7, but sometime in the future. Also, with the term Legacy Office Report, we want to discourage customers from creating new such reports.

## 1.4 Changes to Navigation

Due to the introduction of the new Resource Planning function (see chapter 2), there is a change in the navigation area of Vertec:

Next to the standard view there is now a button for Resource planning (if the module Resource Planning is licensed). The button for the BI view (if the Business Intelligence module is licensed) has been moved one to the right and has a new icon:

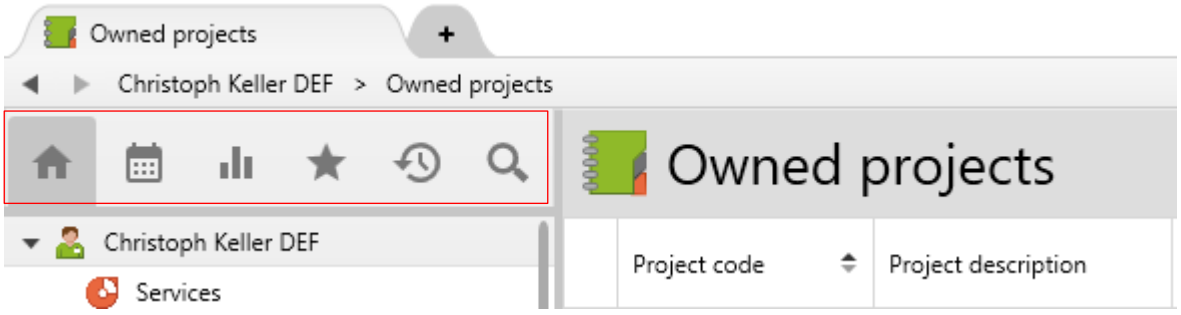



Figure 1 The navigation area with Resource planning and new BI button

This also changes the shortcuts. These new ones are as follows:

- Default view: **F7**
- Resource planning: **F8**
- Business Intelligence: **F9**
- Favorites: **F10**
- History: **F11**
- Search: **F6** / **F12**

## 1.5 Discontinued features of Vertec 6.6

Line: Standard, Expert | Module: PSA | Version: Version 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

### Updates no longer possible across multiple versions

As of Vertec version 6.6, we no longer support updating Vertec across multiple Vertec versions. This means that it is no longer possible to update directly from, for example, Vertec 6.3 to Vertec 6.6, but that it is necessary to proceed step by step. For an update to Vertec 6.6, the customer therefore needs a Vertec version of 6.5.

If an attempt is made to update from an earlier version to 6.6, a corresponding error message is displayed.

The reason for this is that allowing version skipping means that Vertec must always provide all possible scenarios. It is also important from a security point of view that at least the major releases are always installed.

### Excel reports

No more Excel reports are delivered and the existing ones are deleted. See chapter 8.1.

### Link types are removed

The new Resource Planning does completely without the old link types of the existing Resource Planning:

– Project Worker - Resource planning
– Project Worker - Resource planning phases
– Project - Resource planning
– Phase - Resource planning

These link types will be deleted with the upgrade to Vertec 6.6 and thus all related customizing. See chapter 2.18.

### Classes are removed

The classes:

– ResourceContainer
– ResourceRow
– ResourceColumn

will be deleted with the upgrade to Vertec 6.6 and therefore all related customizing. See chapter 2.18.

### System setting will be removed

The previous system setting "Show resource plan for detail line" (internal name: `ShowResourceSubDetail`) will be deleted with the upgrade to Vertec 6.6. See chapter 2.18.

The previous system setting "Service type for vaction" will be deleted with the upgrade to Vertec 6.6. See chapter 13.8.

### ResourceLinks with unique project/phase reference

During the migration, `ResourceLinks` that referenced both a project and a phase will be cleaned up. The (redundant) project reference will be deleted. See chapter 2.18.

## 1.6 Features that will be discontinued in Vertec 6.7

### Authentication via username and password

With version 6.6 **API tokens** are introduced to increase the login security of Web API accesses. The API tokens increase the login security of the Vertec XML interface and the BI API interface. The information about this can be found in chapter 19.1.

The old system (authentication via username and password) will be supported until **Vertec version 6.7** and then discontinued. The changeover should therefore take place soon.

### Excel reports are removed

The Excel reports that are no longer delivered with Vertec 6.6 but are still present in existing installations (see chapter 8.1) are also deleted from existing installations in Vertec 6.7.

# 2 Resource Planning

Line: Standard, Expert | Module: Resource Planning | Version: 6.6.0.1
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Vertec launches with version 6.6 a complete rework of the previous Resource Planning. Similar to the Business Intelligence module, the Vertec tree view can be switched to a Resource planning mode. For this purpose, a Ressource Planning icon is available in the navigation area:

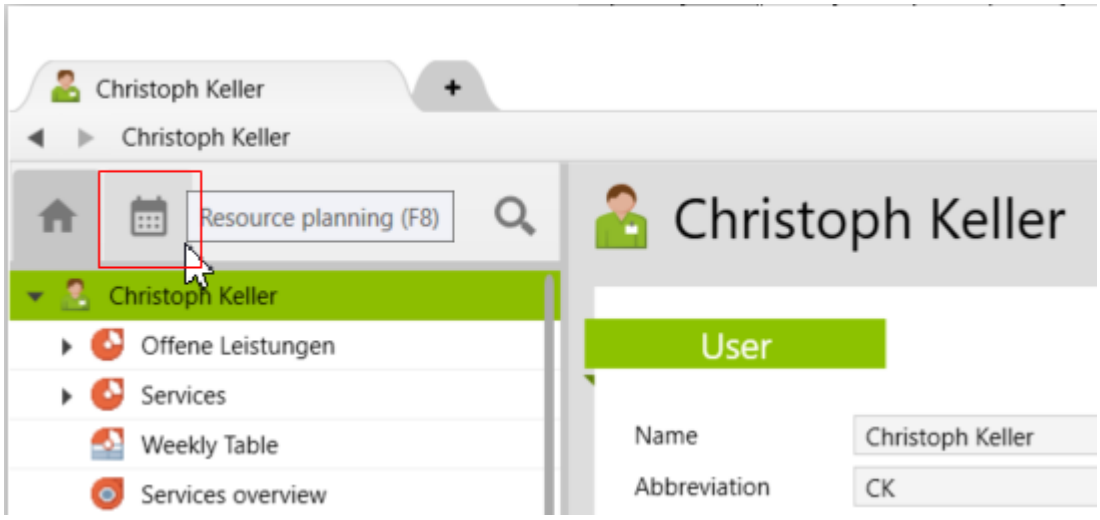



Figure 2 Switch to the Resource planning mode

What exactly is displayed for an object selected in the tree is controlled by Resource planning views. A large number of these are already defined by default. However, the user can define further such Resource planning views. How this is done is described in Chapter 2.5.

If you click on the Resource planning button in the navigation, the following view appears:

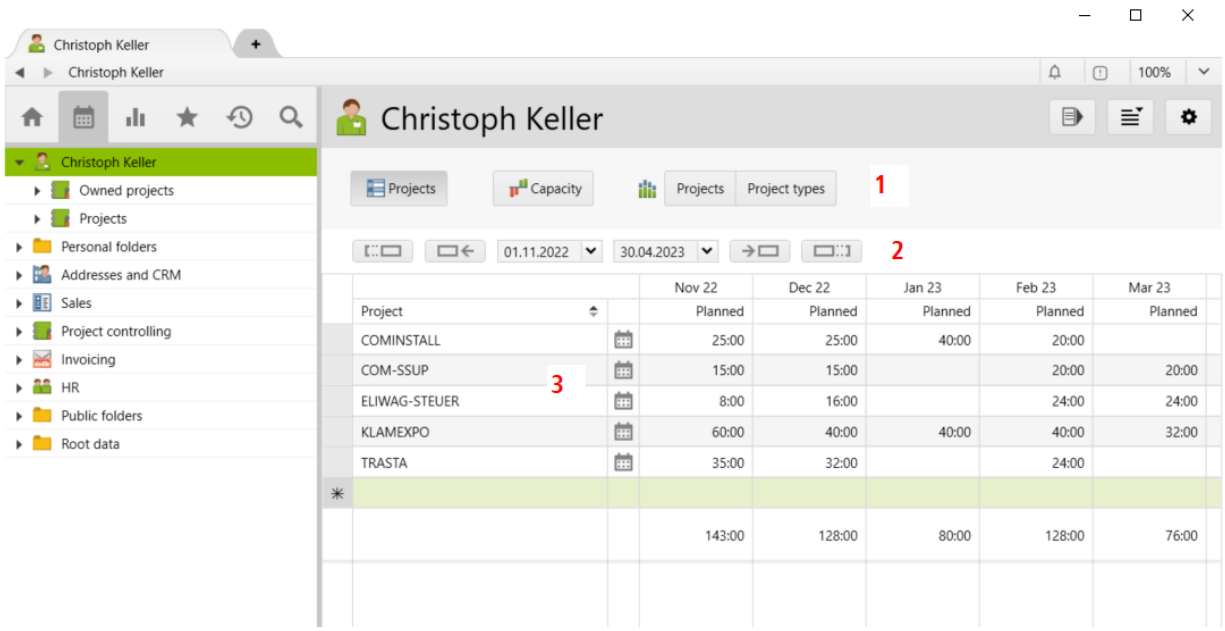



Figure 3 The Resource planning view of an employee

1. **Button** Toolbar for calling up the different views that are defined. Here in the example, these are grouped into three groups. How the views are defined and grouped is explained in chapter 2.5.
2. The **Period Picker**:


This allows the selection of the displayed time period in the following ways:

– Direct selection of start and end interval via date picker
– Shifting the displayed period to the left or right
– Extend the displayed period to the left or right by one interval

The display is preset based on the System Settings > Resource Planning:

– Start of the planning period
– Number of planning intervals

If the period is changed over, this will also be retained when switching to other views until Vertec is closed. Exception: Already opened views in other tabs keep their period.

When Vertec is restarted, the period is displayed again according to the system settings.

Pivot tables are handled separately from time tables and charts. A period change in the pivot table has no influence on time tables and vice versa.

3. The actual **Resource View**. The different types of views are explained in chapter 2.2.

## 2.1 Basic settings for Resource Planning

The settings for Resource Planning can be found in the system settings -> Resource Planning:

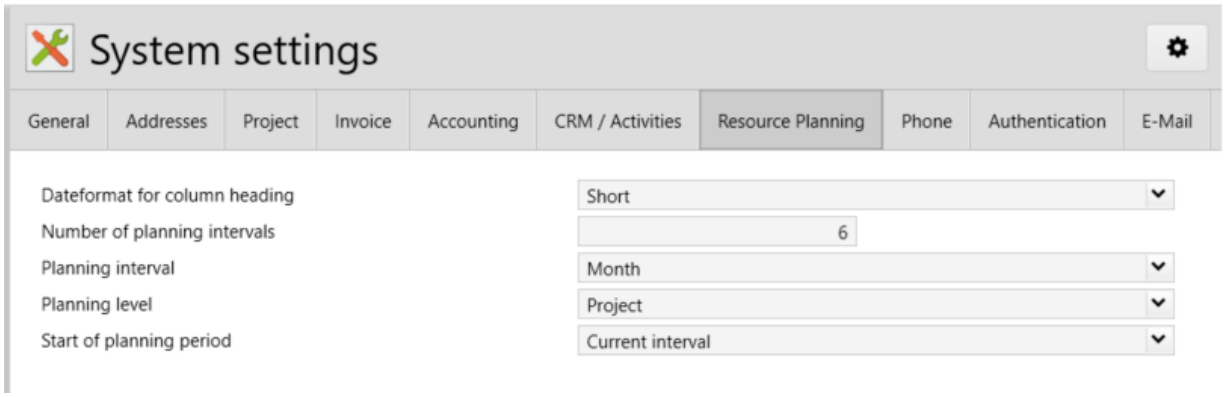



Figure 4 Resource Planning system settings

### Dateformat for column heading

Displays the date values in the column header in **Short** or **Long** as follows.:

| Planning Period | Short | Long |
|---|---|---|
| Day | 07.11.22 | Mo. 07.11.2022 |
| Week | CW 45-2022 | CW 45 07.11.2022 |
| Month | Nov 22 | November 2022 |

PropertyName: `ResourcePlanningHeadingDateFormat`. SelectionProperty. Default setting: Short

### Number of planning intervals

Specifies how many intervals are displayed by default in the views (see chapter 2.2).

The period is calculated from the **Start of the Planning Period** plus the number of planning intervals.

The number of planning intervals should not be too large, for reasons of clarity, performance when building the lists and so that the charts can be displayed as a whole.

PropertyName: `DisplayPeriodIntervalCount`. IntegerProperty. Default setting: 6

### Planning interval

– You can choose between **Day**, **Week** or **Month**. A combination is not possible.

PropertyName: `PlanningInterval`. SelectionProperty. Default setting: Month.

If this setting is not yet set, the message appears when Resource planning is called:

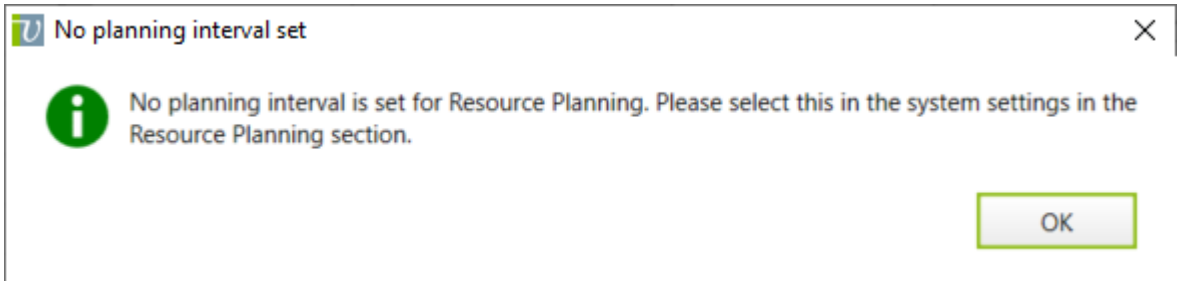
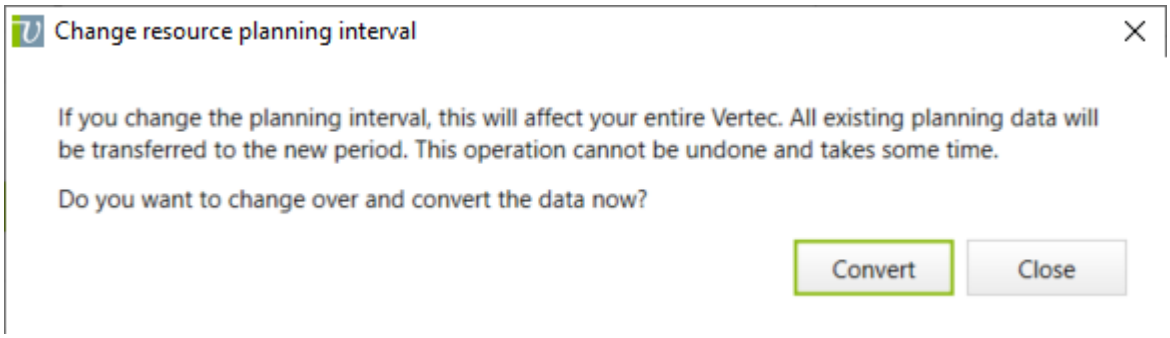



Figure 5 Message displayed when no planning interval has been set yet.

### Changing the Planning interval

If the planning interval is changed while the system is already running, the following message appears:




The existing data is converted to the new planning interval. **This can take a very long time for a lot of data.** We recommend that, if the interval has to be converted, the conversion is carried out at off-peak times or at night and the planning interval is first set to `Not Set`. In this case, no more planning data can be entered. This ensures that no data is lost.

### Planning level

**Project** or **Phase**: You decide whether you want to do Resource planning on project or on phase level. A combination is not possible.

When planning on phases, you can only plan on the first level of phases - planning on subphases is not possible, except when planning on user-phase-links (see 2.15).

PropertyName: `ResourcePlanLevel`. SelectionProperty. Default: Project

### Changing the planning level

If the planning level is changed during operation, the planning data of the other level will no longer be displayed. A transfer from project to phase level and vice versa is not performed.

If planning has been done on both levels so far, Vertec offers a script for harmonizing the planning levels. Please refer to chapter 2.18.

### Start of planning period

Here you can choose between **Current Interval** and **Next Interval**. This is interpreted as follows:

– Planning interval "Day" with number of planning intervals >= 5: first day of the current week
– Planning interval "Day" with number of planning intervals < 5: current day
– Planning interval "Week": first day of the current Week
– Planning interval "Month": first day of the current Month - X

PropertyName: `DisplayPeriodStartInterval`. SelectionProperty. Default: Current interval.

## 2.2 The various types of views

There are the following types of Resource planning views (see section 2.5). Depending on the type of view, the button receives a corresponding icon.

### Time table

In the time table, the planning intervals are displayed against the right in the columns. These tables are displayed very similar to the previous Resource planning lists in Vertec versions before 6.6.

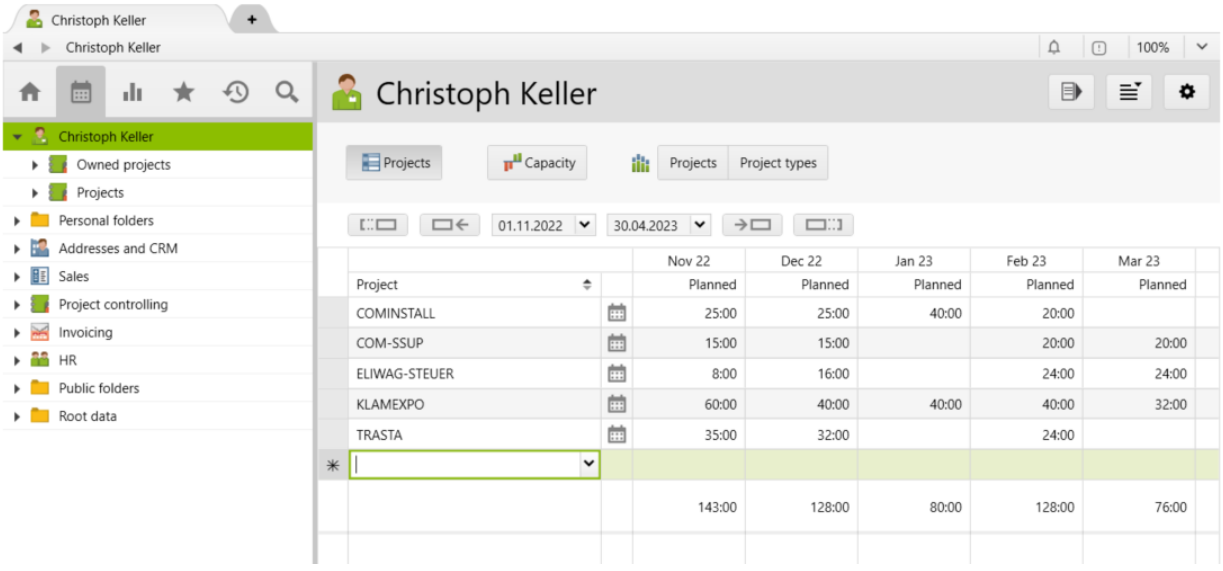



Figure 6 Time table of a User on the projects

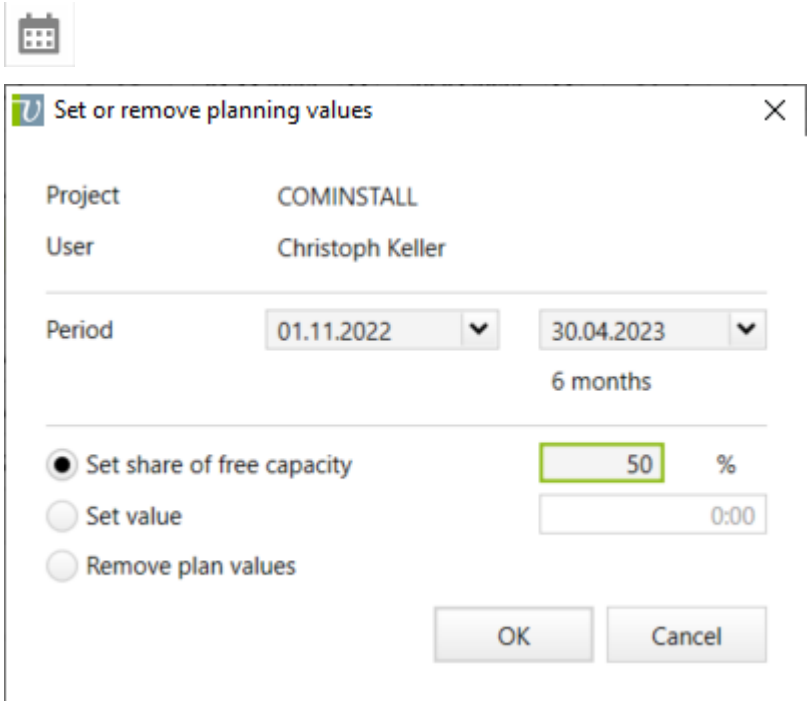
Using the asterisk line, it is possible to add new projects (or phases or workers, depending on the list) for planning.

To ensure that a time control file which starts on a new interval (e.g. current month) is not completely empty, line objects are also displayed which have planning data in the interval before the specified start date.

The time control file has its own list settings, which can also be configured as usual. These setting options have also been extensively expanded, see Chapter 16.

### Set or remove planning values

In each (writable) row of the time control file there is a button with which plan values can be set or removed:




Permits the setting of planned values for a combination of users - project/phase.

User and project (or project phase) are calculated automatically from the corresponding resource list. The time period takes over the values of the current list, but can still be adjusted.

| | |
|---|---|
| **Set share of free capacity** | This allows the available time of the project worker to be distributed linearly over the time period. The available time is the remaining availability, i.e. all times of the user that have not yet been scheduled. At 100%, the entire available time is used, otherwise the corresponding proportion. |
| **Set value** | A value can be specified manually here, and entered. The value must be entered corresponding to the planning interval, i.e. per day, week or month.<br>The specified value is inserted once for each interval that is in the selected period. The fixed value is entered, regardless of how much target time is available. This can lead to overbookings, which may be desired for special assignments such as weekends. Furthermore, over-bookings are an important benefit of Resource planning - the over-bookings indicate bottlenecks that need to be resolved in a management process. |
| **Remove plan values** | Removes all plan data of this project worker - project/phase combination in the specified time period. |

## Pivot table

Starting from a list of entries (user, project or phases) a Resource planning pivot table can be displayed. The pivot table shows the entries as columns and the planned opposite entries as rows.

The pivot table can also be mirrored. In this case it shows the entries as columns and the planned opposite entries as rows.

The advantage of the mirrored pivot table is that new entries can be added via an asterisk line in order to enter planning data on them:

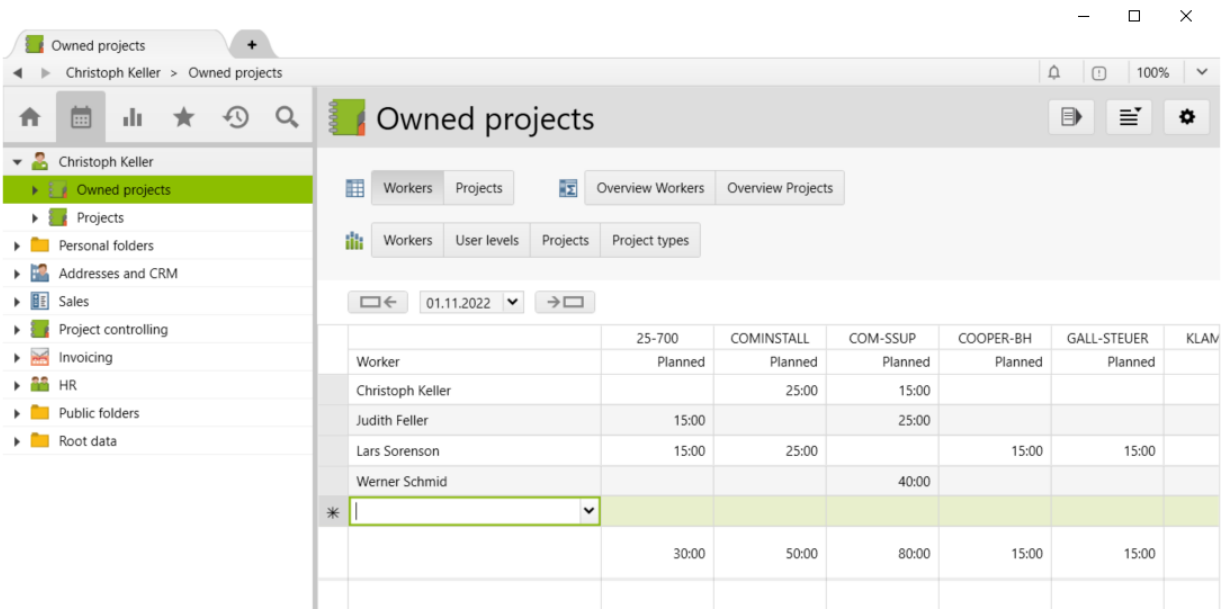



Figure 7 Recording of plan data in a pivot table user - projects.

A single planning interval is always set as the interval for pivot tables.

Like the time control file, the pivot table has its own list settings which can be configured in the usual way. These setting options have also been extensively expanded, see Chapters 3 and 16.

## Capacity chart

The capacity chart is a simplified graphical representation of the planned data with focus on the detection of free capacities or overbookings. The details of the planning (projects, phases) are not shown.

A capacity chart only makes sense based on users or lists of users, since it is based on the capacity data (target times).

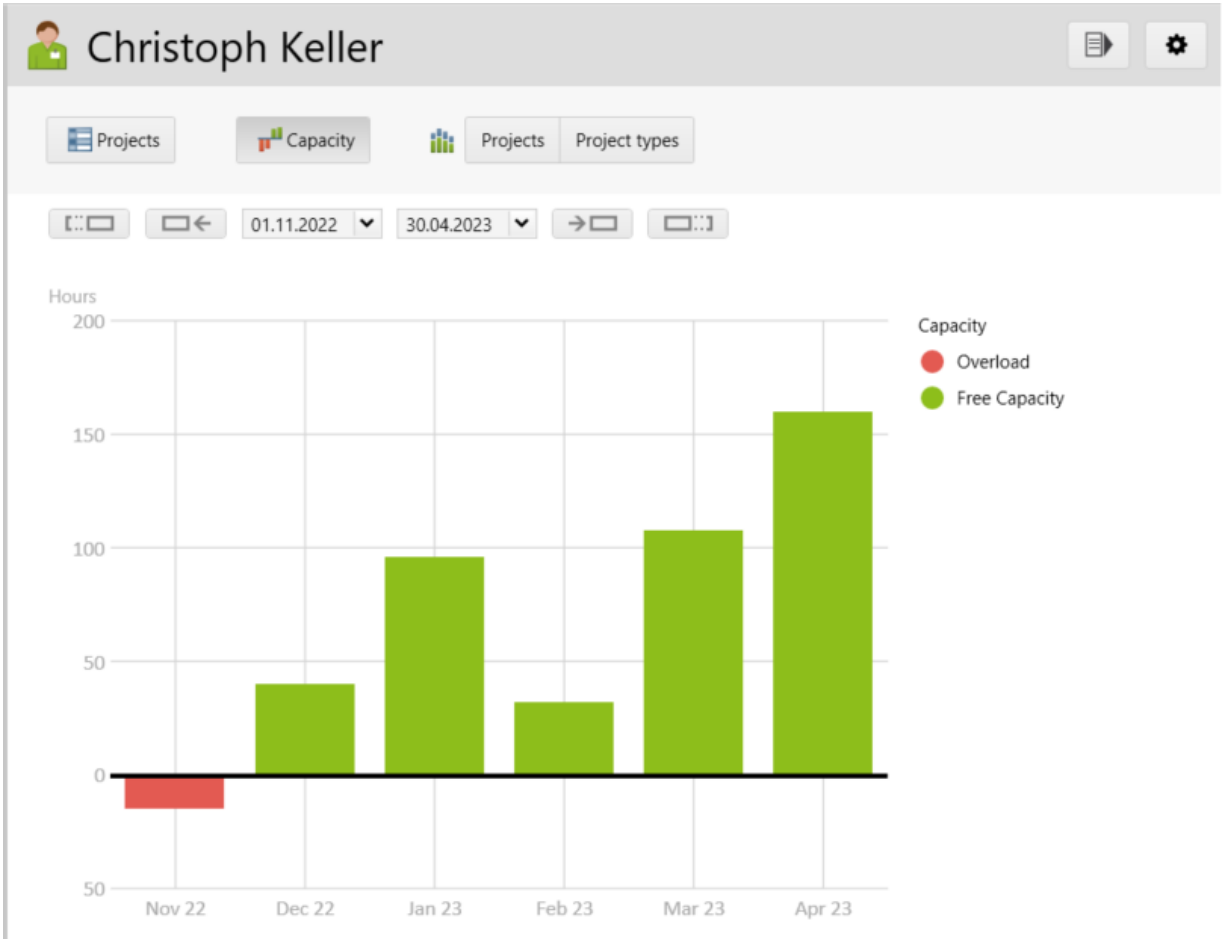



Figure 8 Capacity of a user in the capacity chart

Also here, if you move the mouse over a segment, the detailed number of hours is displayed:

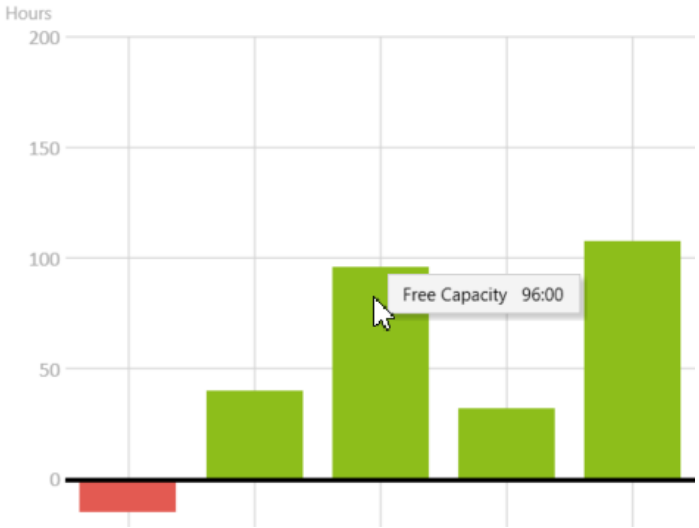



Figure 9 Display of a single segment by mouse-over

The capacities of project workers are calculated as follows:

– Gross capacity: Available working time before considering absences (vacation, sickness, compensation), but including absences of the Free type (e.g. public holidays).
– Net capacity: Available working time after taking all absences into account.

**Resource utilization chart**

A utilization chart presents planning data as a stacked bar graph:

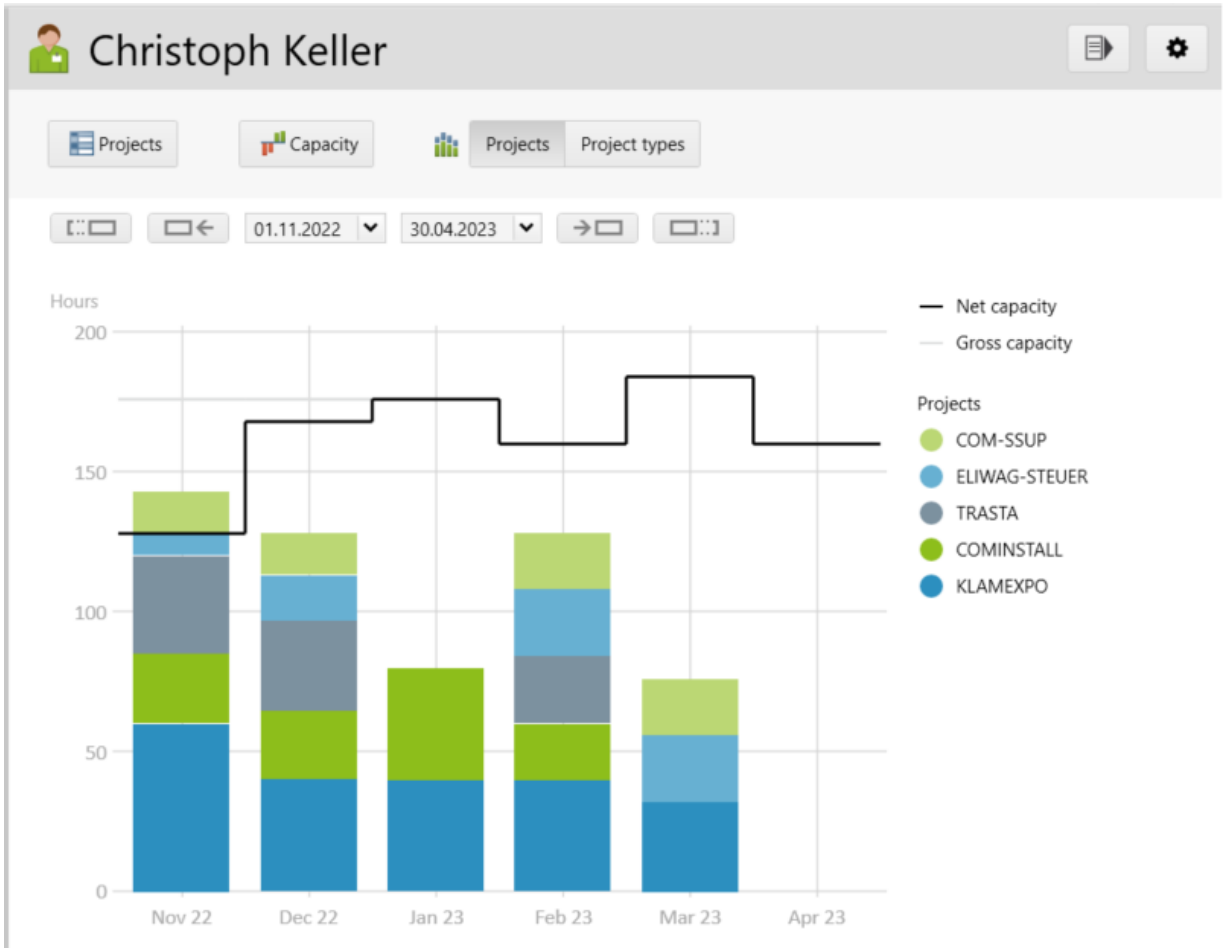



Figure 10 Utilization chart of a user regarding their projects

One bar is displayed per time interval. If you move the mouse over a segment, additional values are displayed in a hint:

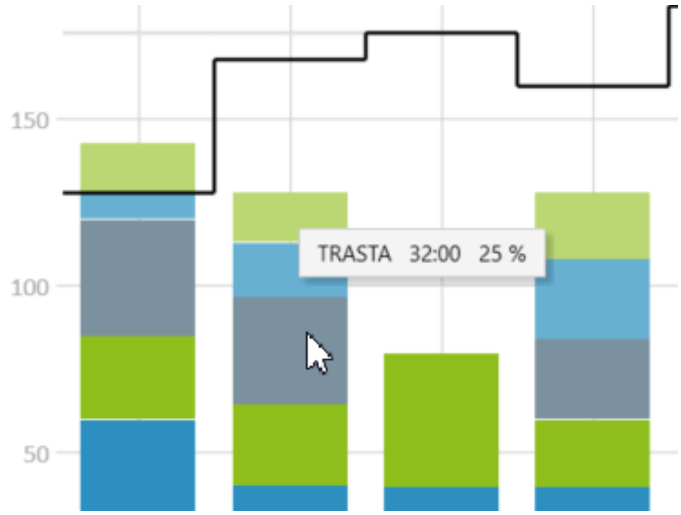



Figure 11 Displaying a single segment by mouse-over

Utilisation charts based on users also represent the capacity of the user or team shown as a line.

The individual segments of a bar usually correspond to the objects on which planning has been performed. However, this view can be further configured by defining Utilization dimensions. See chapter 2.10 for this.

## 2.3 Export data to Excel

On all different views, the numbers behind can be exported to Excel using the `Export to Excel` button:

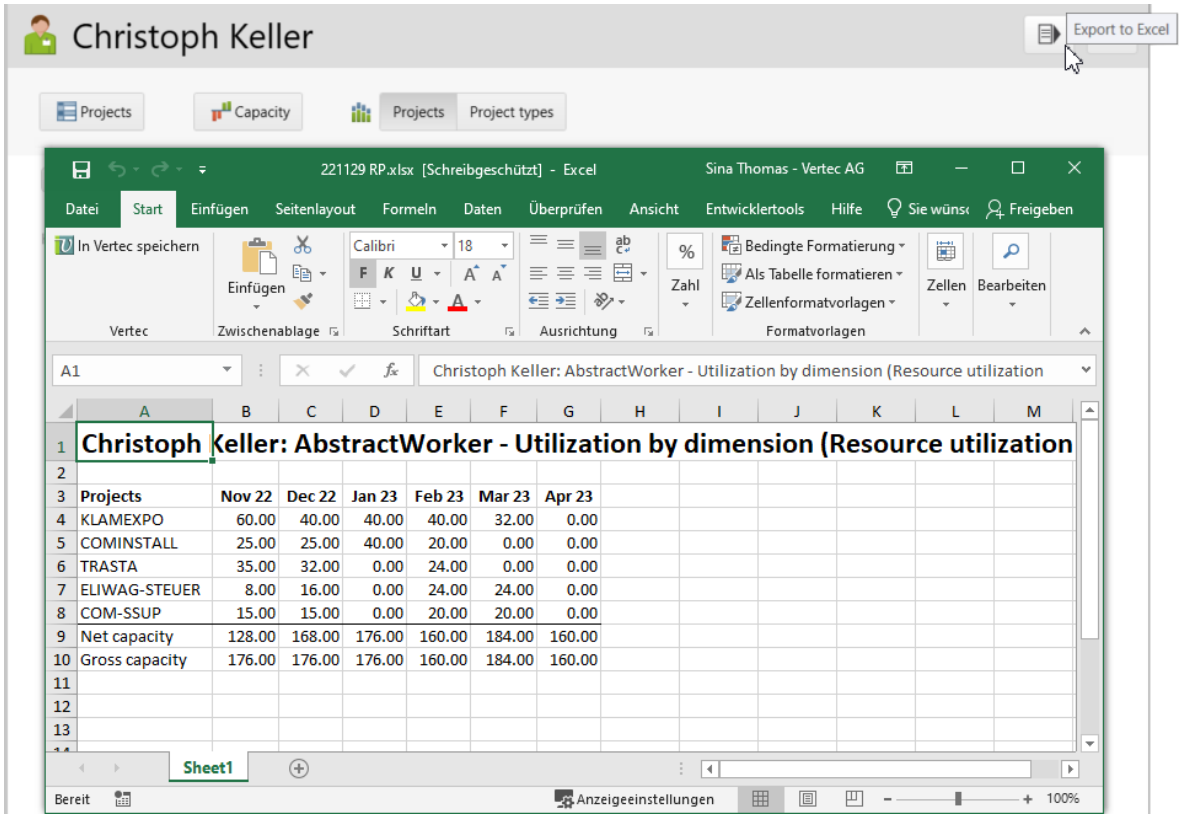



Figure 12 The Resource planning data can be exported to Excel at the push of a button.

## 2.4 The Planning Workers

There is a new planning worker (`PlanningWorker`). This can be used in Resource planning like a user and serves as a placeholder for the planning of a skill or an employee who has not yet been hired or defined.

Planning workers have a capacity (target time) of 0 and have a name, an abbreviation and the possibility to be set to inactive.

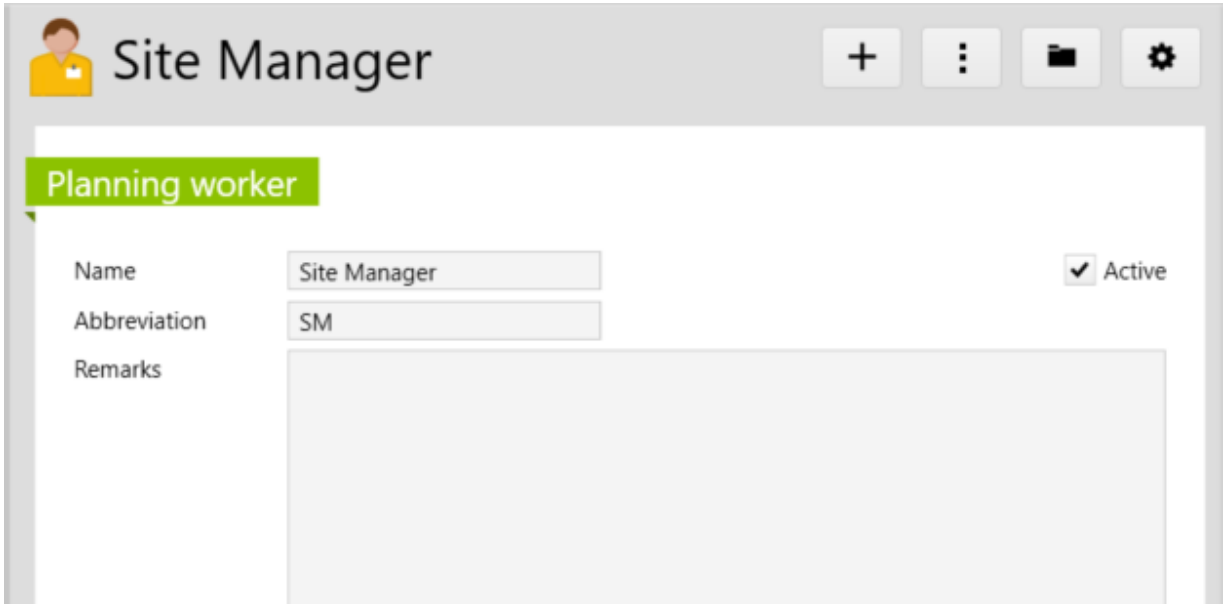
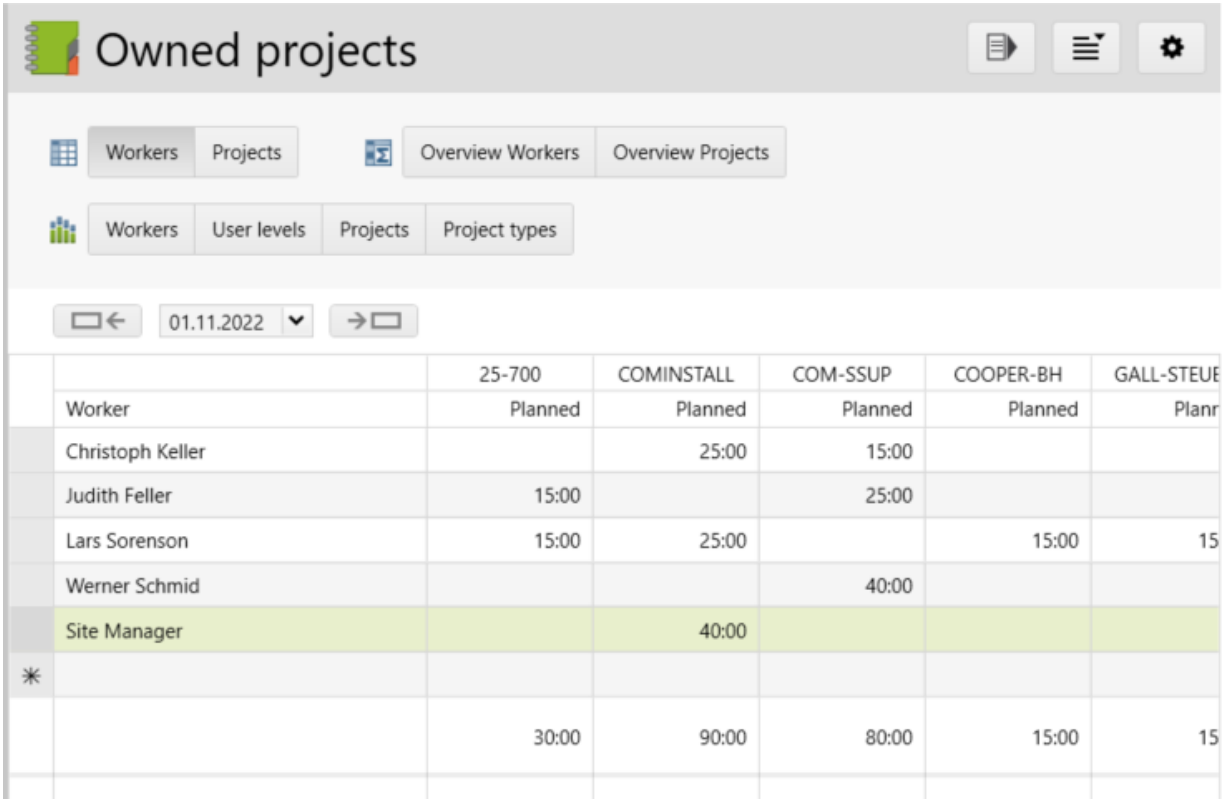



Figure 13 Detailed view of a Planning Worker

For example, a project manager of a civil engineering office does not directly plan individual site managers, but a site manager team planning worker. The team leader then allocates the site manager to the respective projects and transfers the plan values from the team planning worker to the respective user.




| Worker | 25-700 Planned | COMINSTALL Planned | COM-SSUP Planned | COOPER-BH Planned | GALL-STEUE Planr |
|---|---|---|---|---|---|
| Christoph Keller | | 25:00 | 15:00 | | |
| Judith Feller | 15:00 | | 25:00 | | |
| Lars Sorenson | 15:00 | 25:00 | | 15:00 | 15 |
| Werner Schmid | | | 40:00 | | |
| Site Manager | | 40:00 | | | |
| | | | | | |
| | 30:00 | 90:00 | 80:00 | 15:00 | 15 |

Figure 4 Planning with a Planning Worker

Since planning workers have a target time = 0, capacity and utilization can be compared in a team, for example, even if it has assigned planning workers.

**Using Planning Workers**
Planning Workers are entered under Master data > Planning Workers.

The base class of project workers and planning workers is `AbstractWorker`. If both are to be able to be entered in the same folder, you must ensure that you only display list columns that apply to both types. These are the members:

– Name (`name`)
– Active (`aktiv`)
– Abbreviation (`kuerzel`)
– Remarks (`bemerkung`)
– Icon index (`iconindex`)

## 2.5 Resource Planning Views

Resource planning views are configured in the Settings > Resource Planning > Resource planning views folder.

A Resource planning view has the following functions:

– It is created for a specific class and then acts for tree entries of that class
– It can represent a table view (planning table) or a utilization chart
– In case of list views, the Resource planning view saves the list settings

Any number of Resource planning views can be defined per class. For each applicable view, a corresponding button is displayed in the header of the detail view.

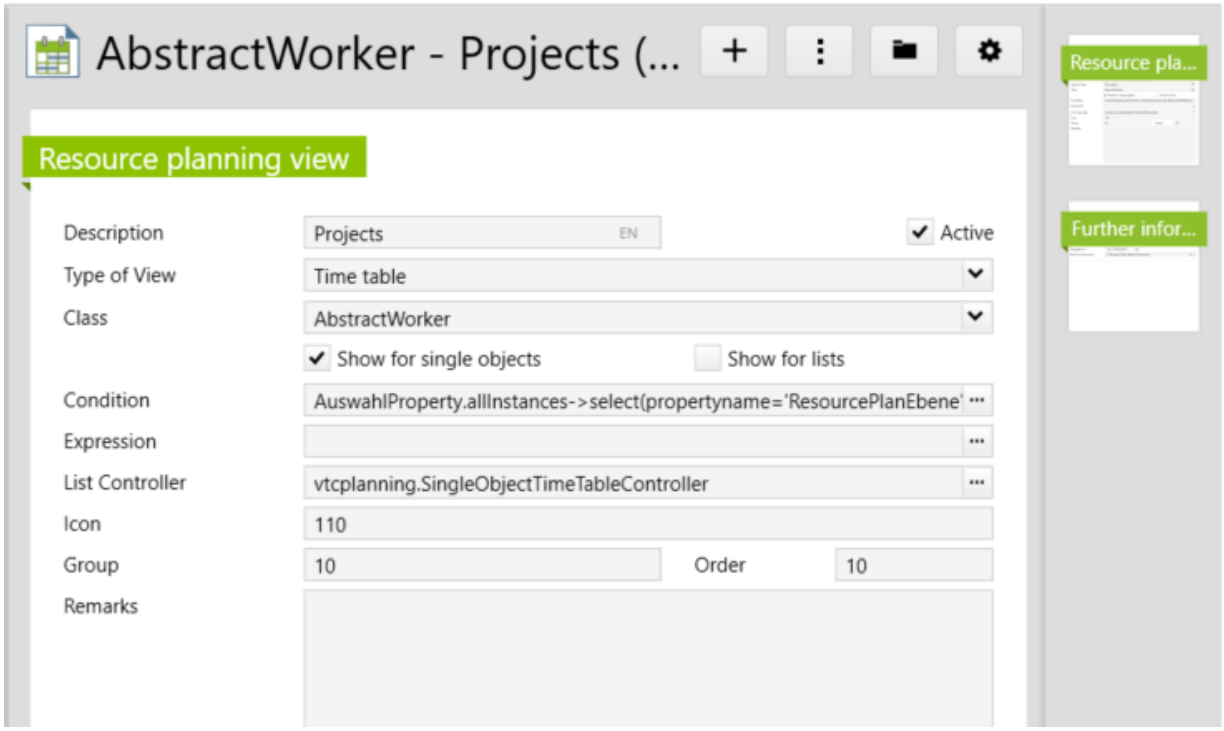



Figure 9 Example of a Resource planning view

A Resource planning view has the following properties:

| Property | Description |
|---|---|
| Description | Name of the Resource planning. This name appears as button text on the detail view and can be stored in different interface languages if required. |
| Active | Controls whether this Resource planning view is applied. |
| Type of View | Four types of views are available:<br>– Time table<br>– Pivot table<br>– Capacity chart<br>– Resource utilization chart<br>These views are described in detail in section 2.2. |
| Class | Controls for which classes this Resource planning view is applied. Available are:<br>– Projektbearbeiter<br>– PlanningWorker (see section 2.4)<br>– AbstractWorker (both types of workers mentioned above).<br>– Projekt / Projektphase (depending on basic setting, see 2.1) |
| Show for single objects / Show for lists | Controls whether the view is available for single objects or for lists. |

| Property | Description |
|---|---|
| Condition | A display condition can be specified here via OCL expression, which can be used to control whether or when this Resource planning view is displayed or not.<br>In the OCL editor, the base `Eintrag` (entry) is displayed. This is the object currently selected in the tree.<br>If the object is used in the OCL, the administrator must write the code so that it works for all combinations of class / single object / list. For this, the concrete type must be checked with `oclIsTypeOf` and/or cast with `oclAsType`, if necessary.<br>The result of this expression must return a Yes/No value (Boolean).<br>This expression is evaluated in the resource view. If the expression is empty or the result is `true`, then the view is displayed.<br>The condition is used, for example, to hide Resource planning views for projects when planning at phase level. |
| Expression | OCL Expression for calculating the entries on which the view is based. Evaluated on the object selected in the tree.<br>This expression only applies to time tables (see 2.2). Pivot tables only show the data that the List Controller calculates, and the charts always take the selected entry or the entries of the container as a basis. |
| List Controller | Here the List Controller (see chapter 16) is specified, which is used for the display of the table (only for table views). Specification is a Python class in the form `<module>.<class>.`<br>Clicking on the three dots next to the field opens the corresponding module code for viewing.<br>An overview of the available prefabricated List Controllers can be found in chapter 18.2, in the module `vtcplanning`. |
| Icon | This icon is displayed on the button (or in case of grouping of several buttons next to the buttons) on the detail view. The default icons are:<br>– Time table: 110<br>– Pivot table: 109<br>– Capacity chart: 111<br>– Utilisation chart: 112 |
| Groups | With a group index several buttons can be combined to a group. On the detail view they will appear together with a featured icon.<br>It doesn't matter which number you choose, buttons with the same number are simply grouped together. |
| Order | Here you can control the order in which the buttons are displayed in the detail view. These are sorted in ascending order and, if groups are defined, in ascending order per group.<br>We recommend choosing larger sorting steps (e.g. steps of 10). This makes it possible to insert buttons in between later. |

## 2.6 Asterisk line and combo boxes for Resource Planning

Certain planning tables support the capture of new rows, which can then be planned on.

In order to display an asterisk line (a so-called ghostrow), the List Controller must implement an `add_row_object method` (see chapter 16) and in the list settings, the tick for `Show add new row` must be set.

For selecting the row object in the list, special ComboBoxes must be used, the "normal" ComboBoxes do not work in this context.

The following ComboBox types are available for this purpose:

| Control Element | ComboBox | Calculation selection list |
|---|---|---|
| cmbResourceProject | ResourceProjectComboBox | Starting from the users from container or single object, for which resource plan data can be entered, the projects, to which the users are assigned (`erfprojekte`), are combined (without duplicates).<br>Only projects that are also writable are displayed (see Chapter - Permissions).<br>Projects that already appear as rows in the planning table do not appear again for selection.<br>Renderer: `rndGhostRowOnlyEditable`.<br>Default column expression is `code`. |
| cmbResourceWorker | ResourceWorkerComboBox | All active users whose resource plan data (`resourcelinks` member) is write-accessible.<br>Users that already appear as rows in the planning table do not appear again for selection.<br>Renderer: `rndGhostRowOnlyEditable`.<br>Default column expression is `name`. |
| cmbResourcePhase | ResourcePhaseComboBox | Starting from users, first the projects are determined as with the **ResourceProjectComboBox** (see above).<br>From these projects all planable phases are displayed for selection. Only active phases of active projects are displayed or inactive phases of active projects that are not `concluded` and not `refused`.<br>Phases that already appear as a row in the planning table do not appear again for selection.<br>Renderer: `rndGhostRowOnlyEditable`.<br>Standard Columns-Expression is `projekt.asstring + ' / ' + asstring` |

**Customizing the ComboBoxes for the asterisk lines**

The selection lists of the resource ComboBoxes can be customized via `ListExpression`.

To do this, the corresponding control must be specified as an XML control via class name and the ListExpression property must be specified.

Within the ListExpression the usual OCL variables like `varContext` or `varParent` are available.

The Resource ComboBoxes may only appear in the ghostrow of Resource planning tables, because unlike other ComboBoxes they select the object for a new row and do not assign a property of the row object.

Therefore, in addition to the combobox control for the column, the `rndGhostRowOnlyEditable` renderer must be used. This prevents the ComboBox from being displayed in a row other than the asterisk line.

---

**Further information** for customizing ComboBoxes in lists can be found in the article Customizing controls in lists under: [www.vertec.com/kb/steuerelementxml/](www.vertec.com/kb/steuerelementxml/).

---

## 2.7 Summation tables (read-only)

Time tables and pivot tables can also be displayed as summation tables. This is useful, for example, on a team container to display an overview of all planning totals of projects.

In this case, the tables are not writable, since the value in the cell is composed of several objects, i.e. it is summed up.

Summation tables can be recognized in the resource view by the fact that they have a sum icon in the icon and are grouped separately:

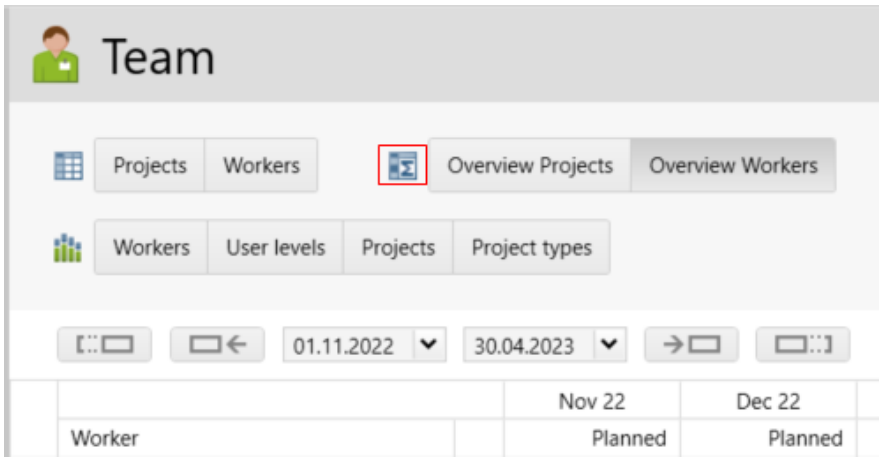



Figure 10 Summation tables have a summation icon in the icon and are grouped together

A number of summation tables are supplied as standard. The complete list can be found in chapter 2.9. For the creation of customer-specific summation tables there are a number of supplied List Controllers. These can be found in chapter 18.2.

## 2.8 URL support for Resource Planning Views

The Vertec breadcrumb URL is also available in the Resource planning views:

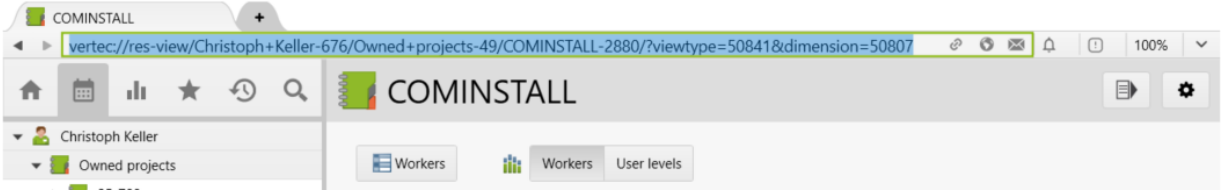



Figure 11 Breadcrumb URL of a Resource planning view

It can be copied and sent using the usual buttons.

The path of a Resource planning URL starts with `res-view`. It represents the object selected in the tree and has the following parameters:

– viewtype=<id>
– dimension=<id>

The Resource planning views can also be saved as favorites. These are displayed with all parameters (see above) when called up. Resource planning views saved as favorites are given a name according to the pattern **Node (RP): Displayname**, for example **Project (RP): User**. In this way, the Resource planning views Favorites can be identified at a glance:

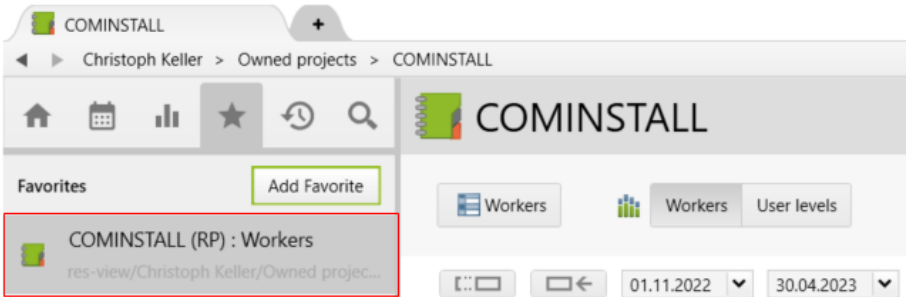


Figure 12 A Resource planning view, saved as a favorite

No parameters are saved in the history. It is therefore only possible to navigate to the specific Resource planning view. The corresponding button for the view must then be pressed again manually.

## 2.9 Supplied Resource Planning Views

The following Resource planning views are supplied by Vertec as standard:

**Planning level Project**

If you plan on projects - i.e. the planning level is set to `Project` in the Resource planning system settings, see 2.1 - the following Resource planning views are available:

– **On Users** (actually: on `AbstractWorker`, i.e. project workers and planning workers, see 2.4):

| Name | View | Description | Active | Single | Lists |
|---|---|---|---|---|---|
| Projects | Time table | Planning of a single user on projects, with projects as rows and the time intervals as columns. | X | X | |
| Projects | Pivot table | Pivot table on a list of users, with users as columns and projects as rows. With asterisk line. | X | | X |
| Workers | Pivot table | Pivot table on a list of users, with users as rows and projects as columns. | X | | X |
| Overview Projects | Time table | Summation table (see 2.7) on a list of users. Shows a time table with projects as rows and time intervals as columns, with the respective totals of the users in the list. Read-only. | X | | X |
| Overview Workers | Time table | Summation table (see 2.7) on a list of users. Shows a time table with users as rows and time intervals as columns, with the respective totals of the users in the list. Read-only. | X | | X |
| Capacity | Capacity chart | Shows a capacity chart (see 2.2) for a single user or for the users of a list. | X | X | X |

| | | | | | |
|---|---|---|---|---|---|
| Utilization by dimension | Resource utilization chart | This Resource planning view represents the view for all defined utilization dimensions (see chapter 2.10) for users. | X | X | X |

– **On Projects:**

| Name | View | Description | Active | Single | Lists |
|---|---|---|---|---|---|
| Workers | Time table | Planning of user (project workers and planning workers) on a single project. With users as rows and the time intervals as columns. | X | X | |
| Workers | Pivot table | Pivot table on a list of projects, with projects as columns and users as rows. With asterisk line. | X | | X |
| Projects | Pivot table | Pivot table on a list of projects, with users as columns and projects as rows. | X | | X |
| Overview Workers | Time table | Summation table (see 2.7) on a list of projects. Shows a time table with users as rows and time intervals as columns, with the respective totals of the projects in the list. Read-only. | X | | X |
| Overview Projects | Time table | Summation table (see 2.7) on a list of projects. Shows a time table with projects as rows and time intervals as columns, with the respective totals of the projects in the list. Read-only. | X | | X |
| Utilization by dimension | Resource utilization chart | This Resource planning view represents the view for all defined utilization dimensions (see chapter 2.10) for projects. | X | X | X |

**Planning level Phases**

If you plan on project phases - i.e. the planning level is set to `Phases` in the Resource planning system settings, see 2.1 - the following Resource planning views are available:

– **On Users** (actually: on `AbstractWorker`, i.e. project workers and planning workers, see 2.4)

| Name | View | Description | Active | Single | Lists |
|---|---|---|---|---|---|
| Phases | Time table | Planning of an individual user in phases | X | X | |
| Phases | Pivot table | Pivot table on a list of users, with users as columns and phases as rows. With asterisk line. | X | | X |
| Workers | Pivot table | Pivot table on a list of users, with phases as columns and users as rows. | X | | X |
| Overview Projects | Time table | Summation table (see 2.7) on a list of users. Shows a time table with projects as rows and time intervals as columns, with the respective totals of the project phases per project. Read-only. | X | X | |
| Overview Phases | Time table | Summation table (see 2.7) on a list of users. Shows a time table with phases as rows and time intervals as columns, with the respective totals of the project phases. Read-only. | X | | X |

| | | | | | |
|---|---|---|---|---|---|
| Overview Workers | Time table | Summation table (see 2.7) on a list of users. Shows a time table with users as rows and time intervals as columns, with the respective totals of all phases. Read-only. | X | | X |
| Capacity | Capacity chart | Shows a capacity chart (see 2.2) for a single user or for the users of a list. | X | X | X |
| Utilization by dimension | Resource utilization chart | This Resource planning view represents the view for all defined utilization dimensions (see chapter 2.10) for users. | X | X | X |

– **On Projects:**

| Name | View | Description | Active | Single | Lists |
|---|---|---|---|---|---|
| Workers | Pivot table | Pivot table on a list of projects. Shows users as rows and as columns all schedulable phases of all projects in the list. With asterisk line. | X | | X |
| Workers | Pivot table | Pivot table on a single project. Shows users as rows and as columns all schedulable phases of the project. With asterisk line. | X | X | |
| Phases | Pivot table | Pivot table on a list of projects. Shows all project phases of the projects in the list as rows and the users as columns. | X | | X |
| Phases | Pivot table | Pivot table on a single project. Shows all project phases of the project as rows and the users as columns. | X | X | |
| Overview Workers | Time table | Summation table (see 2.7) on a single project. Shows a time table with users as rows and the time intervals as columns, with the respective summations. Read-only. | X | X | |
| Overview Phases | Time table | Summation table (see 2.7) on a single project. Shows a time table with the project phases of the project as rows and the time intervals as columns, with the respective summations. Read-only. | X | X | |
| Overview Workers | Time table | Summation table (see 2.7) on a list of projects. Shows a time table with users as rows and the time intervals as columns, with the respective summations. Read-only. | X | | X |
| Overview Projects | Time table | Summation table (see 2.7) on a list of projects. Shows a time table with projects as rows and the time intervals as columns, with the respective summations. Read-only. | X | | X |
| Overview Phases | Time table | Summation table (see 2.7) on a list of projects. Shows a time table with the phases of the projects as rows and the time intervals as columns, with the respective summations. Read-only. | X | | X |
| Overview Workers | Pivot table | Summation table (see 2.7) on a list of projects. Shows a pivot table with users as rows and the projects as | | | X |

| | | | | | |
|---|---|---|---|---|---|
| | | columns, with the respective summations of the phases. Read-only. | | | |
| Overview Projects | Pivot table | Summation table (see 2.7) on a list of projects. Shows a pivot table with the projects as rows and the users as columns, with the respective summations of the phases. Read-only. | | | X |
| Utilization by dimension | Resource utilization chart | This Resource planning view represents the view for all defined utilization dimensions (see chapter 2.10) for projects. | X | X | X |

– **On Phases:**

| Name | View | Description | Active | Single | Lists |
|---|---|---|---|---|---|
| Workers | Time table | Planning of users on a single phase. | X | X | |
| Workers | Pivot table | Pivot table on a list of phases, with users as rows and the phases as columns. With asterisk line. | X | | X |
| Phases | Pivot table | Pivot table on a list of phases, with phases as rows and users as columns. | X | | X |
| Overview Workers | Time table | Summation table (see 2.7) on a list of phases. Shows a time table with users as rows and the time intervals as columns, with the respective summations of the phases in the list. Read-only. | X | | X |
| Overview Phases | Time table | Summation table (see 2.7) on a list of phases. Shows a time table with the phases as rows and the time intervals as columns, with the respective summations of the phases in the list. Read-only. | X | | X |
| Utilization by dimension | Resource utilization chart | This Resource planning view represents the view for all defined utilization dimensions (see chapter 2.10) for project phases. | X | X | X |

The Resource planning views can be disabled or enabled, depending on your needs.

## 2.10 Utilization Dimensions

Resource utilization charts (see chapter 2.2) are displayed divided according to different dimensions. These are the criteria according to which the bars (or values) are divided.

For example, this can be by project manager, by industry, by project type, etc. You can also define your own dimensions as you wish, such as `Sales/Offered`, `Productive/Unproductive`, etc.

Utilization dimensions are defined in the **Settings > Resourceplanning > Utilization dimensions** folder

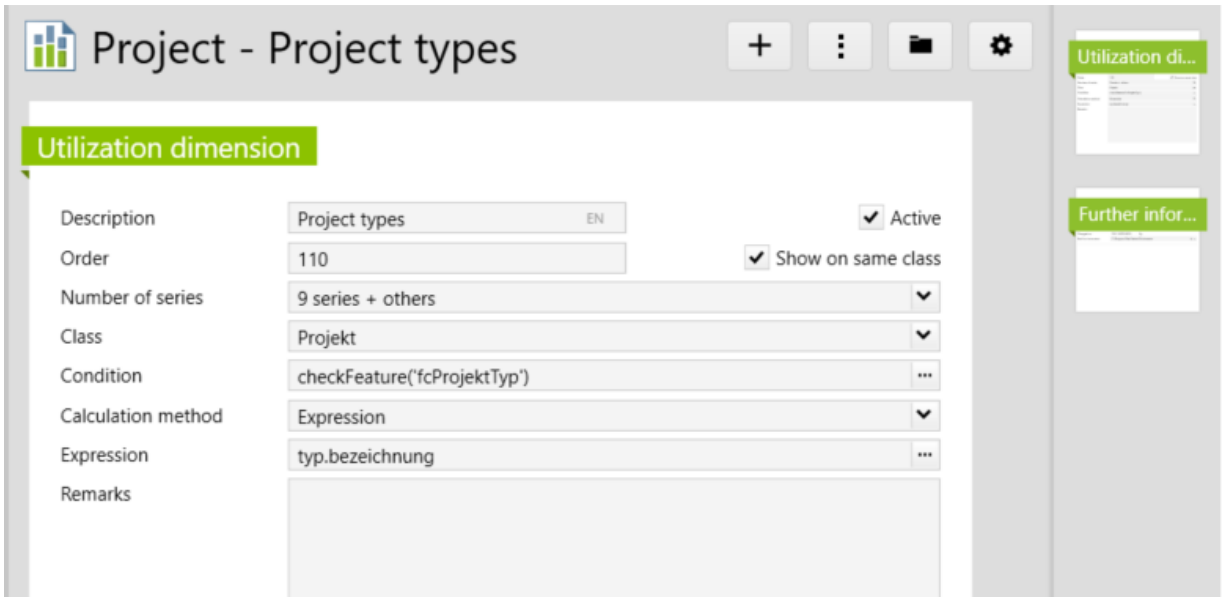



Figure 13 Example of a utilization dimension for the breakdown of projects by project type

| | |
|---|---|
| Desctiption | Name of the dimension. This name appears as button text on the detail view and can be stored in different interface languages if required. |
| Active | Controls whether this utilization dimension is applied. |
| Order | Here you can control the order in which the buttons for resource utilization charts are displayed on the detail view. These are sorted in ascending order.<br>We recommend to choose larger sorting steps (e.g. steps of 10). This makes it possible to insert buttons in between later on. |
| Show on same class | Controls whether the load dimension should also be displayed on the class of the selected object or only on the opposite side (default).<br>See also section **Eigene Klasse / Gegenseite** side below. |
| Number of series | Here you can set how many series (bar segments in the charts) are to be displayed. You can choose between:<br>– All<br>– 3 series and others<br>– 5 series and others<br>– 7 series and others<br>– 9 series and others<br>The corresponding number of series are then displayed as individual segments and the remaining values are shown totaled under `Other`. |
| Classes | Controls which class is available in this load dimension. Possible classes are:<br>– Projektbearbeiter<br>– PlanningWorker (see section 2.4)<br>– AbstractWorker (Both types of users mentioned above)<br>– Projekt/ProjektPhase (depending on the basic setting, see 2.1)<br>See also section **Own class / Opposite** below. |

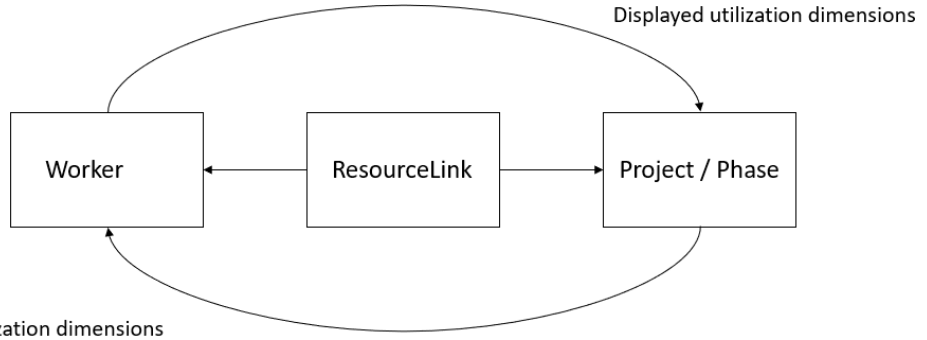
| | |
|---|---|
| Condition | Here, a display condition can be specified via OCL expression, which can be used to control whether or when this utilization dimension is displayed or not.<br>In the OCL editor, the base `Eintrag` (entry) is displayed. This is the object currently selected in the tree.<br>If the object is used in OCL, the administrator must write the code in such a way that it works for all combinations of class / single object / list. For this, the concrete type must be checked with `oclIsTypeOf` and/or cast with `oclAsType`.<br>The result of this expression must return a Yes/No value (Boolean).<br>In the resource view this expression is evaluated. If the expression is empty or the result is `true`, then the view is displayed. |
| Calculation method | The dimensional criteria can be calculated as follows:<br>– **Folder**: If this calculation type is selected, another field appears where a keyword folder can be selected. The objects are then displayed as bar segments according to their assignment (e.g. according to sectors, if the projects are stored in an sectors folder structure).<br>– If the keyword folder is not **Exclusive**, the dimension object is invalid. This invalid dimension cannot be selected in the chart.<br>– If the keyword folder is not **included**, the associated objects are displayed together in a bar segment "Other".<br>– **Expression**: An OCL expression can be specified here. The expression must return a string value as result. The bar segments correspond to the grouped results of this evaluation. Example: Grouping of plan data on projects by project type. The expression in this case would be `typ.bezeichnung`.<br>If a criterion does not match all the underlying objects (e.g. `bearbeiterstufe` (user level) does not match the planning users), then the non-matching objects are displayed grouped together in a bar segment **Other**. No error appears.<br>– **Python class**: A Python class can also be used for the calculation, in which the data is prepared accordingly. This makes it possible to define a completely own logic for the representation of the bars. While the other calculation methods only group the existing planning data (the sum of a bar is always the same), the values can be calculated completely independently using the Python class. For this, a so-called `UtilizationProvider` must be specified, see chapter 2.11. |
| Expression / Keyword folder / Python class reference | Depending on the set calculation method (see previous point), an `Expression, Keyword folder` or `Python class reference` field is displayed at this point and can be filled accordingly. |
| Remarks | Free text field for the description of the utilization dimension. This field has a purely informative character and no further functionality. |

**Own class / Opposite**

The utilization dimensions are always displayed for the opposite side. This means the following:

A Resource planning entry (so-called `ResourceLinks`) always contains a worker (Projektbearbeiter, PlanningWorker, AbstractWorker) on one side and a project or project phase on the other side, depending on the planning level (see 2.1).

The opposite side is always the other. So, if a utilization dimension is defined for the Project class, it will appear as a utilization chart on the user and vice versa.




Displayed utilization dimensions

By setting the checkmark "Show on same class", the utilization chart is additionally displayed on its own class, which can be useful on lists of objects (for example, the utilization dimension `Worker` on a team list):

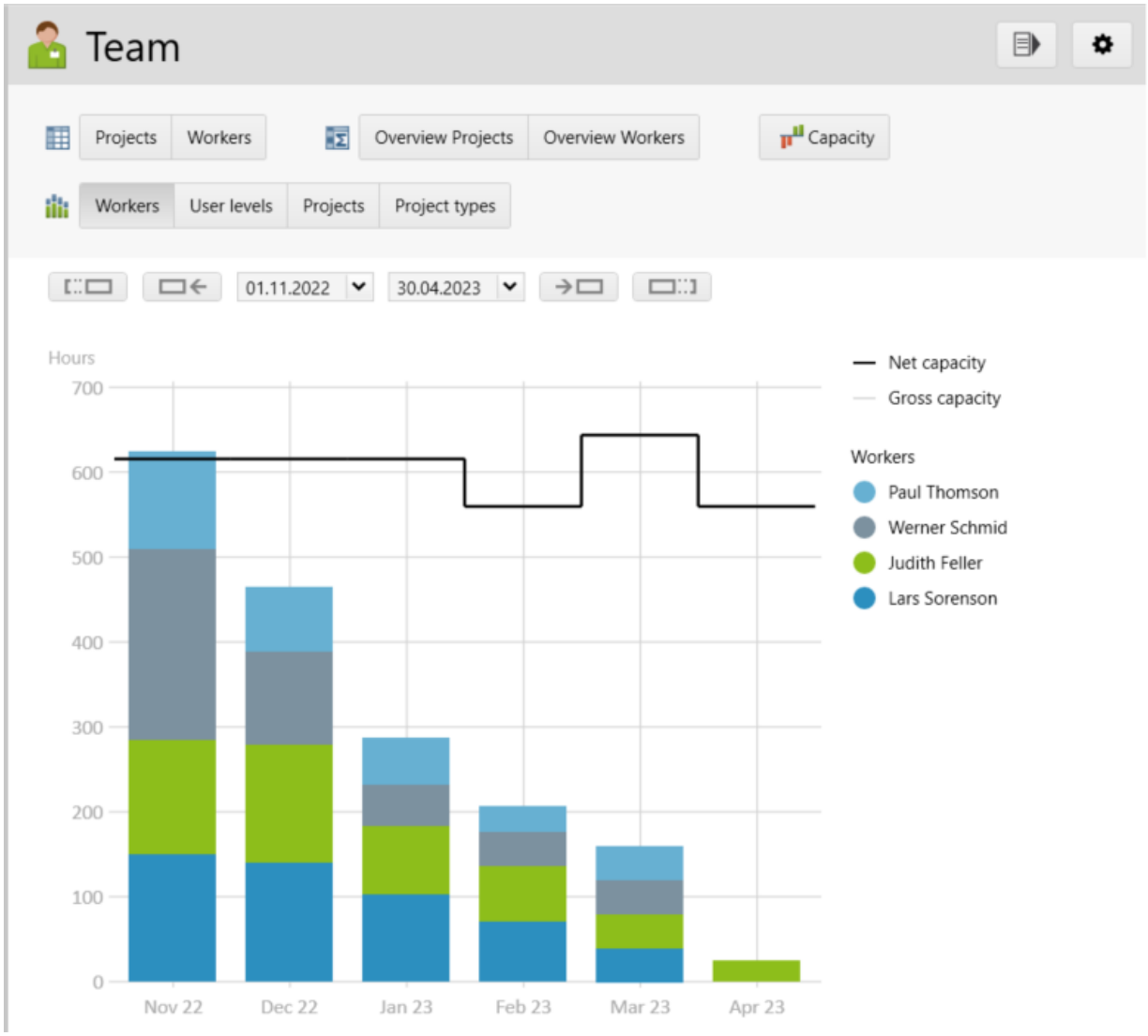



Figure 14 Utilization chart according to the dimension workers on a team overview

Caution is advised when calculating the dimensions via Python if the option "Show on same class" is activated: A case distinction must then be made here so that a utilization dimension works both on its own class and on the opposite class.

## 2.11 Example Utilization Dimension by keyword folder

In Vertec, it is possible to define your own categories by means of keyword folders and to assign entries to these categories. These can be, for example, locations, industries, skills, etc.

The resource plan data can be grouped according to these categories and evaluated in total by defining a utilization dimension for it.

Here in the example, we want to evaluate the resource plan data by sectors. For this purpose, there is a keyword folder Projects by sectors:

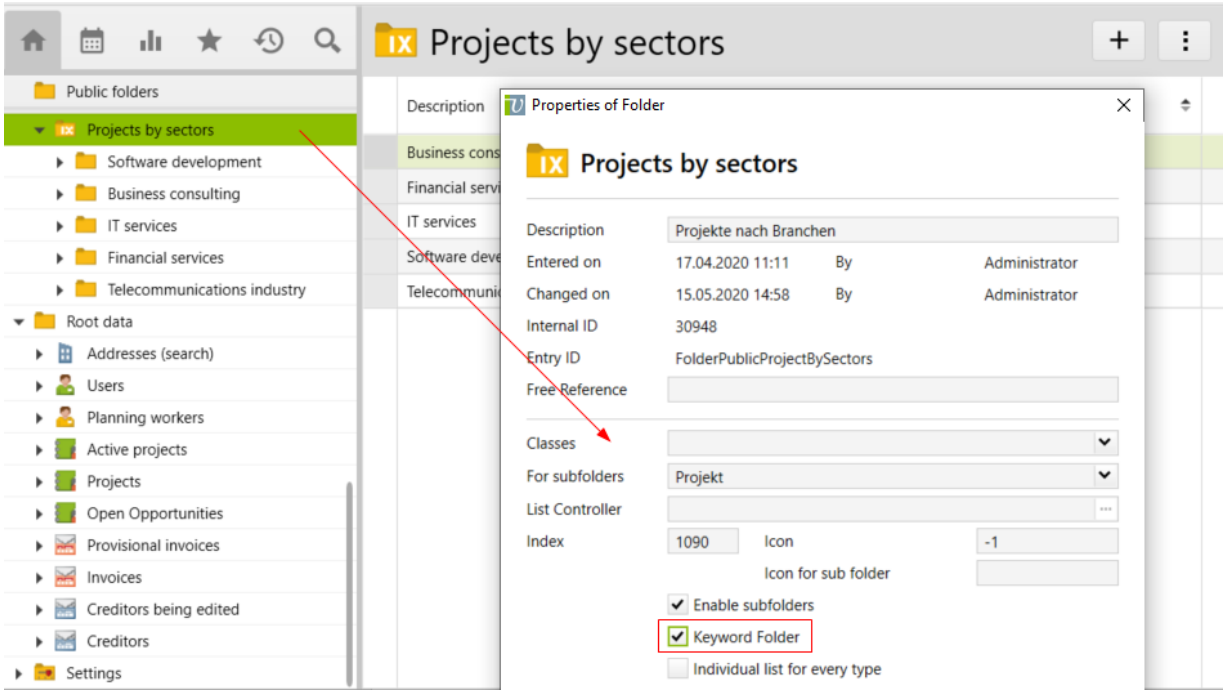



Figure 15 A keyword folder organizes the projects into branches

Now we create a new utilization dimension in the Settings > Resource Planning > Utilization Dimensions folder:

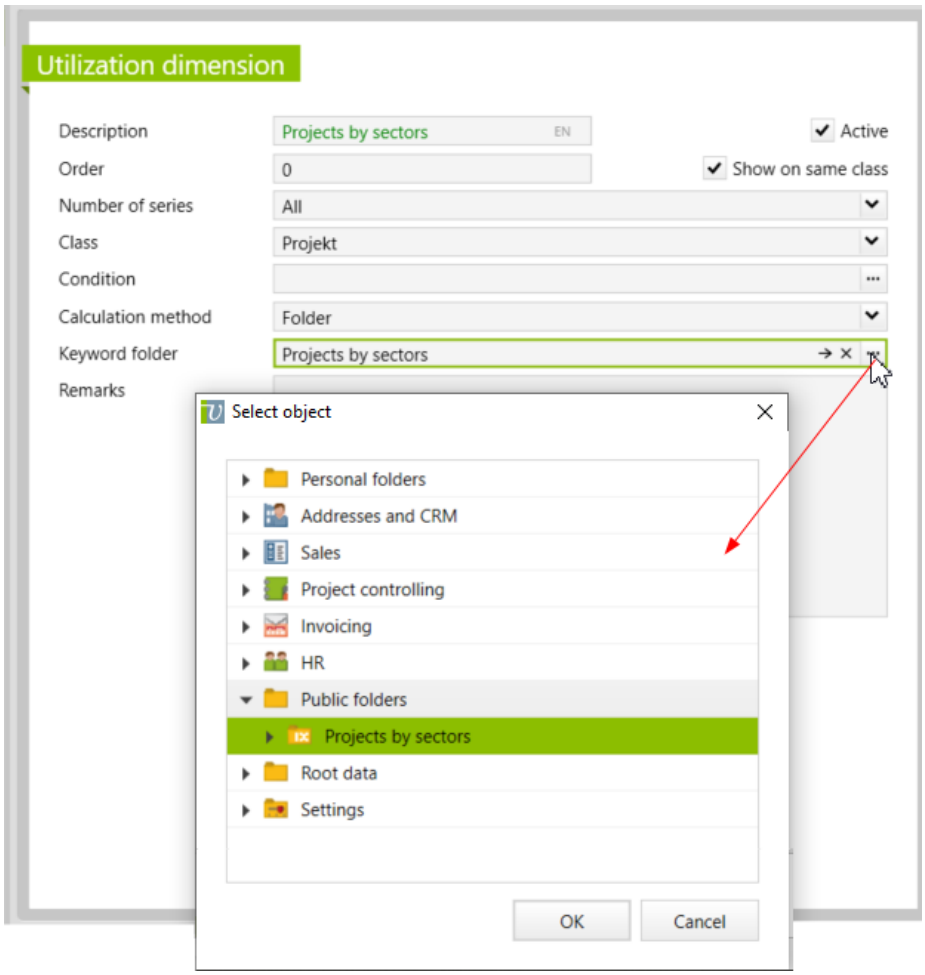



Figure 16 Utilization dimension by keyword folder

We select `Folder` as the **calculation method**.

Then a field **Keyword folder** appears below, where the desired keyword folder can be selected by means of buttons with the three dots.

As a **class** we select `Projekt`, because the keyword folder evaluates projects.

If we do not want to display the folder only on workers (by default, the utilization dimension is displayed on the opposite side, i.e. in our example on workers (AbstractWorker)), we check the box **Show on same class** (see also 2.10).

Now the evaluation will automatically appear on workers and, because we have set the checkmark, on projects:

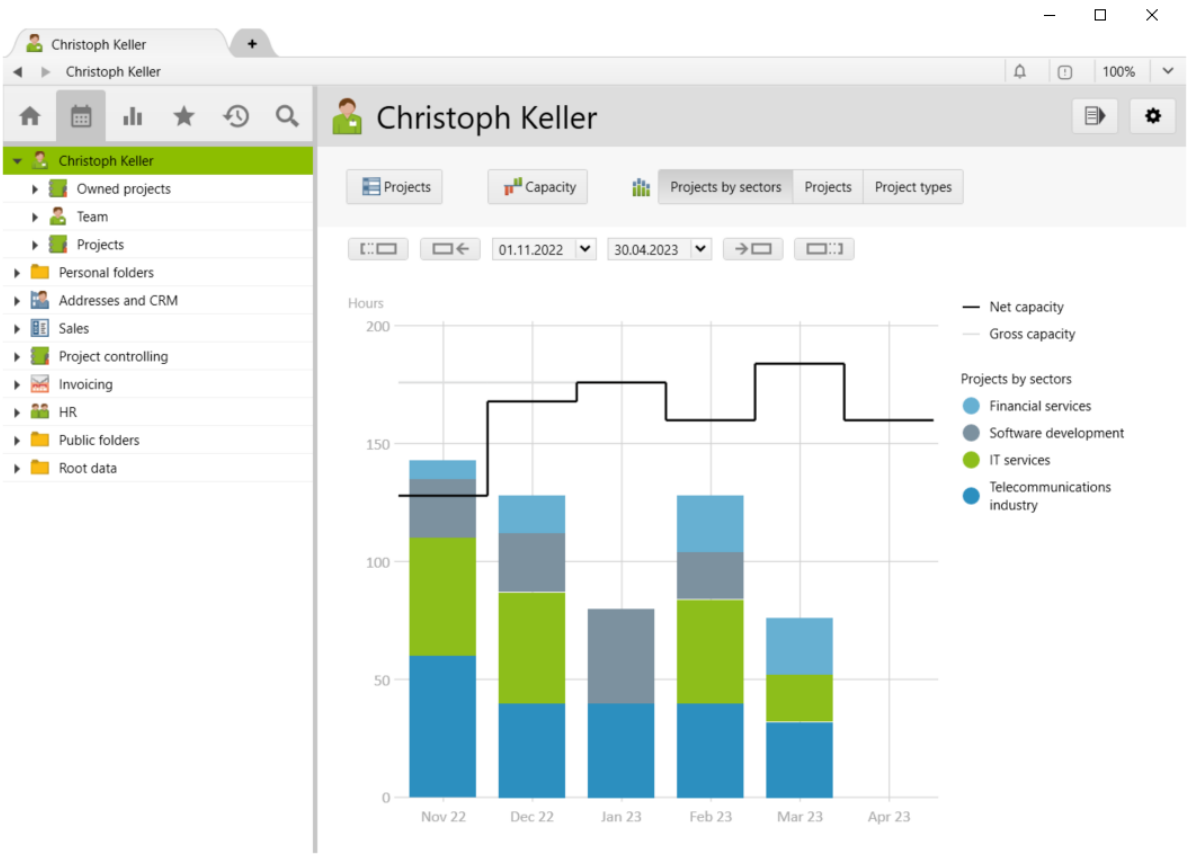



Figure 17 The utilization chart is displayed grouped and totaled by sector

For the absolute figures, the chart can then additionally be exported to Excel.

## 2.12 Calculating Utilization Dimensions via Python

The most extensive control over the calculation of a utilization chart allows the calculation method via Python (see Chapter 2.10).

The calculation is done by a `UtilizationProvider` class, which defines the following methods:

| | |
|---|---|
| initialize(self, entries, startdate, enddate, subscriber) | Is called at the beginning of the calculation and initializes the provider with the initial objects for the evaluation (single object or content of a selected container) and the period to be represented. |
| generate(self, start, end, subscriber) | Called for each bar (interval) of the chart.<br>The return value is the list of segments for the bar as a tuple (caption, value).<br>The generation of the return list can be done (as with BI generators, see www.vertec.com/kb/bi-generatoren/) with the Python `yield` operator. |
| get_series_properties(self) | Optional method. Is called at the beginning of the calculation and allows to specify the representation of the individual segments even more precisely.<br>For each possible segment caption, a dictionary with specifications for:<br>– caption<br>– orderIdx and<br>– color<br>are returned. |

Example of a provider generating a utilization chart by productive/unproductive (Planning Level `project`, see 2.1)

```
import vtcplanningcore
import vtcplanning

class ProdInternalUtilizationProvider(object):
    productive_label_text = vtcapp.translate('label_productive')
    internal_label_text = vtcapp.translate('label_internal')

    def initialize(self, entries, start, end, subscriber):
        self.provider = vtcplanningcore.ResourcePlanningProvider(
            entries, start, end)
        self.entries_are_workers = vtcplanning.is_abstractworker(entries)


        if self.entries_are_workers:
            self.projects = self.provider.get_otherside_entries(subscriber)
        else:
            self.projects = entries

    def get_series_properties(self):
        prod_series = {
            'caption': self.productive_label_text,
            'orderIdx': 1,
            'color': 'clDarkRed',
        }
        internal_series = {
            'caption': self.internal_label_text,
            'orderIdx': 2,
            'color': 'clLightRed',
        }
        return [internal_series, prod_series]

    def generate(self, startdate, enddate, subscriber):
        """ Generates a list of series. Each series is identified by
        its name in the return value """
        for project in self.projects:
            if self.entries_are_workers:
                planned_minutes = self.provider.get_planned_minutes_aggregated(
                    None, project, startdate, None, subscriber)
            else:
                planned_minutes = self.provider.get_planned_minutes_aggregated(
                    project, None, startdate, None, subscriber)

            # self.evalocl subscribes implicitly
            if self.evalocl("typ.produktiv", project):
                yield (self.productive_label_text, planned_minutes)
            else:
                yield (self.internal_label_text, planned_minutes)
```

The class must be defined in a script module. On the utilization dimension definition (2.10) it is then referenced as `<module>.<class>`.

## 2.13 Utilization dimensions supplied

The following utilization dimensions are supplied by Vertec as standard:

– **Projects** (for class `Projekt`)
– **Project types** (for class `Projekt`)
– **Workers** (for class `AbstractWorkers`)
– **User levesl** (for class `Projektbearbeiter`)
– **Phase status** (for class `Projektphase`)

## 2.14 Authorizations in Resource Planning

The following permissions are relevant for querying resource plan data:

A new **ResourcePlanning** right type has been introduced:

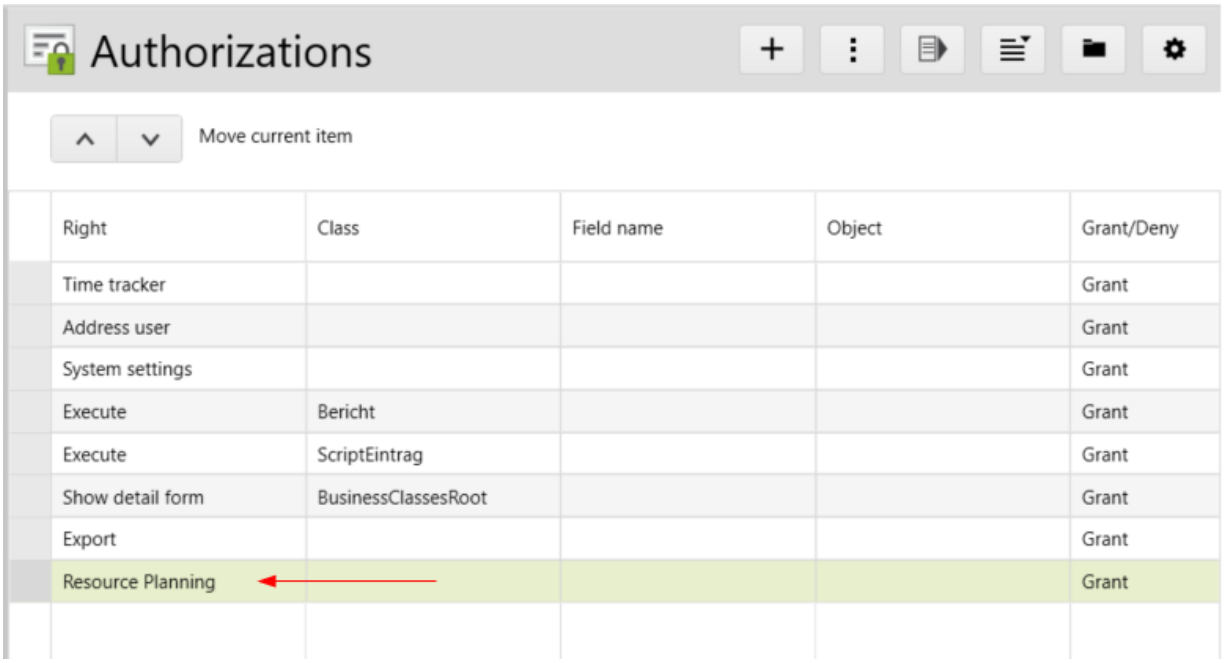



Figure 18 The new Resource Planning right-type

This is added by default to the "Standard Users" group if the Resource Planning module is licensed and controls whether the Resource planning navigation button (see Introduction in Chapter 2) is visible to the logged-in user.

In the Resource planning itself, the access rights to a plan data cell are based on the read and write rights to the `resourcelinks` attributes of the business objects involved (User, Project, Phase), which are the actual plan data of these combinations. The access rights from both sides are linked with OR, so the access right to one side is sufficient.

The following standard user rights are available:

– The Users have read and write access to `resourcelinks` of themselves.
– Users normally have no rights to `resourcelinks` from other users, except:
  – All Users have access to Planning Workers (see chapter 2.4) and thus also to their `resourcelinks`.
  – Team Leaders with team leader rights have read and write access to `resourcelinks` of users of their team.

– Project Managers have read and write access to the projects on which they have project manager rights.
– Project Supervisors have general read and write access.

The Resource planning table views implement authorization checks according to the logic described above as follows:

– If there is read-only permission, the cell will be displayed but will not be writable.
– If read and write permission exists, the cell is writable.
– If there is not read permission on ALL `resourcelinks` members of projects or phases in the list, then the Resource planning View will not be displayed (as otherwise conclusions could still be drawn about the planning).

Graphics (utilization and capacity charts, see chapter 2.2) are only displayed if the logged-in user has access to ALL displayed planning data. Otherwise, a "No access" display appears instead of the graphic.

In the planning mode **User-Phase-Links** (see chapter 2.15) an analogous logic is applied due to project and user. Instead of the `resourcelink` attribute, in this case the authorization is checked on the attributes `bearbeiterPhasen` of users and phases.

## 2.15 Planning on User-Phase-Links

Line: Expert | Module: Budget & Phases, Resource Planning | Version: 6.6.
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

As an alternative to normal Resource planning, it is also possible to plan on the link of users to phases (so-called User-phase-links, see www.vertec.com/kb/projektphasen) instead.

For this purpose, the **plug-in: Resource planning with User-phase-links** must be imported. This can be downloaded from www.vertec.com/kb/plug-in-ressourcenplanung-mit-bearbeiterzuordnungen.

This creates the property **Type of resource planning** in the **System Settings > Resource planning** with change from `Standard` to `User-Phase-Link`.

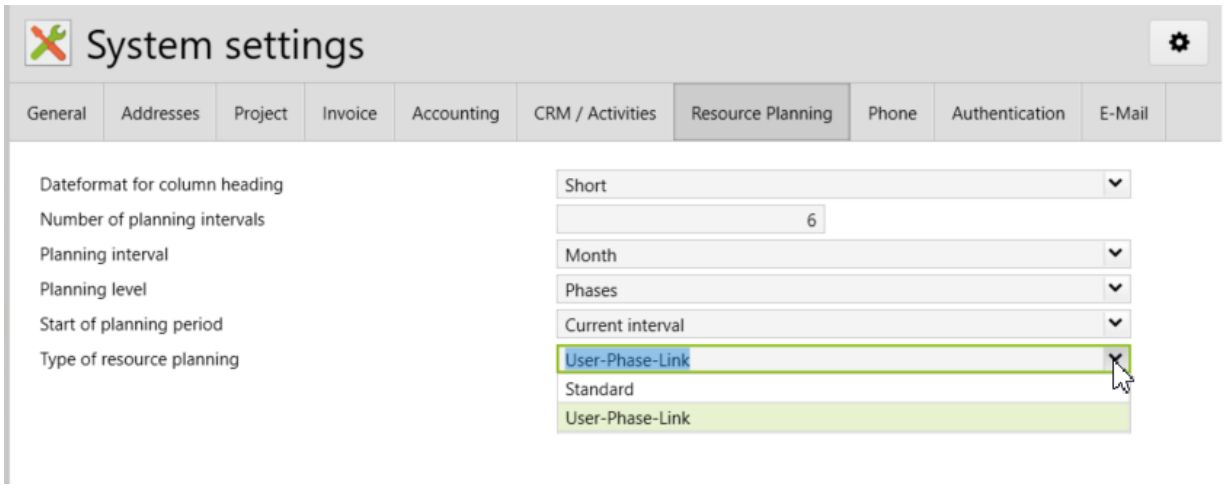



Figure 19 The system setting "Type of resource planning" is generated by the plug-in and set to User-Phase-Link

Resource planning is then based on the link of users to phases:

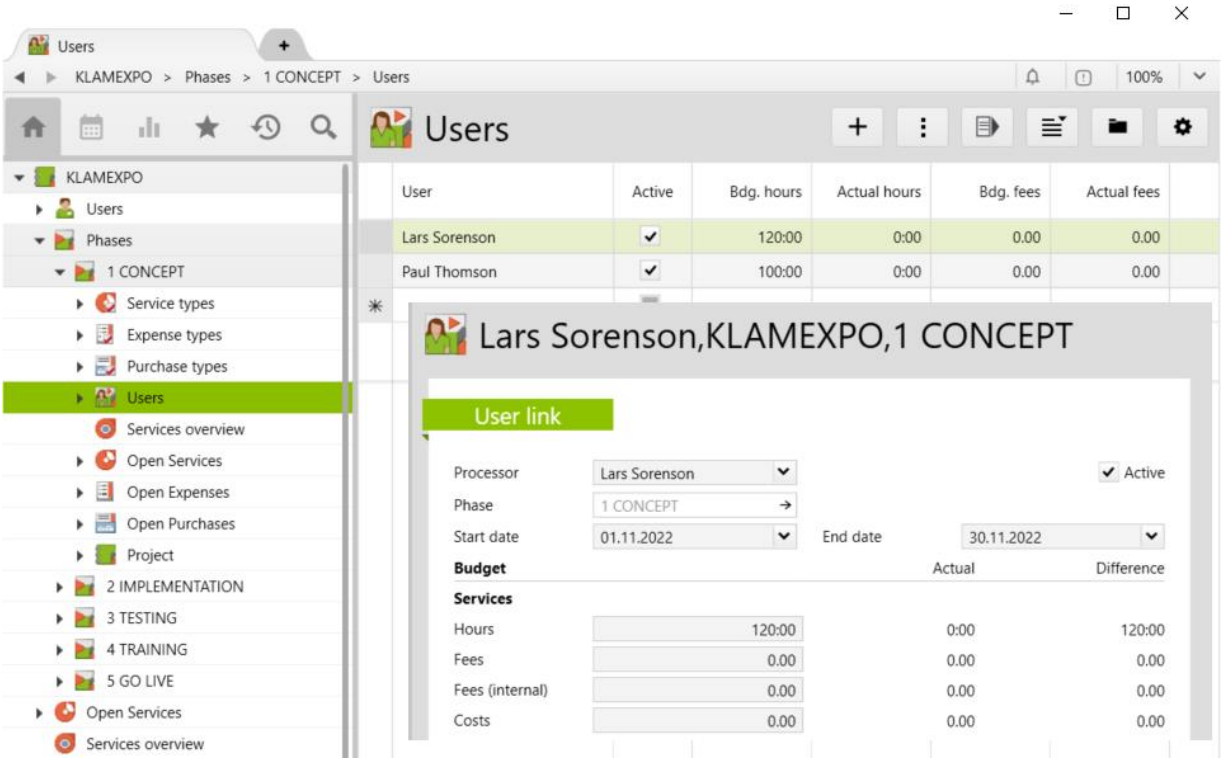



Figure 20 On a User link to a phase the required effort is planned

– The User-phase-links now have the attributes `StartDate` and `EndDate`.
– The calculated (derived) attributes `StartDate` and `EndDate` on project phases consider the data of all user links, analogous to the service type links.

An effort budget is set on the link of a user to a phase. Here in the example, Lars Sorenson is scheduled for 120 hours for the concept phase from 11/1 - 12/31/2022, Judith Feller for 100 hours.

In the Resource planning view you can see the distribution on the phase:

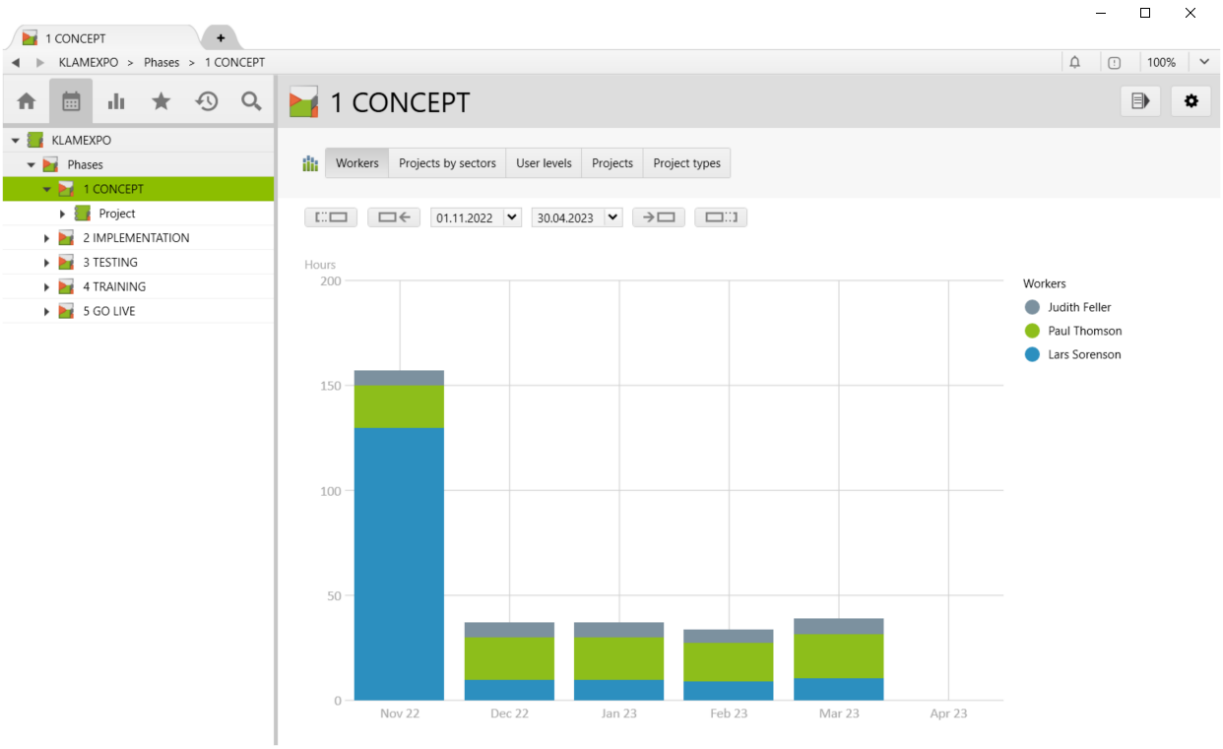



Figure 21 Utilization chart of a planning on user-phase link

The following rules apply:

– Planning is always done in the user-phase-links. No planning data can be entered in the time and pivot tables (see 2.2) of Resource planning. These are read-only in this case.
– The workload of an employee assigned by user-phase-link is calculated from the budget of the user-phase-link. The time distribution of the workload results from a distribution over the runtime of the user-phase-link (start-end date).
– The distribution to the individual planning intervals (day, week, month) is calculated via resource time. For each interval, the resource time is calculated and thus the share of the interval in the total phase runtime is determined.
– User-phase-links without budget are included in the calculation with 0 minutes. There is no distribution of phase budgets to users.
– User-phase-links without start and end dates are not considered.
– For the display of plan values from User-Phase-Links in the Resource planning views, the objects that are relevant for the displayed period due to their start and end data are considered.
– No planning workers (see 2.4) can be assigned. Planning is carried out exclusively on project workers.

## 2.16 Python Methods for Resource Planning

The following Python methods are available for Resource planning, which can be used to set Resource planning values via Python code:

On AbstractWorker, ProjectWorker and PlanningWorker (see 2.4) there is the method

```
setresourceplanvalue(project_or_phase, date, value)
```

On Project und Phase is the Method

```
setresourceplanvalue(worker, date, value)
```

– **date**: The period in which the date is located is filled with the value.

– **value**: Value in minutes.

## 2.17 OCL Variables for Resource Planning

The following new OCL variables are available for Resource planning:

| | |
|---|---|
| varView | Specifies the type of list view. Possible values are:<br>– **default**: a standard list view (container)<br>– **resources**: a Resource planning view (container or user entry, configured via Resource planning view (see 2.5))<br>– **bi** a BI (Business Intelligence) view.<br>The variable can be used in the condition of scripts to control in which views the script appears in the menu, e.g.<br>`(varView='default') or (varView='bi')` |
| varStartDate | The start date of the Resource planning view (see 2.5):<br>– Time Table: Start-Date of the first interval in period<br>– Pivot Table: Start-Date of the Interval |
| varEndDate | The end date of the Resource planning view (see 2.5):<br>– Time Table: End-Date of the last interval in period<br>– Pivot Table: End-Date oft he Interval |

These are available in Resource planning views and the associated List Controllers (see Chapter 16) as well as in the Custom Renderers (see chapter 3).

In order to be able to use the OCL variables easily in List Controllers and Custom Renderers, both provide the **self.evalocl()** method, which uses the specific OCL evaluator on which these variables are defined. The mentioned OCL variables are only defined in this evaluator (so vtcapp.evalocl() does not know them).

The existing container variables **varContainer** and **varParent** contain `null` in the case of a single object.

## 2.18 Backwards Compatibility / Migration

For existing customers using the current Resource planning, please note the following:

**Fixed planning interval**

In the new Resource planning you can only plan on one planning interval: day, week or month.

When version 6.6 is started for the first time, all existing Resource planning data is converted. Therefore, it is important that the desired planning interval is selected in the system settings **before the upgrade** so that the Resource planning data is converted correctly.

**Fixed planning level**

In the new Resource planning, planning can only be done on one planning level: **Project** or **Phase** (planning level Phase only available with module Budget & Subproject).

If planning has been done on both levels so far, some of the data will not be visible after the upgrade. (e.g., if you set the planning level to Project, all planned values on Phases will no longer be visible).

For these cases, we provide a plug-in that helps to harmonize the planning data. You can find it in the Online Knowledge Base in the **plug-in: Resource planning Maintenance** at www.vertec.com/kb/plug-in-ressourcen-planung-maintenance/.

### Excel-Reports

No more Excel reports will be delivered and the existing ones will be deactivated. See also chapter 8.1.

The Resource planning views (see 2.5) and the utilization dimensions (see 2.6) are new configurations that replace these reports. All Resource planning views can be exported to Excel, see chapter 2.3.

If customer-specific reports exist, they must be deleted manually or, if necessary, tested again after the upgrade.

### Link Types removed

The new Resource planning completely abandons the old link types of the existing Resource planning:

– User – Resource planning
– User – Resource planning for phases
– Project – Resource planning
– Phase – Resource planning

These link types will be deleted with the upgrade to Vertec 6.6 and therefore all related customizing.

### Classes removed

The Classes:

– ResourceContainer (`ResourcenContainer`)
– ResourceRow (`ResourceZeile`)
– ResourceColumn (`ResourceSpalte`)

will be deleted with the upgrade to Vertec 6.6 and therefore all related customizing.

### System Setting removed

The previous system setting "Show Resource planning for detail row" (internal name: `ShowResourceSubDetail`) will be deleted with the upgrade to Vertec 6.6.

### Setting on Prerequisites Page removed

The setting "Planning level for Resource planning" on the page `Prerequisits` for projects will be removed with the upgrade to Vertec 6.6.

If this field is used in self-configured pages, it will remain, but will no longer do anything useful. We recommend removing this field.

### OCL-Expressions

The existing OCL operators for Resource planning (see www.vertec.com/kb/ocl/#oclressourcenplanung) continue to work. So, no adjustments to existing customizing have to be made here

### Python-Methoden

The Python-Methode

```
vtcapp.setresourceplanvalue(bearbeiter, projekt, phase, date, intervalType, value)
```

will still be executable for backwards compatibility reasons. However, the intervalType parameter is no longer observed, since the planning interval is fixed as of Vertec 6.6 and is always taken from the system settings.

There is now the same method **without** the **intervalType** parameter on the individual business objects, see the chapter 2.16)

### ResourceLinks with unique project/phase reference

During the migration, `ResourceLinks` (see section Own class / Opposite in chapter 2.10), which had referenced both a project and a phase, are cleaned up. The (redundant) project reference will be deleted.

# 3 Custom Renderer

Line: Expert | Module: PSA | Version: 6.5.0.21
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

With Custom Renderers it is possible to configure the display and calculation of the underlying data of list cells and fields on pages themselves.

In contrast to the built-in renderers, which - inserted in the column configuration of list settings - apply a predefined and unchangeable operation to the displayed values, Custom Renderers can be used to define their own operations, which are applied when displaying and setting data.

So, while in a normal cell input and output must match, this process can be done separately using Custom Renderers. This is important, for example, for calculated (derived) attributes, which are not writable.

Icons and buttons can also be displayed directly in the cells and thus, for example, scripts can be executed directly on a list object or the list object can be linked to another object.

## 3.1 Attributes and Methods

The Custom Renderers are created as Python scripts. For each Custom Renderer, a corresponding class is defined, in which the individual attributes or methods are then set.

The Custom Renderer is entered as a renderer in the corresponding column in the list settings.

The call of a renderer is then done via `<Scriptname>.<Rendererclass>`. Therefore, it is recommended to assign speaking names.

A Custom Renderer class can define the following attributes and methods:

**Parameters**
– **rowobj**: Current row object in the list or current object of the page.
– **expression**: The expression given, In the list settings the value from the field with the same name, in the page settings the `ValueExpression`.
– **subscriber**: Optional notification of member changes, see section 3.2.

| Attribute / Method | Description |
|---|---|
| **value** / **get_value**(self, rowobj, expression, subscriber) | This is the actual Custom Renderer method. It specifies what will be displayed in the cell.<br>A Custom Renderer without value or get_value() will not display anything. So, this value or method must always be set.<br>`def get_value(self, rowobj, expression, subscriber):`<br>`  return self.evalocl(expression, rowobj)` |
| **set_value**(self, rowobj, expression, value) | If the cell is writable, the value entered by the user is written using this method.<br>`def set_value(self, rowobj, expression, value):`<br>`  setattr(rowobj, expression, value)`<br>It must be ensured that the passed value has the desired type. By default, `value` is a string value. For all data types that cannot be written as string, a suitable control (`display_type`) must be used or the `value_type` must be set accordingly. |

| Attribute / Method | Description |
|---|---|
| **value_type** /<br>**get_value_type**(self) | Allows to specify the expected data type for the Custom Renderer.<br>The `set_value` method (see above) then receives a value of the specified type as an argument, and the `get_value` method is expected to return that type.<br>The following values are allowed:<br>– integer<br>– currency<br>– float – floating point numbers<br>– date – dates and date values<br>– datetime – date and time<br>– boolean: values are either True or False<br>– string: text content<br>– blob: bigger text content and images<br>Application example: A Custom Renderer works with hourly rates (currency). In order for the set_value method to get a currency value, the get_value_type method must be defined or the value_type must be set to currency:<br>`class Ansatz:`<br>`  value_type = "currency"`<br>`  def get_value(self, rowobj, expression, subscriber):`<br>`  return self.evalocl(expression, rowobj)`<br>`  def set_value(self, rowobj, expression, value):`<br>`  setattr(rowobj, expression, value)` |
| **display_type** /<br>**get_display_type**(self)<br>Evaluated per column, therefore `rowobj`, `expression` and `subscriber` objects are not included. | Permitted values are:<br>– **checkbox**: The column represents a checkbox. The corresponding get_value and set_value methods then process boolean values.<br>`class Aktiv:`<br>`  display_type = 'checkbox'`<br>`  def get_value(self, rowobj, expression, subscriber):`<br>`  return self.evalocl('aktiv', rowobj)`<br>`  def set_value(self, rowobj, expression, value):`<br>`  rowobj.aktiv = value`<br>– **icon**: The column is represented as an icon. The get_value function must return an integer (icon index) or string (icon name).<br>`class Icon:`<br>`  display_type = 'icon'`<br>`  def get_value(self, rowobj, expression, subscriber):`<br>`  return self.evalocl('iconindex', rowobj)` |

| Attribute / Method | Description |
|---|---|
| | – **button**: The cell displays a button.<br>The `get_value` method is used to find out what is displayed as the button text.<br>Additionally, an icon can be added by setting the `buttoniconid` or calling the `get_buttoniconid` method. This must return an integer (icon index) or string (icon name).<br>With `buttontooltipp` or the `get_buttontooltipp` method a tooltip can be displayed, e.g. to inform the user what the click on the button does.<br>The `button_clicked` method can be used to define what should happen when the user clicks on the button. |
| **buttoniconid** /<br>**get_buttoniconid**(self, rowobj, expression, subscriber) | `display_type = "button"`<br>`def get_buttoniconid(self, rowobj, expression, subscriber):`<br>`  return self.evalocl('iconIndex', rowobj)`<br><br>If nothing is specified, the `flash` icon is displayed.<br>If an empty string "" is explicitly specified, the button for this cell will be hidden. |
| **button_clicked**(self, rowobj, expression) | `display_type = "button"`<br>`def button_clicked(self, rowobj, expression):`<br>`  return vtcapp.msgbox("Hello World")` |
| **buttontooltipp** /<br>**get_buttontooltip**(self, rowbj, expression, subscriber) | `class simpleBtn:`<br>`  display_type='button'`<br>`  buttontooltip='Hier ein Tipp'`<br><br>`class complexBtn:`<br>`  display_type='button'`<br>`  def get_buttontooltip(self,rowObject, expression, subscriber):`<br>`  return self.evalocl("bemerkung", rowObject)` |
| **converter** /<br>**get_converter**(self)<br>Evaluated per column, therefore `rowobj`, `expression` and `subscriber` objects are not included. | A converter is used to display values formatted in a certain way, for example time values as hours:minutes, etc.<br>The following converters can be used:<br>– **minutes**: For integer values. Represents minute values according to system setting Display minutes.<br>– **noroundminutes**: For integer values. Same as minutes, but ignores the system setting Round off minutes to.<br>– **rate**: For currency values. For rates, considers the system setting Don't round rates (when using gross rates).<br>– **translation**: For string values. The string is translated according to the Vertec translation system.<br><br>`class Minutes:`<br>`  value_type = "integer"`<br>`  converter = "minutes"` |

| Attribute / Method | Description |
|---|---|
| | `def get_value(self, rowobj, expression, subscriber):`<br>`  return rowobj.evalocl(expression)`<br>`def set_value(self, rowobj, expression, value):`<br>`  setattr(rowobj, expression, value)` |
| **is_readonly** /<br>**get_is_readonly**(self, rowobj, expression, subscriber) | This can be used to control whether a cell is read-only or not.<br>Possible values: `True` or `False`. Default value: `False`. |
| **is_cascadedset** /<br>**get_is_cascadedset**(self, rowobj, expression, subscriber) | More experienced Vertec users know the behavior that the font turns green if you overwrite certain values and is black if it is a calculated value, for example in project budgeting.<br>This way you can see at first glance on the interface whether a value has been set on the object itself (i.e. some standard has been overwritten), or whether it is the default value.<br>This can be emulated with `is_cascadedset` or `get_is_cascadedset`:<br>– The **get_value** looks if a certain field is described and takes the default value otherwise.<br>– The **set_value** describes the certain field<br>– Das **is_cascadedset** is True if the certain field is described.<br>Possible values: `True` (green font) or `False` (black font). Default value: `False`. |
| **text_color** /<br>**get_text_color**(self, rowobj, expression, subscriber) | This can be used to control the font color. All values of the Vertec color palette are valid (see also 13.1).<br>Standard: The Vertec standard applies. |
| **background_color** /<br>**get_background_color**(self, rowobj, expression, subscriber) | This can be used to control the background color of the cell. All values of the Vertec color palette are valid (see also 13.1).<br>Standard: The Vertec standard applies. |

The object instance (self) offers the following attributes and methods:

| Attribute / Method | Description |
|---|---|
| **self.evalocl**(expression, rootobj=None) | Allows evaluating OCL expressions on the object.<br>– **expression**: The column or field expression<br>– **rootobj**: Optional. Any Vertec object can be specified here. If specified, the evaluation is performed on this object, otherwise globally.<br>In the OCL Evaluator a variable `varRowObject` is available. This allows direct access to the current row object in the list.<br>`self.evalocl("varRowObject")` |
| **self.container** | Provides access to the container business object of the list view or page. |
| **self.context** | Refers to the context object. For normal lists, the context object is the container (folder or link container). For Resource planning views (see 2.5), the context object can also be a single user entry. |

| Attribute / Method | Description |
|---|---|
| **self.controller** | In use with List Controllers (see 16) the reference to the List Controller object: If in a list with a List Controller a column uses a Custom Renderer, then the Custom Renderer gets a reference to the List Controller under `self.controller`. This is the same object for all Custom Renderers in the list.<br>This allows, for example, to precompute data in the `initialize()` method of the List Controller, which is then available to all Custom Renderers of the list. |

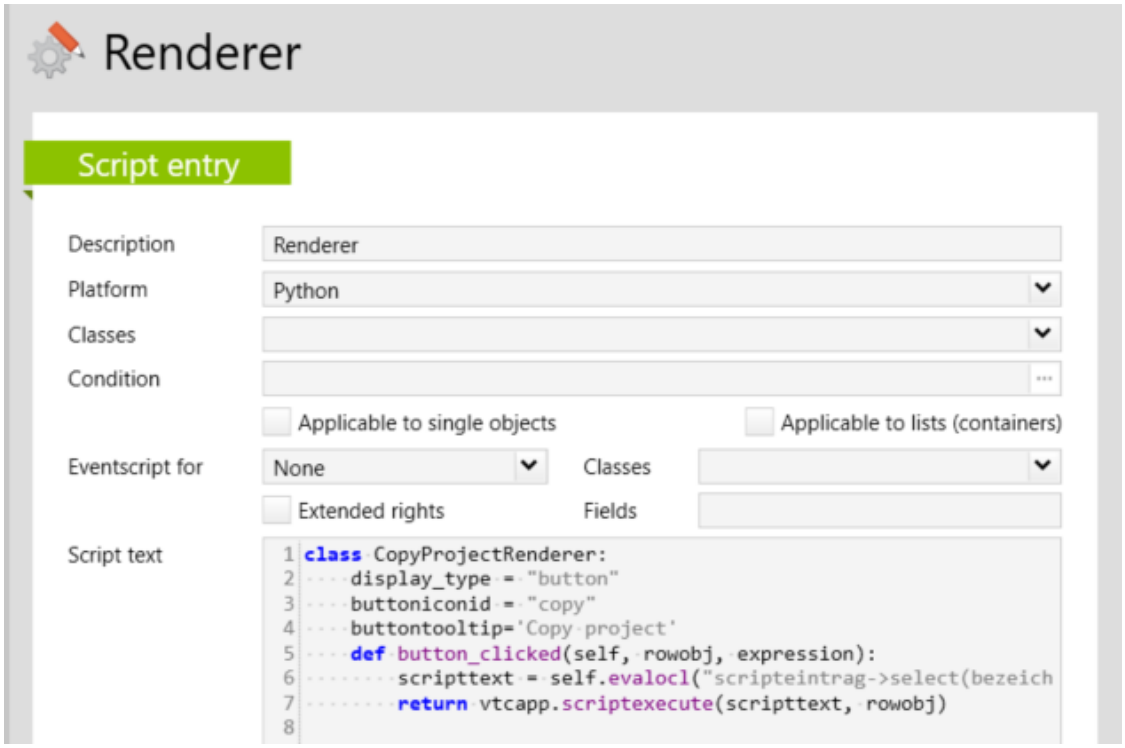
**Note:** With Restrict Scripting, Custom Renderers can be used, but the Python code itself is subject to the usual restrictions associated with it.

## 3.2 Example

In a project list a button is displayed which executes the script **Copy Project**.

The Copy Project script comes from the KB and is registered as a script in Vertec.

We create a Custom Renderer:




with the following script text:

```
class CopyProjectRenderer:
    display_type = "button"
    buttoniconid = "copy"
    buttontooltip='Copy project'
    def button_clicked(self, rowobj, expression):
        scripttext = self.evalocl("scripteintrag->select(bezeichnung='Copy
```

```
project')->first.scripttext")
        return vtcapp.scriptexecute(scripttext, rowobj)
```

Then we specify the Custom Renderer in the list settings of a project list:

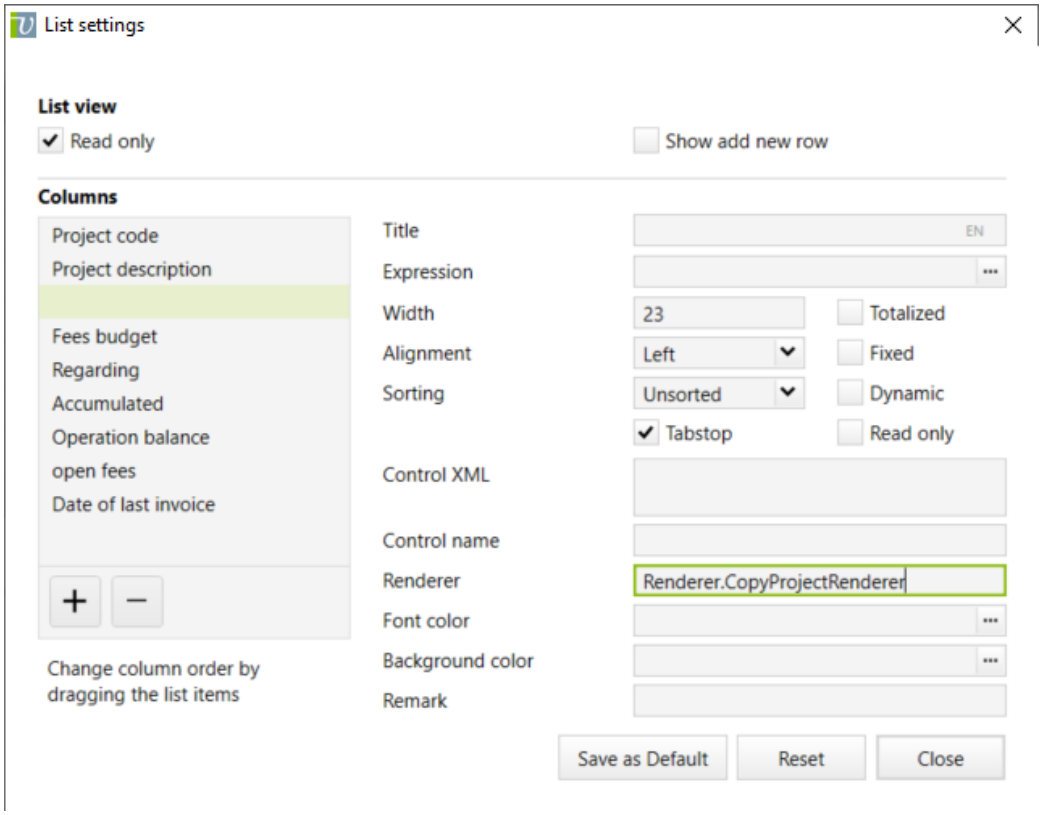



Figure 22 The Custom Renderer is specified in the list settings

The project list will now show the button:

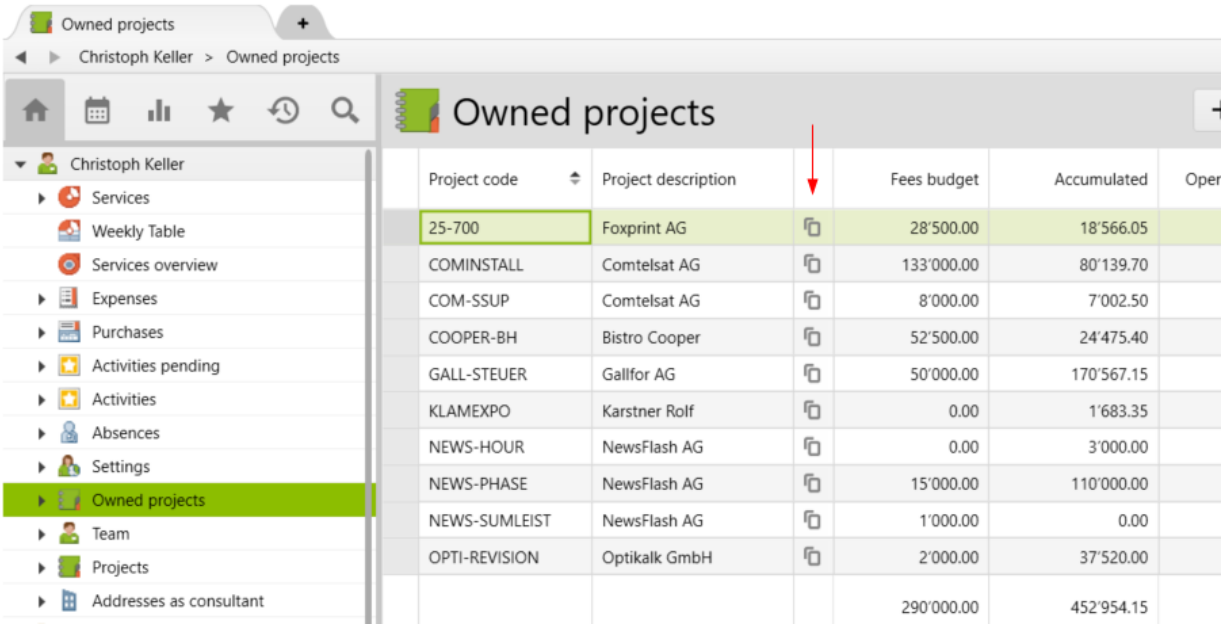



Figure 23 With the button "Copy project" you can now execute the script directly and copy the project in this line

**Further Information** The entire feature is described in the Knowledge Base at www.vertec.com/kb/custom-renderer/.

## 3.3 Subscriber

"Subscribing" means that you tell a certain member that you want to be notified when it changes. You subscribe to the notification list of this member, so to speak.

If you evaluate the return value of `get_value()` via OCL (`self.evalocl()`), this happens automatically. In this case, `get_value()` is called again when the underlying data changes. The OCL evaluator automatically subscribes to the OCL evaluation and additionally to the result member. In this case there is no need to specify a subscriber.

For this `self.evalocl()` must be used. Additionally, the `evalocl` methods on Vertec objects support an optional subscriber argument, where the **subscriber** parameter can be passed.

**Example**

| Expression | Creates Subscription? |
|---|---|
| `aktivitaet.evalocl("entry")` | No |
| `self.evalocl("entry", aktivitaet)` | Yes |
| `aktivitaet.evalocl("entry", subscriber)` | Yes |

To set subscriptions via Python code, there is the method.

`subscribetomember(membername, subscriber)`

on all Vertec objects.

Example with OCL and without subscriber

```
def get_value(self, rowobj, expression, subscriber):
    return self.evalocl("code", rowobj)
```

Example with Python and with Subscriber

```
def get_value(self, rowobj, expression, subscriber):
    rowobj.subscribetomember("code", subscriber)
    return rowobj.code
```

It is important to subscribe to all members from whom you want to see changes.

A manually set subscription is only ever set on the corresponding member, so it does not make a "chain". We therefore recommend always using `self.evalocl()`.

# 4 Notifications

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

With the new Expert feature **Notifications**, customizable notifications are available to the user in Vertec. This means that certain people can be notified automatically for defined processes in Vertec. This allows workflows to be supported in Vertec. Some of them are delivered via plug-ins (see 4.1), others are customer-specific and can be created by the user (see 4.3).

There is now a bell icon in the title bar. The red dot indicates reminders or tasks to be performed (e.g. activities to be completed or absences to be released).

Clicking on the bell icon opens the list of notifications:

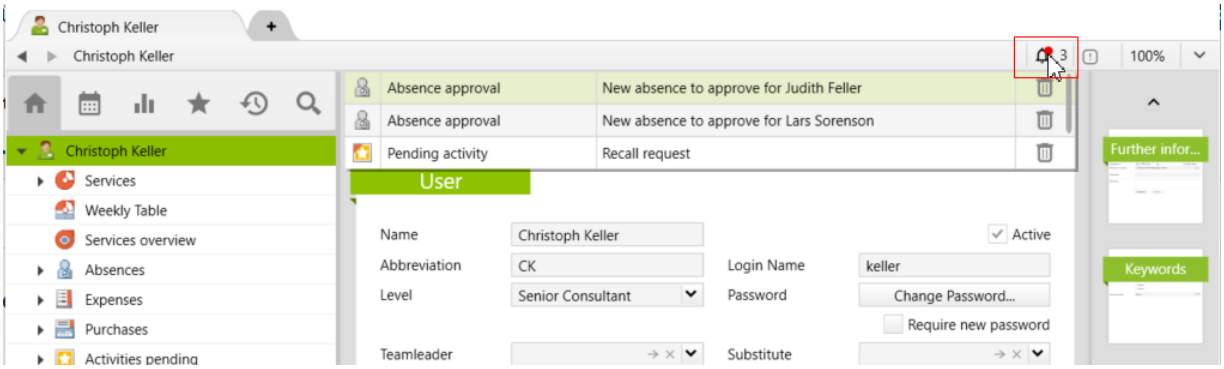



Figure 24 List of Notifications

In the list of notifications, you can navigate directly to the desired entry by double-clicking on an individual notification.

– `Ctrl+DoubleClick` opens the notification in a new tab
– `Shift+DoubleClick` opens it in a new window
– Using the trash can button, a notification can be deleted directly in the list.

## 4.1 Notifications supplied by Vertec

Vertec also provides two plug-ins with the release of Vertec 6.6, which can be loaded and implement the following notifications:

**Notifications for absence approvals:**
This plug-in can be found in the Knowledge Base in the **Plug-in: Absence approvals** at www.vertec.com/kb/plug-in-abwesenheitsfreigaben/.

– When a new absence is created, a notification is created for the team leader
– When the absence is released, the notification is deleted.
– If an absence is made inactive again by a change (e.g. of the data), a notification is generated.
– When the team leader of a user changes, existing notifications for absence approvals on the old team leader are deleted and new ones are created on the new team leader.
– If no team leader is entered for an employee, no notification is created.

**Notifications for pending activities**
This plug-in can be found in the Knowledge Base in **Plug-in: Notifications for pending activities** at www.vertec.com/kb/plug-in-notifications-aktivitaeten/.

A notification with the title of the activity will be generated if the following criteria are met:

– A responsible person is registered (this person receives the notification)
– The activity is not yet done
– The activity has a title (otherwise the notification has no text)
– An appointment is entered for the activity and the appointment is reached. For this purpose, there is a system setting in which you can define how many days before the deadline a notification will be created. The default is 0 days, i.e. no notification before the deadline.

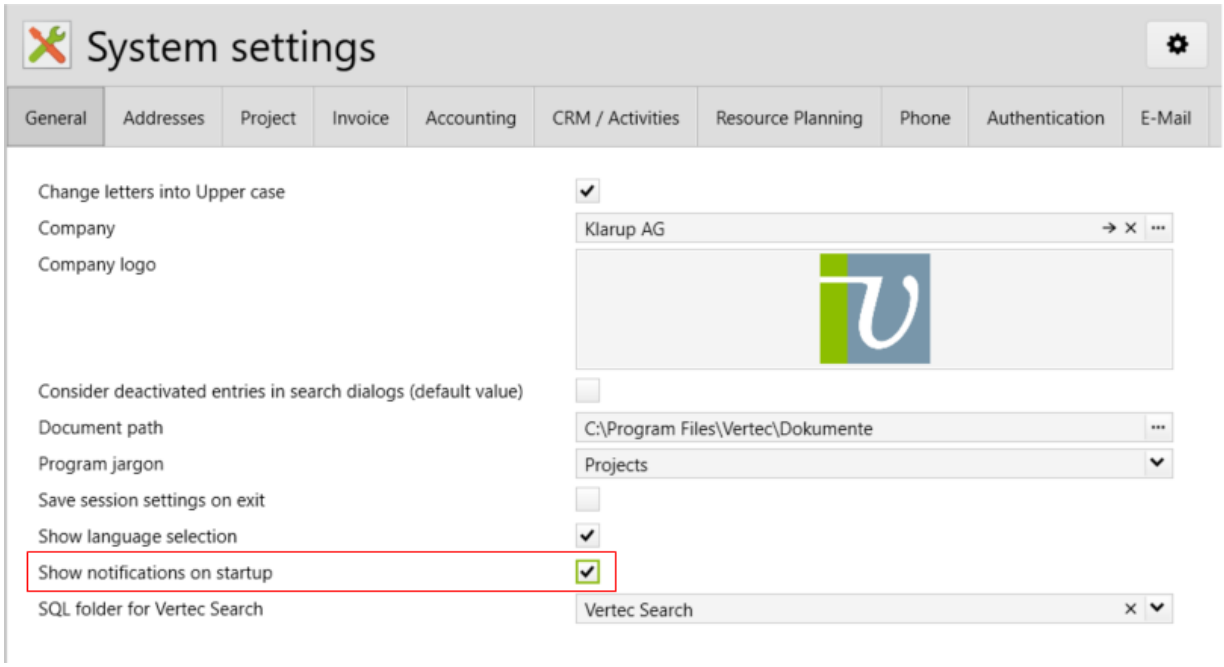
There is also a system setting that determines whether or not a notification is created for activities without an appointment. If this setting is activated, a notification is always created immediately if no appointment is entered. If not, there will be no notification for activities without appointment.

When the responsible person on the activity changes, the new responsible person receives a notification, the notification on the old responsible person is deleted.

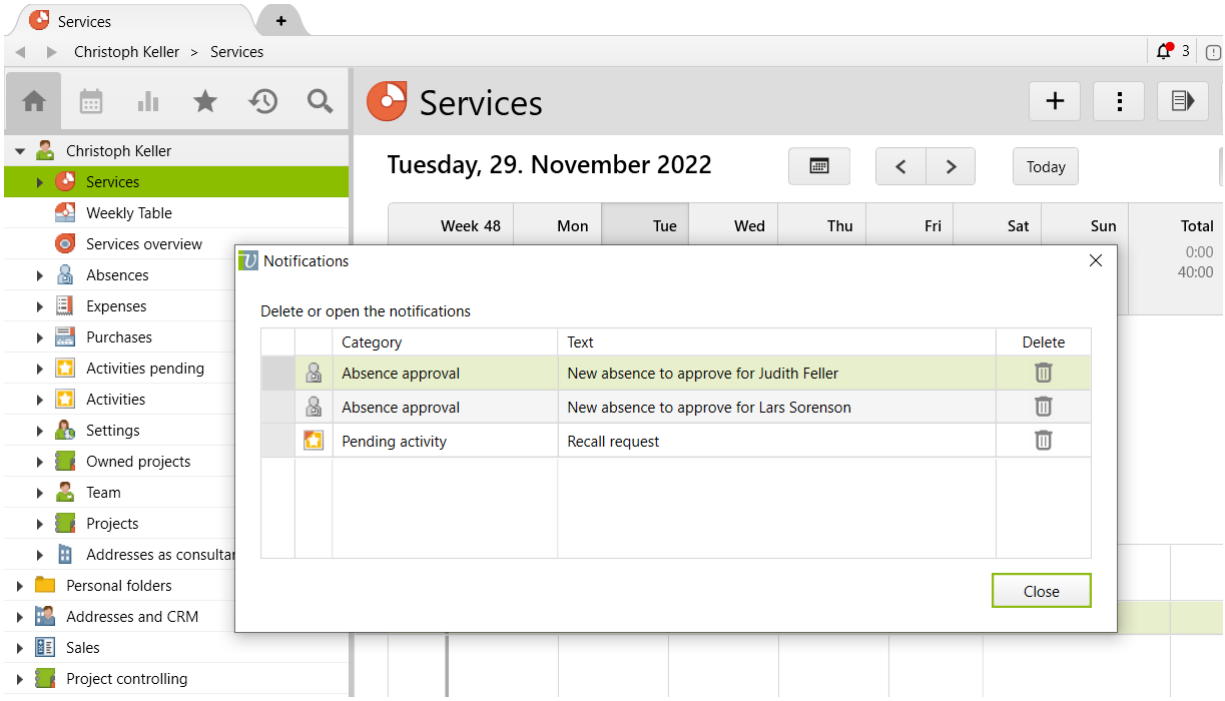
Every night it is calculated which activities are due according to the deadline from the system settings and the authorizations are created for them. This is done automatically via scheduled tasks. These must be running for the feature to work. The information about this can be found in the article [www.vertec.com/kb/geplante-aufgaben/](www.vertec.com/kb/geplante-aufgaben/).

## 4.2 Show notifications when starting Vertec

In the system settings under **General**, the option `Show notifications on startup` can be used to configure whether the Notifications dialog should be displayed automatically after login if notifications are available. The default value is `False`.




If the checkbox is checked, the following dialog appears automatically after login:




With a double click you can navigate directly to a notification or close the dialog with the `Close` button.

## 4.3 Create Notifications

Notifications are created by scripts. **Two new Python functions are available on the user** for creating, modifying and deleting notifications. The notifications are uniquely identified by the triple of user, `category` and optional `link`. If there is no write access to one of the members, the functions report an error.

- setnotification(category, text, link): adds a notification with specified category, text and optionally linked user entry. If a notification with the same category and the same linked object already exists on the user, the text will be updated.
- deletenotification(category, link): deletes the notification on the user with specified category and linked object. If no corresponding notification is found, nothing happens. No error message appears.

The notifications are `UserEntries` with the following attributes:

- `category` is a string that indicates the category (type) of the notification. This is freely selectable.
- `text` is an arbitrary text that describes the notification and usually also contains information about the linked object.
- `link` (optional): is a Vertec object (`UserEintrag`). If specified, the notification will be displayed with the corresponding icon and the string representation of the object (counterlink: `linkedNotifications`).
- `owner`: The user to whom the notification belongs (counterlink: `notifications`)

**Example**

With a double click on the first notification the activity opens. The following event script can be used to configure that the notification automatically disappears from the list when the `Finished` checkbox on the activity is ticked:

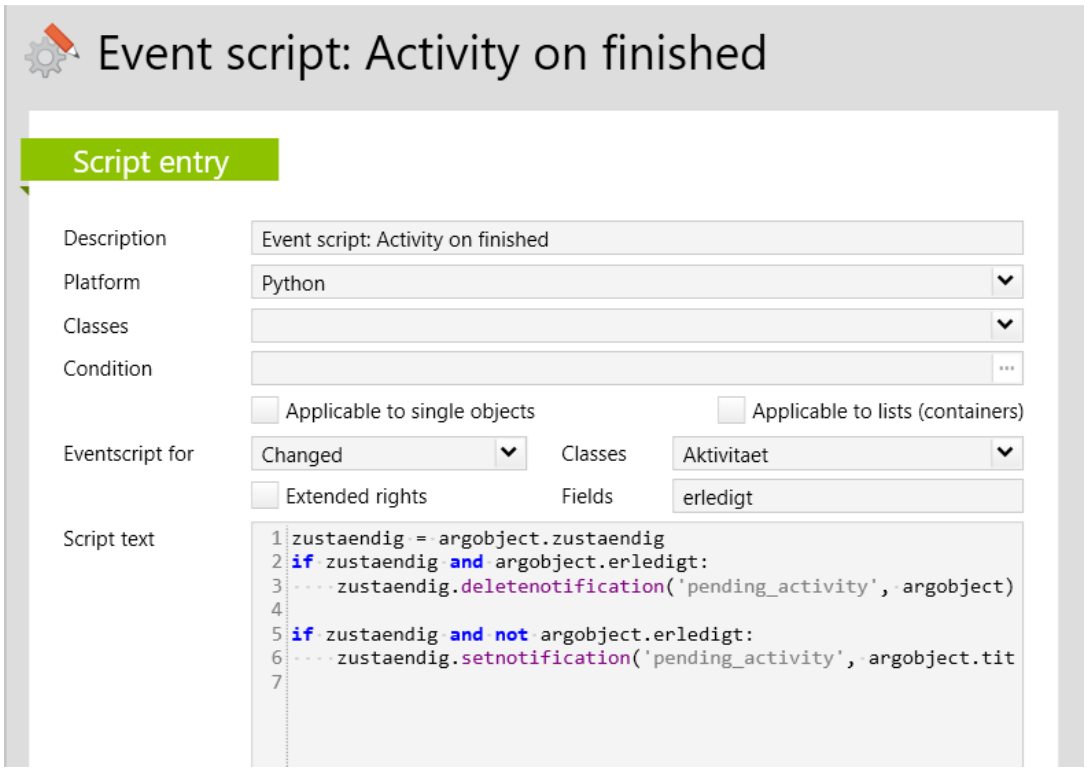



Figure 25 Example of an event script for notifications on activities

Script text:

```
#
zustaendig = argobject.zustaendig
if zustaendig and argobject.erledigt:
    zustaendig.deletenotification('pending_activity', argobject)

if zustaendig and not argobject.erledigt:
    zustaendig.setnotification('pending_activity', argobject.titel, argobject)
```

Before completion:

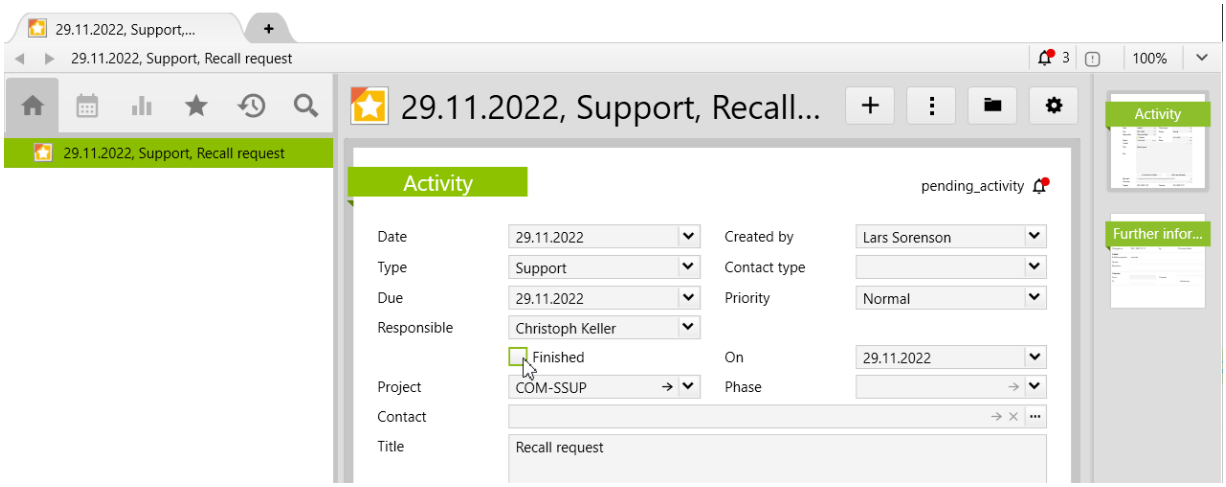



Figure 26 The notification is displayed on the activity by the bell icon

After completion:

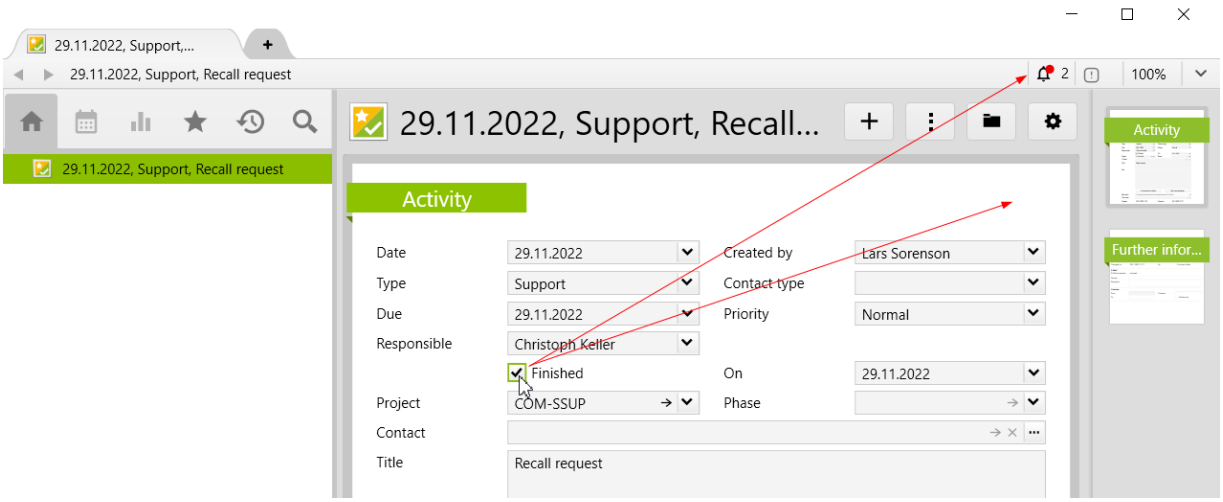



Figure 27 Checking the "Finished" box triggers the eventscript and deletes the notification automatically

## 4.4 Notification permissions

The permissions of the notifications (**Notification** Objects) are as follows:

- **Read access:** The text of notifications can be read only by the `owner` of the notification.
- **Write access:** No one has write access. Since notifications are always set via Python method, write access is not necessary.
- The `setnotification()` and `deletnotification()` methods (see 4.3) for creating and deleting notifications check the write/read access to the `bearbeiter.notifications` Member.

# 5 Outlook App

## 5.1 Assigning an opportunity

Line: Expert | Module: PSA | Version: 6.5.0.7
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

When saving an e-mail in Outlook App, an opportunity can also be assigned. The opportunities are only visible if the `Use opportunities in Outlook App` option is enabled under **System Settings > CRM / Activities**.

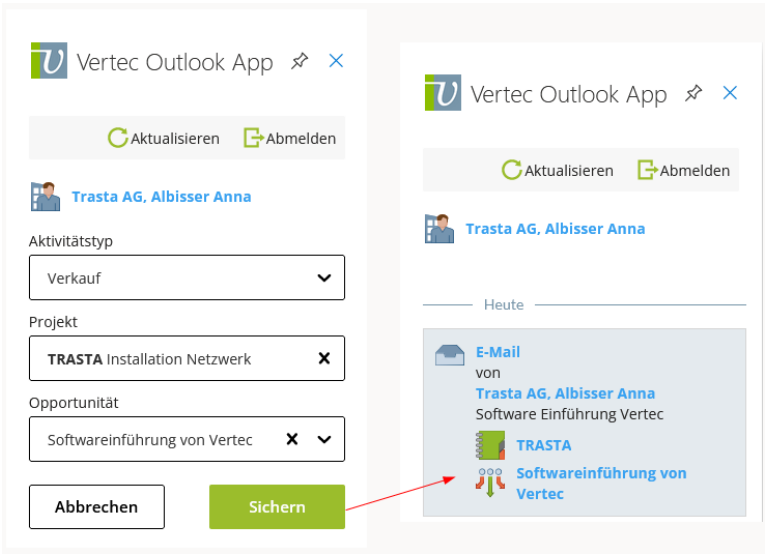



Figure 28 Assign opportunity in the Outlook app

The active opportunities of the address entry appear for selection. However, a new opportunity can also be typed in and thus created. If an e-mail has several recipients, the opportunities of all recipients are offered for selection.

## 5.2 Preselection of project and phase based on a calendar entry

Line: Standard, Expert | Module: PSA, Budget & Phases | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

If a project and a phase are mentioned in the title of a calendar entry, they can be preselected automatically when creating a new activity. For this, the option `Phase assignment in Outlook App` must be activated in the **system settings** under **CRM / Activities**.

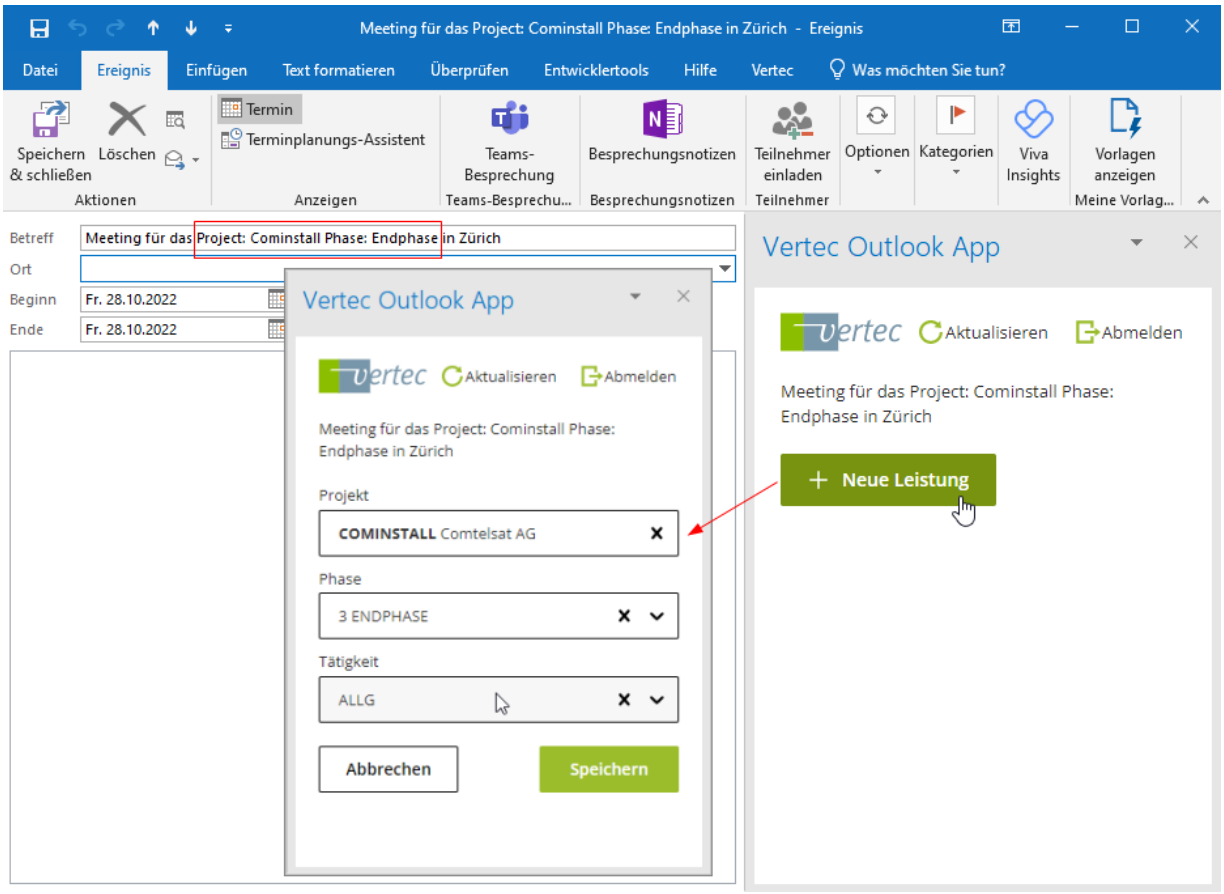
In order to recognize the phase as well as the associated project, one of the following notations must be selected in the calendar entry:

- **Only Project**:
  - Project: <ProjectCode> ...
  - Project: <ProjectCode>: ...
  - Project:<ProjectCode> ...
  - ... Project:<ProjectCode> ...
  - ... (Project:<ProjectCode>)
  - ... [Project: <ProjectCode>]...
  - ... Project: (<ProjectCode>)

– Project and Phase:

  – Project: <ProjectCode> Phase: <PhaseCode> ...
  – Project:<ProjectCode> Phase:<PhaseCode>: ...
  – (Project:<ProjectCode> Phase:<PhaseCode>) ...
  – ...[Project: <ProjectCode> Phase:<PhaseCode>] ...

  Example: Meeting for the Project: Cominstall Phase: Final Phase in Zurich:




– The following characters are supported in the project code:

  – : ; , . ' ` ´ , › ‹ " ‟ ″ „ » « ( ) [ ] { }

## 5.3 Address entered in Outlook is displayed in the Outlook App

Line: Standard, Expert | Module: PSA | Apps: Outlook App
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

When composing new emails, it was previously possible to enter addresses in the Outlook App and transfer them to the e-mail. Now the reverse is also possible: when a new e-mail is composed in Outlook and an address is entered, it is transferred to the Outlook App. If the address is already known, the corresponding activity history is displayed.

## 5.4 Entering a new address when sending e-mails

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

When composing a new e-mail, all recipient addresses are displayed in the Outlook App using arrow keys. E-mail addresses that are not yet saved in Vertec can be created directly in Vertec using the `New Address` button.

## 5.5 Extensions in the project search

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

The project search can be extended by activating the checkbox `Advanced project search (Outlook App)` under **System Settings > CRM/Activities**. This way, in addition to `code`, the fields `description` and `concerning` are also searched.

## 5.6 Assigning Phases

Line: Standard, Expert | Module: Budget & Phases | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

When creating an activity based on an e-mail, from this version onwards a phase can also be selected in the Outlook App in addition to the project. As soon as a project has been selected, the corresponding phases and subphases appear in the next selection box for selection, on which the logged-in user may record. For this purpose, the option `Phase assignment in Outlook App` is available in the **system settings** under **CRM / Activities**.

## 5.7 Text for checkbox adjusted

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

In the Outlook App, it is not always possible to automatically detect whether an e-mail is outgoing or incoming. In such a situation a checkbox is displayed. This was previously labeled `Outgoing`. Now it is called `Save as outgoing e-mail`.

## 5.8 Entering multiple services for serial appointments

Line: Standard, Expert | Module: PSA | Version: 6.5.0.16
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

In the Windows Edition of the Outlook App, it is also possible to enter a service for each appointment individually in the case of serial appointments. This was previously only possible in the Web Edition.

## 5.9 Optional startup in Windows edition

Line: Standard, Expert | Module: PSA | Version: 6.5.0.16
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

The state of the Vertec panel in the Windows edition is taken from the last Outlook session. If the Vertec panel was open, it is displayed, otherwise it is hidden. Each time the panel is changed (shown or hidden), the new state is written to a local settings file. If no value is found in the settings file at startup, the panel will be shown.

## 5.10 Alphabetical sorting of lists

Line: Standard, Expert | Module: PSA | Version: 6.5.0.21
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

Objects are sorted by the display text and not by the Internal Id, so that they are sorted alphabetically in the lists.

## 5.11 Improvement for Outlook App Auto Update

Line: Standard, Expert | Module: PSA | Version: 6.5.0.3
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

If an update of the Outlook App is necessary, a corresponding note is displayed in the foreground. With a button the update can be started directly. To avoid error situations, it is no longer possible to run the Outlook App with an outdated version.

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

In the Windows edition of Outlook, updates are automatically notified even if Outlook is not closed.

## 5.12 New system setting for the Web Edition

Line: Standard, Expert | Module: PSA | Version: 6.5.0.22
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

From this version on, links from the Web Edition of the Outlook App can also be handled in the Cloud or Desktop App. For this purpose, the setting Handling of Vertec links (Out-look App Web) is available in the system settings under CRM/Activities, where you can choose between the options:

- **Open links in Web App**, and
- **Open links in Cloud/Desktop App**

## 5.13 New default value in the system setting

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Outlook App

The default value of the system setting `Default value for "Save on send" (Outlook App)` has been changed from **True** to **False**, so that the user has to explicitly choose whether to save emails automatically.

## 5.14 Webview2 installations without administrator rights

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: On-Premises | Apps: Outlook App

An installation with the Outlook App Installer no longer needs administrator rights for the installation of Webview2 (Web Viewer for the Windows Edition).

# 6 Opportunities

## 6.1 Adjusting link types and string representation

Line: Expert | Module: PSA | Version: 6.5.0.3
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

When linking from a company to opportunities, the opportunities of contacts of the company are now also displayed. The string representation also contains the description.

## 6.2 Linking activity with opportunity

Line: Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

When running reports on an opportunity, the activity is directly linked to the opportunity and is thus visible on the opportunity in the tree.

# 7 Invoicing and Billing

## 7.1 EPC-QR-Code Support

Line: Standard, Expert | Module: PSA, Third Party Costs | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In addition to Swiss QR codes, European EPC QR codes on accounts payable are supported as of this version. These can also be evaluated when reading in creditor documents. To be able to use the EPC QR code, the currency on the project of the invoice must be set to `EUR`.

---

**Further Information** For more information on EPC QR codes, see the article **Setting up ESR/QR Invoicing in Vertec** in the Online Knowledge Base at: www.vertec.com/kb/qr-esr.

---

## 7.2 Authorization logic changed

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The authorization logic for creating invoices on projects has been changed. Now only the write permission on project.invoices (`projekt.rechnungen`) is checked and not whether the user has project manager rights on the project.

## 7.3 Importing creditor documents

Line: Standard, Expert | Module: PSA, Third Party Costs | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The following improvements have been made:

- When importing a document **without** a QR code, the existing information on the creditor is retained.
- When importing a document **with** QR code, if one of the properties of the creditor to be set is read-only (for example, if the creditor is already posted), import is possible only if the two values match.
- When importing a receipt **with** QR code, if the display is charged (if the number cannot be written), import is possible. However, only `xWertIntFWBrutto` is set.
  If incorrect or incompatible data is found when importing a document **with** QR code (for example, if the currency is not found), the import is canceled and the document screen is not changed.

---

**Further Information** For more information about importing creditor documents, see the article **Posting creditors with Vertec** in the Online Knowledge Base at: www.vertec.com/kb/kreditoren.

---

## 7.4 Importing creditor documents with multiple QR codes

Line: Expert | Module: Fremdkosten | Version: 6.5.0.18
Mode of Operation: On-Premises | Apps: Full-featured

Even a creditor document that contains a QR code **not** supported by Vertec can be imported. The document image is saved but no creditor data is written.

If the creditor document contains multiple QR codes, the first invoice QR code is processed and the irrelevant codes are ignored.

## 7.5 New Python function for processing creditor documents

Line: Expert | Module: PSA | Version: 6.5.0.7
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

An invoice can also be processed with QR code using a script. The Python function `readinvoicedocument()` accepts **PDF**, **JPG** and **PNG** formats of creditor documents and processes the QR code contained in them, which it returns as a data object.

| Property | Description |
|---|---|
| Name | Name of payee |
| Address | Address information of the payee |
| Zip | Postal code of the payee |
| City | City of the payee |
| Country | Country of the payee |
| Amount | Gross amount |
| Currency | Currency |
| Account | Account |
| Reference | Reference Number |
| *DocumentNumber | Document Number |
| *DocumentDate | Date on Document |
| *DueDate | Due Date |

* Optional data: will only be output if the QR code contains additional information.

**Further Information** For more information and a sample script, see the **Vertec Python Functions** article in the Online Knowledge Base at: www.vertec.com/kb/pythonfunktionen.

## 7.6 Removed menu item Charge Internal Services

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The menu item `Charge Internal Services, Expenses and Purchases` on projects has been removed. However, the functionality has been rebuilt and is available for download as **Plug-in: Charge Internal Entries** in the Knowledge Base.

You can download it here: www.vertec.com/kb/plug-in-interne-eintraege-verrechnen/.

# 8 Documents and Reports

## 8.1 Removal of the legacy Excel reports

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

As of Vertec 6.6, we no longer deliver legacy Excel reports. The reports are as follows:

– Work hours per user (Excel) - this is now supplied as an Office report, see 8.8.
– List of absences per user (Excel) - this is replaced by a list **Overview Absences** in the HR folder by the **Plug-in: Overview Absences** (www.vertec.com/kb/plug-in-uebersicht-abwesenheiten/).
– Weekly overview (Excel)
– Project-controlling (Excel), see Chapter 1.5
– Resource table (Excel), see Chapter 2.18
– Resource overview (Excel), see Chapter 2.18
– Resource plan (Excel), see Chapter 2.18

**Backwards Compatibility**
All Resource Reports are deleted even in existing installations. The other reports are left in the report templates, but disabled.

With Vertec 6.7, Excel reports will no longer be available even for existing installations.

## 8.2 New menu of the Report templates folder

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the **Settings** root folder under **Report Templates**, only **Office Reports** (formerly "Extended Office reports") can be created in the New menu.

## 8.3 Simplifications in the coding of Office reports

Line: Standard, Expert | Module: PSA | Version: 6.5.0.20
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python code in the Office reports (previously: Extended Office reports, see 1.3) has received various simplifications: OCL field calculations without declaration directly in the report, an initialization method on tables, support of the report language in the code, an automatic frame table and much more.

For the **new structure** of Office reports, see the article **Python Code for Office reports** in the Online Knowledge Base at https://www.vertec.com/ch/kb/python-code.

For an overview of all **changes**, see the article **Changes in Python Code for Office reports as of Version 6.5.0.20** in the Online Knowledge Base at https://www.vertec.com/ch/kb/eob-umstellungen.

**Backwards Compatibility**
The changes are fully backward compatible. All previous reports continue to run unchanged.

## 8.4 Upgrading built-in Office reports

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The standard Office reports (previously: Extended Office reports, see 1.3) have been adapted to the new features (see 8.3).

In order to deliver these, we had to develop a system so that no data would be lost in the "customer customized the report/code/both" scenario.

| Scenario | Template unchanged | Template adapted |
|---|---|---|
| Script unchanged | OK | Problem Type A |
| Script angepasst | Problem Type B | OK |

- Problem Type A: If new built-in script is no longer compatible with existing uploaded template (e.g. a field does not exist anymore) the Office report runs into an error or behaves undesirably differently after upgrade.
- Problem Type B: If a new built-in template is no longer compatible with the customer customized script (e.g. a field no longer exists) the Office report runs into an error after the upgrade.

Both problems have to do with the fact that in Office reports it is possible to customize template and script independently, and in particular to customize only one part, while for the other our standard is used.

To avoid such upgrade problems, the Office report system has been adapted so that when the customer makes adjustments via the user interface, both parts are always custom-fixed, even if only one part is adjusted.

This means that when a specific document (template) is uploaded, we also copy the script text. This is therefore also considered as customized.

Resetting is now only supported for the entire report, i.e. for templates and code together.

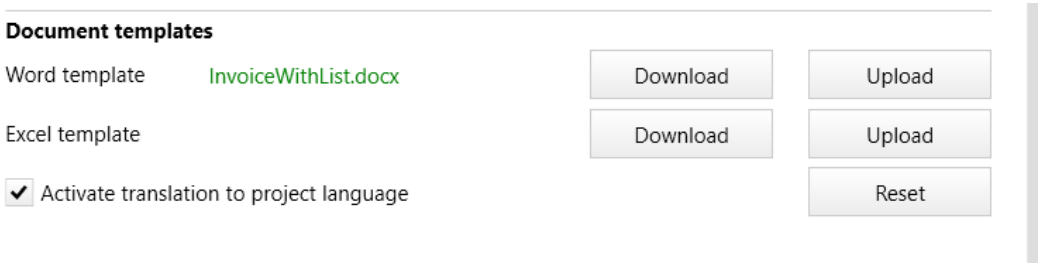



Figure 29 In the office report registration there is only one reset button

- In the templates in the Office report detail view there is only one common "Reset" button if it is a built-in report (Internal name in the report definition is set)
- The common "Reset" button is only enabled if one of the templates or the script has been customized.
- For custom (self-created) Office reports (`Internal Name` in report definition is not set), "Delete" buttons for Excel and Word templates are displayed as before.
- When pressing "Reset", a warning dialog appears: "Do you want to reset the report templates and report code to their default values?"
  - After resetting, both templates and the script text are at their default (built-in) value.
  - On the "Report definition" page, the "Customization" view is empty.
- If there is a customized script on the "Report definition" page, it cannot be deleted completely. A message dialog will appear "To reset the customized report definition, please use the "Reset" button on the main report page".

### Backwards Compatibility

To ensure that no data is lost during the update, we include the previous versions of the scripts and templates.

- If the customer has customized one of the parts, the previous version is used as another part.
- If the customer has customized both parts, the new versions are inserted "behind" them, the customized parts remain as they are.

Customers updating from an early Vertec version 6.4 or older who receive an error message of the type `qr_adresstext is missing` in the invoice templates, please refer to the **Adjust Code** section in the KB article at www.vertec.com/kb/rechnungsvorlagen-fuer-qr-code-bereit-machen/.

## 8.5 Python stub file for reporting modules

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

With the changes described in chapter 8.3 we now also provide a Python stub file for Reporting Module. This is called **reporting.py** and is stored like the other stub files in the subfolder PythonStubs in the Vertec installation directory.

This must be imported for use: `from reporting import *`

The methods that need to be declared in the office report code itself (e.g. `def initialize_row(context, row)`) are included in the stub file, but only as documentation help. This is a bit confusing, but the doc is so complete.

For the **context** variable to work, the following type annotation must be used in Python (example):

```
def calc_table(context):
    # type: (Context) -> Table
```

**More information** For more information about the Python stub file for Reporting Module, see the KB article at www.vertec.com/kb/pythonstubfiles/.

## 8.6 Display individual advance on invoice with text

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

If an invoice contains only ONE advance, the text of the advance appears on the first page instead of "Advances".

In the case of the Invoice with services list, the list of advances on the detail page is also omitted in this case.

This concerns the following reports:

– Invoice with services list (**InvoiceWithList**)
– Invoice with user sums (**InvoiceWithUserSums**)
– Calculation with phase sums (**InvoiceWithPhaseSums**)

**Backwards Compatibility**
The update follows the same scheme as explained in chapter 8.4. The new feature is only available if the built-in reports are unchanged or after clicking `Reset` (but the customized template or code will be lost).

## 8.7 Offer as Office Report

Line: Standard, Expert | Module: PSA | Version: 6.5.0.23
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Word report "Offer" (Offer2.dotx) has been rewritten as an Office Report.

**German Specific Point:** The system uses the term **offer** (High German: **Angebot**) instead of **offer** (Swiss German: **Offerte**), if German (High German) is selected as the presentation language. **Note:** in English the words are the same.

The term **Costs** has been replaced by **Expenses** and **Purchases**. Previously, "costs" included expenses and purchases, but we don't do that anywhere else and in Vertec, costs are something else.

If there is an overwritten plan value on the phase (green), a row with this plan value is simply displayed. If not, the corresponding project entry phase links are displayed, one per line.

Only phases that have any plan value (`xPlanValue` or `ProjectEntryPhaseLinks`) are listed.

**Backwards Compatibility**
- For new installations, there is only the new Office report.
- For existing installations, the new Office report is registered as **Inactive**, the previous Offer report remains **Active**. If you want to use the new Office report, set the previous report to **Inactive** and the new one to **Active**.

## 8.8 Work hours per user as office report

Line: Standard, Expert | Module: PSA | Version: 6.5.0.23
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Excel report `Work hours per user` has been rewritten as an Office report.

**Backwards Compatibility**
In the report templates, the old report is automatically set to **Inactive** and the new report `Work hours per user` is set to **Active**.

---

**Further information:** For more information about the Work hours per user report, see the **Reports for Project Workers** article in the Online Knowledge Base at https://www.vertec.com/kb/reports_projektbearbeiter.

---

## 8.9 Monthly overview with target times does not show empty lines

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In case of lump sum invoices or phases without services, the monetary value is attributed to the project manager and therefore a `ServiceSum`-Object is created for it.

This resulted in blank lines appearing in the Monthly Overview with Planned Times report, since it only deals with expenses that are omitted in the case of flat rates.

These empty lines have been removed. Only projects on which effort has actually been spent will appear in the list.

**Backwards Compatibility**
The update follows the same scheme as explained in chapter 8.4. The new feature is only available if the built-in reports are unchanged or after clicking `Reset` (but the customized template or code will be lost).

## 8.10 Treatment of German language variants

Line: Standard, Expert | Module: PSA | Version: 6.5.0.12
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The current interface language is also considered when running reports. This means that for Office Reports where `Activate translation to project language` is set, the respective correct translation is applied for `DE` projects, i.e. **German (Deutschland)**, if that German interface language variant is set.

---

Further information see the article **Multilingualism with Vertec** in the Online Knowledge Base at https://www.vertec.com/kb/mehrsprachigkeit-mit-vertec.

---

## 8.11 System setting for a Company Logo

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the **system settings General**, the logo of the company can be stored in order to use it in reports by default. By right-clicking in the field, the image can be loaded, opened, saved and also deleted again.

PropertyName: `CompanyLogo`. BlobProperty.

## 8.12 Support for images in context variables

Line: Standard, Expert | Module: PSA | Version: 6.5.0.12
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The `set_image(name, value)` method can be used to set an image on the context object in Office reports:

```
logo = vtcapp.getpropertyvalue('CompanyLogo')
context.set_image('logo', logo)
```

This can be output as a normal context expression and also in headers and footers of Word reports.

## 8.13 Separate e-mail content from activity

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Currently, e-mails are stored in the `Content` blob field directly on the activity.

This causes the e-mail content to be loaded as part of the activity when displaying it. For e-mails with large attachments, this can lead to high memory consumption and poor performance.

Therefore, as of version 6.6, the e-mail contents are stored separately and linked to the activity. Access to this "document" content is then only made via the interface when required.

### Model

There is a new class called `DocumentData` which stores these document contents. It inherits from `BusinessClassesRoot` and has a member `Data` of type Blob.

The association to the activity is DocumentData - OwningEntry (1:1).

On the activity (Aktivitaet) there are the following new features:

| | |
|---|---|
| **documentData** | Link to the associated document object. |
| **documentFullName** | String. Name of the file incl. file extension (e.g. `MyExcelFile.xlsx`). |
| **documentSize** | Integer. Size of the document in bytes. |
| **documentModifiedOn** | DateTime. Contains the date when the document was created or last modified. |
| **documentName** | String. Calculated attribute (derived). Contains the name of the file without file extension (e.g. `MyExcelFile`). |
| **documentExtension** | String. Calculated attribute (derived). Contains the file extension of the file (e.g. `xlsx`). |

As before, one document can be stored per activity. If the activity is deleted, the linked DocumentData object is also deleted.

Previous features:

– **Content**: The previous blob field is newly derived (calculated at runtime) and loads the content from the linked `DocumentData` object.
– **EmailAttachmentNames:** String. The previous derived attribute is now persistent and contains the list of attachments, separated by commas.
– **EmailHtml:** String. This previous derived attribute remains. It is important that this attribute is no longer used, e.g. in a list, otherwise the entire content will be reloaded.

**Permissions**
The access rights to properties of a DocumentData object depend on the access rights to the associated activity:

– Read access to the `Data` property of DocumentData objects is coupled to the read permission on `Aktivitaet.Content`.
– If the DocumentData object is not linked to an activity, the read access is blocked, only an administrator can access the members of the DocumentData object.
– Restrictions on the authorization on the member `Aktivitaet.Content` are inherited by the following members:
  – Aktivitaet.EmailAttachmentNames
  – Aktivitaet.EmailHtml
  – Aktivitaet.EmailOutgoing
  – Aktivitaet.EmailRecipients
  – Aktivitaet.EmailSender
  – DocumentData.Data

**Backwards Compatibility**
Existing documents on activities are automatically converted during the update.

This is not noticeable on the user interface.

For e-mails in the activities, no concrete data can be captured for `DocumentFullName` and `DocumentModifiedOn`. Therefore, the following values are set when converting and creating an activity in the Outlook App:

– DocumentFullName: `Email.eml`
– DocumentModifiedOn: `CreationDateTime` of the activity

# 9 Controlling / BI

## 9.1 BI queries return the object ids of dimension objects

Line: Standard, Expert | Module: Business Intelligence | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The BI API `getdata` endpoint now provides the ID of the dimension objects. For each dimension value there is another property with the same name and suffix `_Id`, which contains the object id of the dimension value as value.

## 9.2 OCL queries on dimension values

Line: Standard, Expert | Module: Business Intelligence | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the results of a BI `getdata` query, one or more OCL expressions can now (optionally) be specified per queried dimension. For this purpose, an alias with an OCL expression can be specified for each dimension parameter (e.g. `dimension0_Projekttyp=typ.bezeichnung`). This causes a new field `Projekttyp` to appear in the response, which contains the name of the project type of the project dimension value.

Several alias parameters can be specified per dimension. If the parameter used corresponds to a name that is already in use (as measure names, class names or their translations, or alias of another dimension), an error is reported.

---

Further Information: For more information, see the **BI API** article in the Online Knowledge Base at https://www.vertec.com/kb/bi-api.

---

# 10 Service Recording

## 10.1 New option "Open image"

Line: Standard, Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

For posted expenses and disbursements, the context menu shows next to the option `Save image...` also an `Open image...` option.

## 10.2 Grouping by phases in Services overview on project

Line: Standard, Expert | Module: Budget & Phases | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The **Services overview** on a project can be grouped according to various assignments, including "User, Phase" and "Phase, User". Grouping by "Phase" only is now also available.

---

**Further Information** For more information on grouping, see the Services overview article in the Online Knowledge Base at https://www.vertec.com/kb/uebersicht-leistungen.

---

# 11 Users

## 11.1 User-phase-Links with Data Interval

Line: Expert | Module: Budget & Phases | Version: 6.6
Method of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

User-phase-links now have a date interval, analogous to the link of service types to phases. For more information, see the chapter 2.15.

## 11.2 Member Active for User-links

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

There is a new checkbox `Active` for the **user link** to projects and phases:

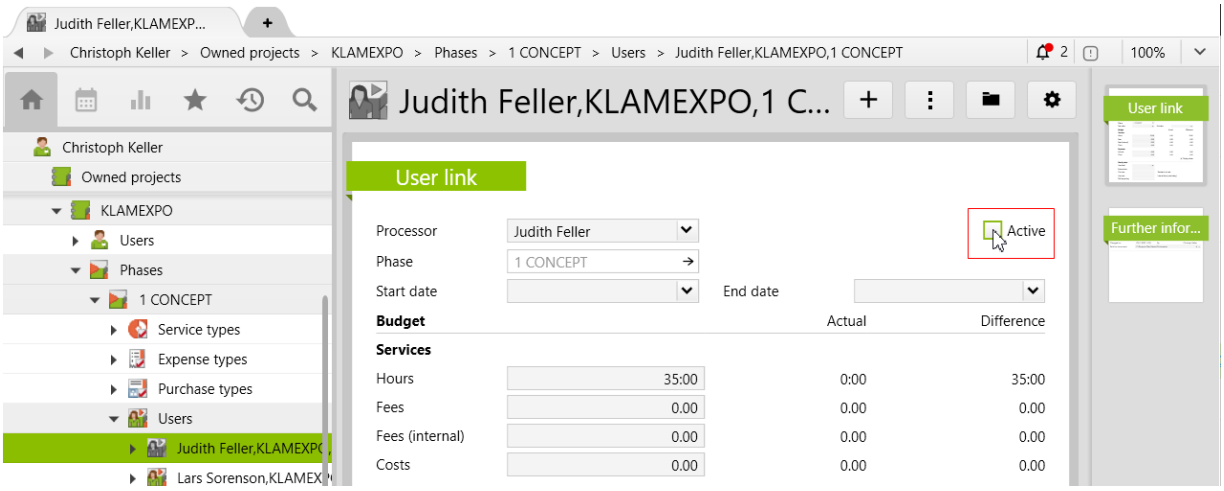



Figure 30 Active checkbox for the users

User links to project can be set to **inactive** and are thus displayed in gray. This makes it possible to track who was previously assigned to the Project.

If the option **Users must be assigned to a project** is activated under System settings > Project, only the projects to which the user is assigned as active user appear for selection during service entry.

The default value is `True`.

In connection with this, the **standard list settings** of the User-link lists and User-phase-link lists have been extended by a column with an active checkbox:

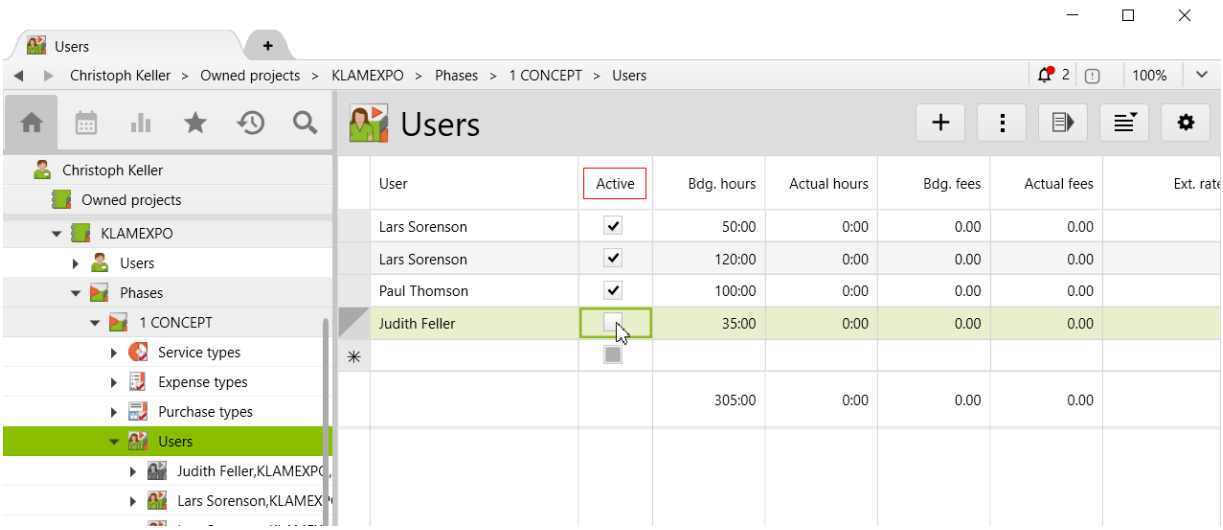



Figure 31 Active checkbox in the Lists of User-phase-links

## 11.3 Hiding group absences for Users

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The list of absences is displayed on employees as a two-part list: In the upper part, the absences of groups of which the user is a member, and in the lower part, the list of absences of the user himself.

In order to display only the simple list of absences of the user himself, without the absences from the groups, a separate container class AbsencesContainer (`AbwesenheitenContainer`) was created. The `AbwesenheitenContainer` contains the previous, two-part view and is entered on the existing link type **User - Absences**. So nothing changes in the view.

If you want to hide the upper group part, you can enter `WrapperLinkContainer` as container class on the link type. This way only the absences of the user will be displayed:

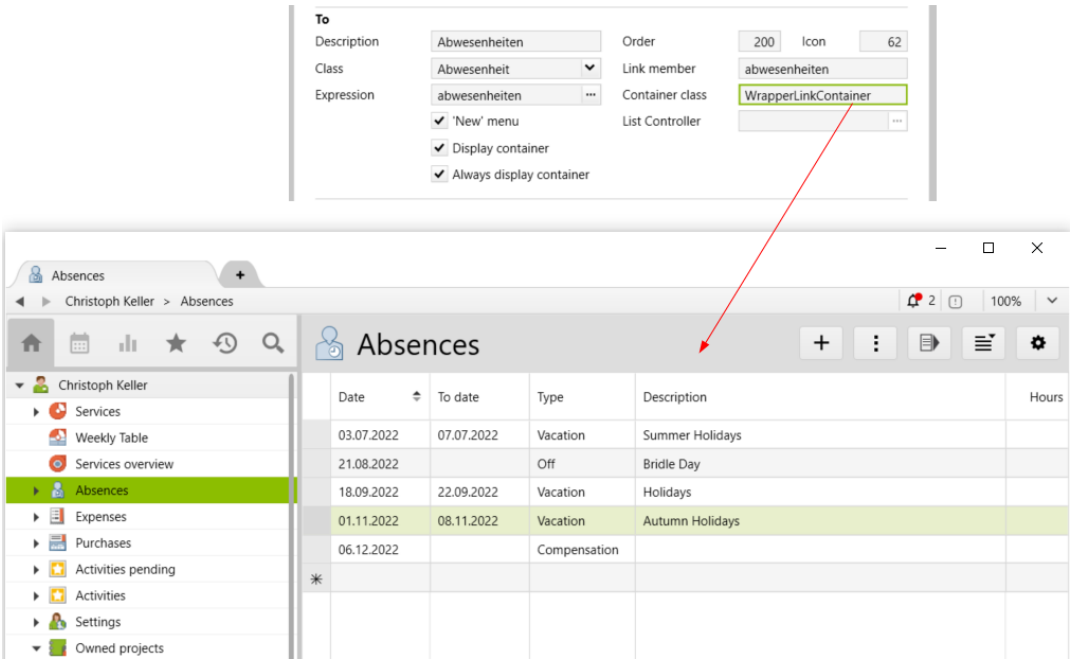



Figure 32 Enter the Container class WrapperLinkContainer

# 12 Customizing

## 12.1 Warning message when resetting in the list settings

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the list settings, clicking the `Save as default` and `Reset` buttons displays a warning dialog that can be used to confirm or cancel the action:

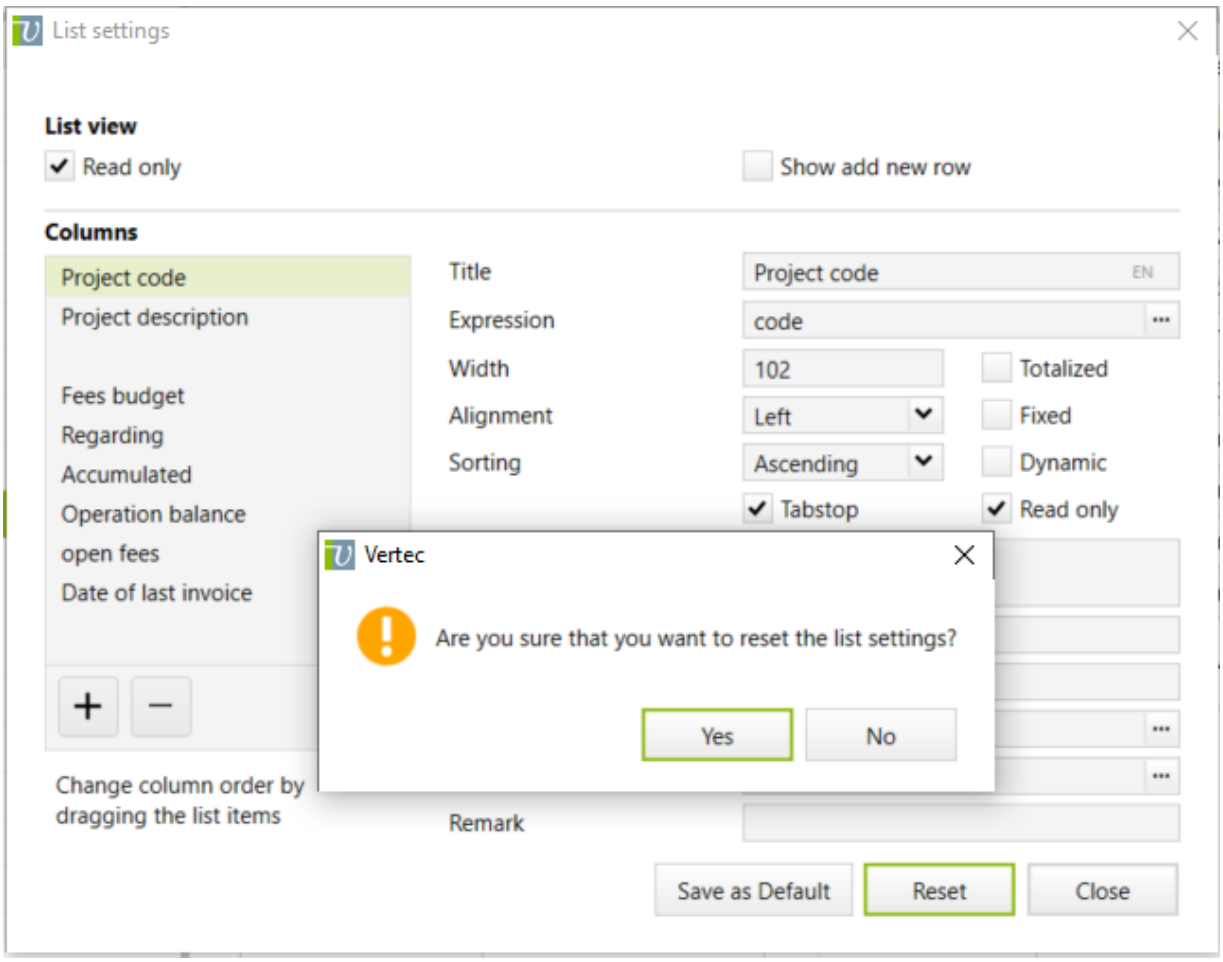



Figure 33 Warning dialog in the class settings

## 12.2 More detailed warning message in case of "invalid" OCL

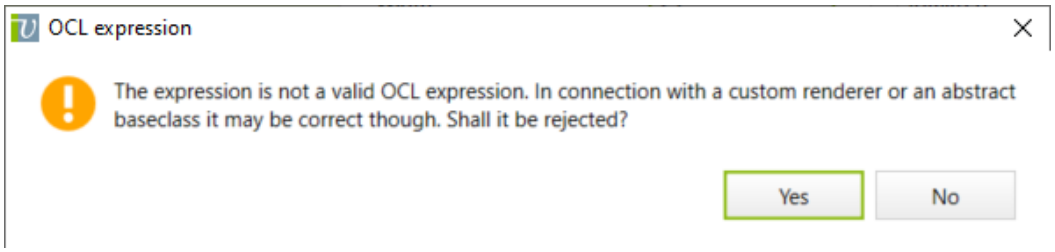
Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

If a column expression is entered in the list settings and it is not valid on the business class of the list as OCL, a warning message was displayed with the wording:

`The OCL expression entered is invalid. Should it be discarded?`

In connection with Custom Renderers (see chapter 3) it can happen that these expressions are valid anyway.

Therefore, the following message is now displayed:




## 12.3 New Customer Classes

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In addition to the existing 30 custom classes (customclass0 to customclass29), the following new additional classes are supported:

– 20 Customer Classes for individual customizing: **ZusatzKlasse30** to **ZusatzKlasse49**.
– 50 Customer Classes for Plug-ins: **PluginCustomClass00** to **PluginCustomClass49**. These additional classes are only used internally by Vertec and may not be used for individual customizing.

## 12.4 Additional OCL Call Operators

Line: Expert | Module: PSA | Version: 6.5.0.7
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

There are the following two new OCL call operators:

– `callCurrString` for the Python function `func(obj, string) -> float`
– `callCurrDateDateString` for the Python function `func(obj, date, date, string) -> float`

## 12.5 Keys in expressions for permissions

Line: Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Keys and other OCL operators like tags or getlinks in permissions:

The authorization check has been adapted so that the OCL in authorizations can also be checked if there is no authorization on the member itself that is being checked. This could lead to an endless loop in certain constellations. The expression is now evaluated with **extended authorizations** (in the SystemContext).

## 12.6 Renaming additional field type Unicode Text to Text

Line: Expert | Module: PSA | Version: 6.5.0.7
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The additional field type `Unicode Text` introduced with Vertec version 6.5 is now called `Text`, while the additional field type previously called `Text` is now called `ANSI Text`.

---

**Further Information** For more information about field types, see the Additional fields (Zusatzfelder) article in the Online Knowledge Base at https://www.vertec.com/kb/zusatzfelder.

---

## 12.7 Filter property replaces FileMask and FileMaskName of the PathBox

Line: Expert | Module: PSA | Version: 6.5.0.20
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The new property only supports the new format for the file selection filter, which is already used by the Python function `requestfilefromclient`. Existing applications that use the old properties will no longer run with the new version and must be adapted.

---

**Further Information** For more information about the new feature, see the PathBox article in the Onli-ne Knowledge Base at https://www.vertec.com/kb/pathbox.

---

## 12.8 Support of messages for OCL requirements for plug-ins

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Plug-ins now support an optional message argument in the OCL requirements. This makes it possible to display meaningful messages to the user if a requirement is not fulfilled during import.

**Example**

```
<ocl-requirement message="CustomlClass26 is already in use.">
  ClassSettings.allInstances-&gt;select(klasse='ZusatzKlasse26'-&gt;size = 0
</ocl-requirement>
```

A translation of the text does not take place. The message is output as it is stored.

**Backwards Compatibility**
OCL Requirements without message still work the same and show the OCL Expression in their error message.

# 13 Settings

## 13.1 Expansion of the Color Palette

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured Apps, Outlook App

The Vertec color palette has been extended by the following 19 colors:

```
clMediumlightGreen, clDarkKhaki, clSemidarkKhaki, clKhaki, clLightKhaki,
clSemidarkOrange, clSemidarkYellow, clSemidarkOrangered, clSemidarkMaroon,
clSemidarkRed, clSemidarkFuchsia, clSemidarkPurple, clSuperdarkGrey clSemidarkGrey,
clSemidarkNavy, clSemidarkBlue, clSemidarkAqua, clSemidarkTeal, clSemidarkOlive
```

The existing colors have also been slightly adjusted. The following color palette shows all colors from Vertec 6.6:

| | | |
|---|---|---|
| clDarkGreen | | clDarkKhaki |
| clGreen | clDarkLime | clSemidarkKhaki |
| clMediumlightGreen | clLime | clKhaki |
| clLightGreen | clLightLime | clLightKhaki |
| clDarkOrangered | clDarkYellow | clDarkOrange |
| clSemidarkOrangered | clSemidarkYellow | clSemidarkOrange |
| clOrangered | clYellow | clOrange |
| clLightOrangered | clLightYellow | clLightOrange |
| clDarkMaroon | clDarkRed | clDarkFuchsia |
| clSemidarkMaroon | clSemidarkRed | clSemidarkFuchsia |
| clMaroon | clRed | clFuchsia |
| clLightMaroon | clLightRed | clLightFuchsia |
| clDarkNavy | clSuperdarkGrey | clDarkPurple |
| clSemidarkBlue | clDarkGrey | clSemidarkPurple |
| clBlue | clSemidarkGrey | clPurple |
| clLightBlue | clGrey | clLightPurple |
| | clLightGrey | |
| clDarkAqua | clDarkTeal | clDarkOlive |
| clSemidarkAqua | clSemidarkTeal | clSemidarkOlive |
| clAqua | clTeal | clOlive |
| clLightAqua | clLightTeal | clLightOlive |

The reason for this were the bar segments in the Resource utilization charts (see 2.2). The colors can also be set in self-written utilization dimensions. An example of this can be found in chapter 2.11.

---

**Further Information** For all information about colors in Vertec, see the **List Settings** article in the Online Knowledge Base at https://www.vertec.com/kb/farbpalette.

---

## 13.2 Selection of additional genders on contacts and persons

Line: Standard, Expert | Module: PSA | Version: 6.6

Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured Apps, Outlook App

In addition to the options `Female` and `Male`, additional genders can now also be created or selected. For this purpose, the fields **Additional genders** and **Default gender** are available in the system settings in the Addresses section.

Under **Additional genders**, additional genders can be created using comma separation, which can be selected as the default gender and when creating contacts and persons. The entered additional genders are saved as native terms and, if necessary, automatically translated into other languages.

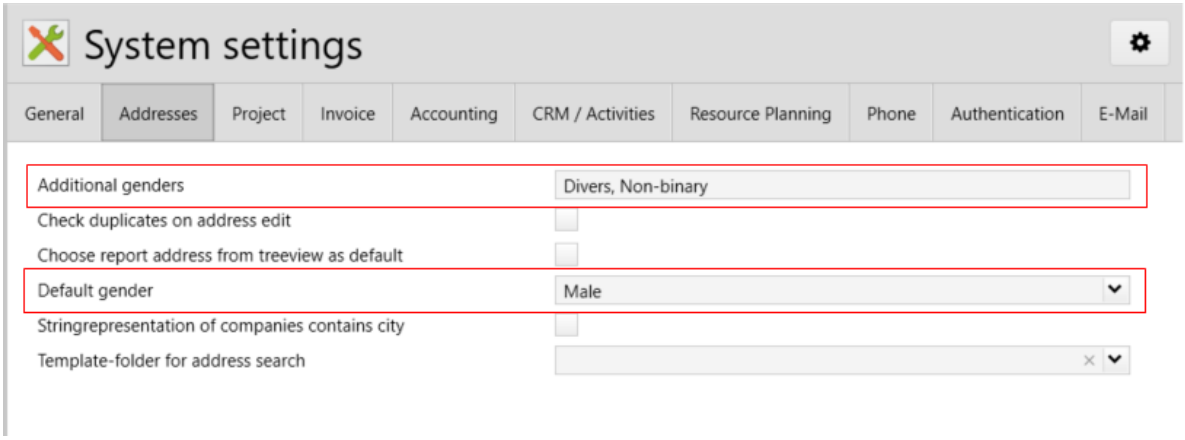



Figure 34 Addresses section in the system settings

As an example, the new gender `Divers` is supplied, which outputs the neutral letter salutation "Dear". If additional genders are created, the address layout must be adapted accordingly. All information about the address layouts can be found in the article **Address Layouts** in the Online Knowledge Base at https://www.vertec.com/kb/adresslayout/.

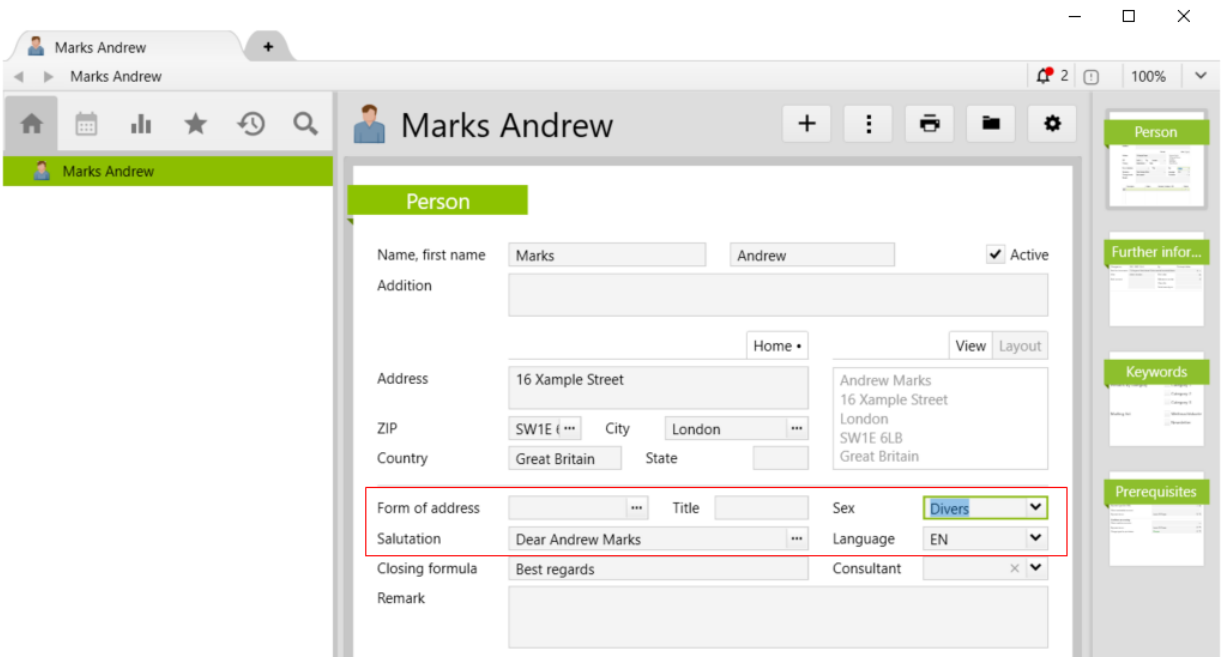


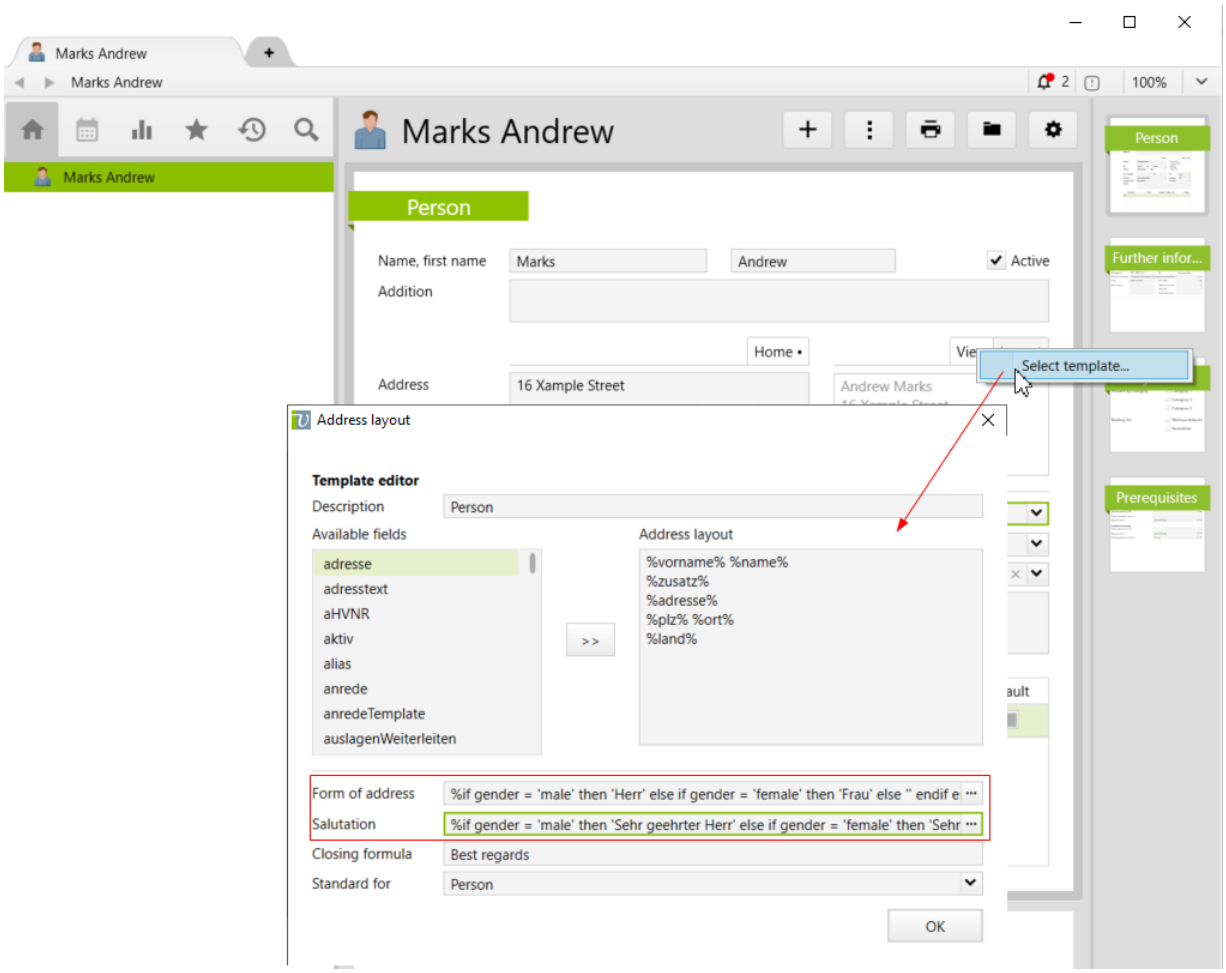

Figure 35 Selection of the gender "Divers" on the person

**Note:** The configuration applies to contacts and persons that are newly created. If the gender has been changed for existing entries, the salutation and letter salutation must be adjusted manually.

**The customization can be done as follows via the template editor of the address layout:**

On the company, contact, person or couple, open the window with the predefined layout templates by right-clicking on `View / Layout` and select `Edit`.... After that, the salutation and letter salutation can be manually adjusted in the template editor and confirmed with the `Apply` button.

To transfer the changes to all addresses, the script **Transfer address layout of one address to others** is available for download in the Knowledge Base at https://www.vertec.com/ch/kb/addlayouttoalloftype/.




Enter the following OCL expression manually:

Company **Salutation:**

```
%if sprache.asstring='DE' then 'Sehr geehrte Damen und Herren' else
if sprache.asstring='FR' then 'Madame, Monsieur' else
if sprache.asstring='IT' then 'Spettabile Ditta' else 'Dear Sir or
Madam' endif endif endif%
```

**Contact**

**Form of address:**

```
%if sprache.asstring='DE' then if gender = 'male' then 'Herr' else if
gender = 'female' then 'Frau' else if person.notNull then
person.vorname else vorname endif endif endif

else if sprache.asstring='FR' then if gender = 'male' then 'Monsieur'
else if gender = 'female' then 'Madame' else if person.notNull then
person.vorname else vorname endif endif endif

else if sprache.asstring='IT' then if gender = 'male' then 'Signore'
else if gender = 'female' then 'Signora' else if person.notNull then
person.vorname else vorname endif endif endif

else if gender = 'male' then 'Mr' else if gender = 'female' then 'Ms'
else if person.notNull then person.vorname else vorname endif endif
endif endif endif endif%
```

**Salutation:**

```
%if sprache.asstring='DE' then
  if gender = 'male' then 'Sehr geehrter' else if gender = 'female'
then 'Sehr geehrte' else 'Guten Tag' endif endif
else
  if sprache.asstring='FR' then
    if gender = 'male' then 'Cher' else if gender = 'female' then
'Chère' else 'Bonjour' endif endif
  else
    if sprache.asstring='IT' then
      if gender = 'male' then 'Egregio' else if gender = 'female'
then 'Egregia' else 'Buon giorno' endif endif
    else
      'Dear'
    endif
  endif

endif% %anrede% %if person.notNull then person.name else name endif%
```

**Person / Simple Address**

**Form of address:**

```
%if sprache.asstring='DE' then if gender = 'male' then 'Herr' else if
gender = 'female' then 'Frau' else vorname endif endif

else if sprache.asstring='FR' then if gender = 'male' then 'Monsieur'
else if gender = 'female' then 'Madame' else vorname endif endif

else if sprache.asstring='IT' then if gender = 'male' then 'Signore'
else if gender = 'female' then 'Signora' else vorname endif endif

else if gender = 'male' then 'Mr' else if gender = 'female' then 'Ms'
else vorname endif endif endif endif endif%
```

**Salutation:**

```
%if sprache.asstring='DE' then
  if gender = 'male' then 'Sehr geehrter' else if gender = 'female'
then 'Sehr geehrte' else 'Guten Tag' endif endif
```

```
else
  if sprache.asstring='FR' then
    if gender = 'male' then 'Cher' else if gender = 'female' then
'Chère' else 'Bonjour' endif endif
  else
    if sprache.asstring='IT' then
      if gender = 'male' then 'Egregio' else if gender = 'female'
then 'Egregia' else 'Buon giorno' endif endif
    else
      'Dear'
    endif
  endif

endif% %anrede% %name%
```

Couple

**Salutation:**

```
%if personA.notNull then personA.briefanrede else

if sprache.asstring='DE' then 'Guten Tag ' else if
sprache.asstring='FR' then 'Bonjour ' else if sprache.asstring='IT'
then 'Buon giorno ' else 'Dear ' endif endif endif + vornameA + ' ' +
nameA endif%

%if personB.notNull then personB.briefanrede else

if sprache.asstring='DE' then 'Guten Tag ' else if
sprache.asstring='FR' then 'Bonjour ' else if sprache.asstring='IT'
then 'Buon giorno ' else 'Dear ' endif endif endif + vornameB + ' ' +
nameB endif%
```

**Backwards compatibility**

The previously writable member isMale is now derived (calculated at runtime).

There is a new member gender - a text field in which the selected gender is stored as text.

'`male`' and '`female`' are fixed, the others can be added via system settings as described here.

isMale is `true` if `gender = 'male'`, `false` in all other cases.

This must be considered when importing addresses. If you try to write the member isMale, the error message appears:

```
no write access to derived member ismale
```

## 13.3 New subfolder structure in the Settings folder

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Settings folder gets a new subfolder structure to make it easier and clearer to find the individual settings folders. The structure is as follows:

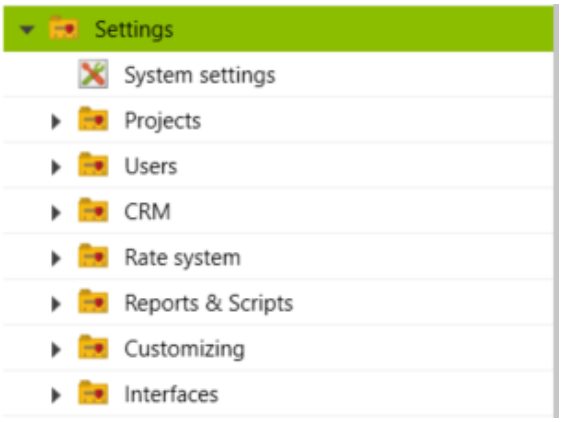
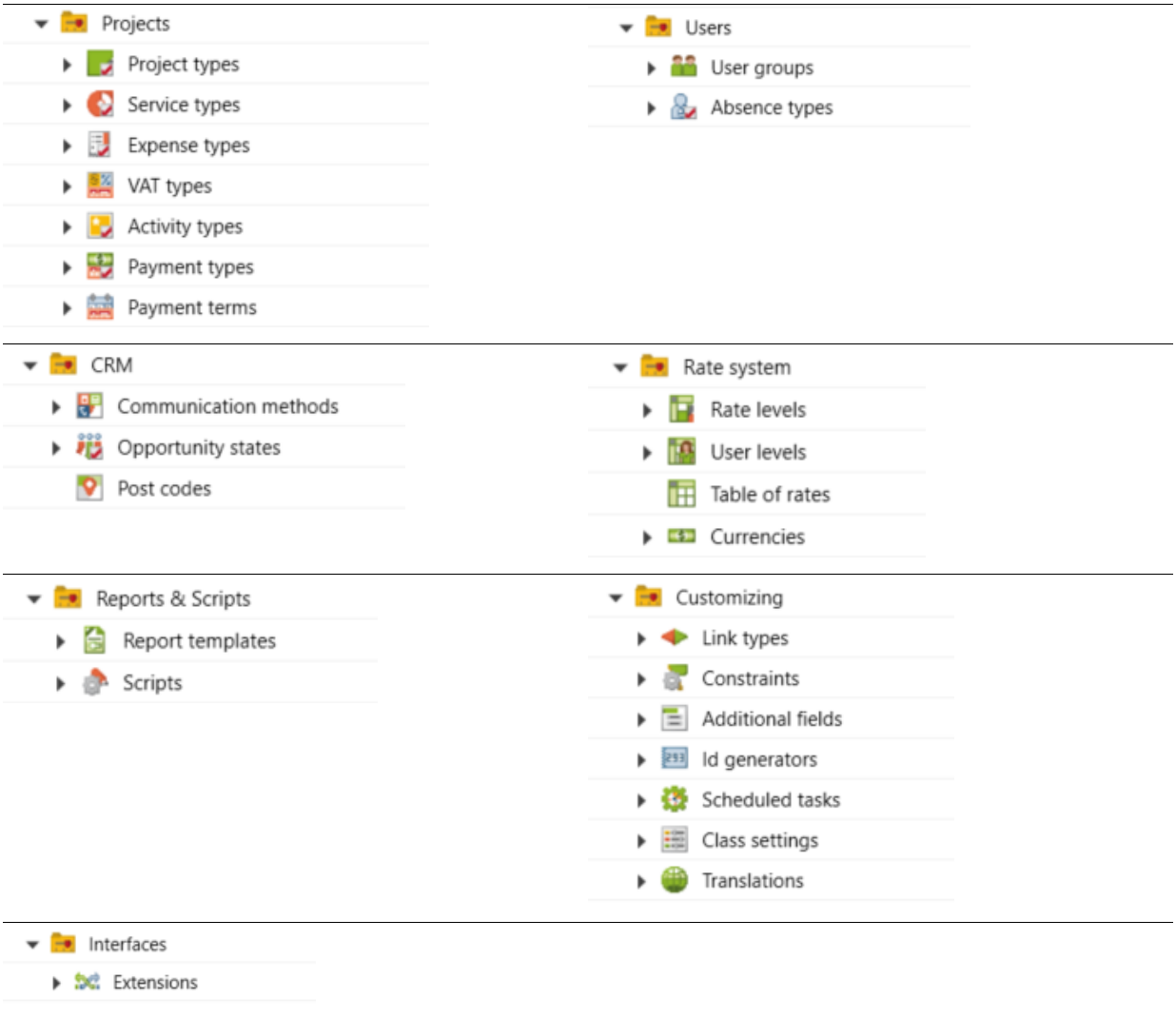



Figure 36 The settings folder is newly structured thematically

The system settings can still be found at the first place in the tree. The rest of the settings folders are divided as follows (line Expert - if you work with line Standard, certain folders are not visible):




This folder structure is only created for new Vertec installations.

## 13.4 Show favorites in new tab/window

Line: Standard, Expert | Module: PSA | Version: 6.5.0.5
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The favorites in the navigation view can now also be opened in a new tab or a new window via the right mouse button or the simultaneous click with the key combination with **Ctrl** or **Shift**.

## 13.5 IX-Keyword Folder as Dropdown in Lists

Line: Standard, Expert | Module: PSA | Version: 6.5.0.21
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the list settings, the `rndExclusiveKeywordFolder` renderer can be used to display **IX-Keyword Folder** as dropdowns.

In the **list settings**, the following must be entered for this in addition to the renderer:
- as expression the `internal id` of the keyword parentfolder
- as control element `cmbExclusiveKeywordFolder`

The name of the currently assigned subfolder of the folder hierarchy is displayed in the cell. Changing the combo box moves the row entry to the selected folder.

---

**Further Information** For more information, see the Renderer article in the Online Knowledge Base at https://www.vertec.com/kb/renderer.

---

## 13.6 Changes on rounding expenses and outlays

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

There are the following changes in the system setting `Round up/down VAT on expenses`:

- The system setting is now called only `Expenses and outlays round`.
- If the system setting is deactivated - so the expenses and outlays are not rounded individually, but only in the total sum - then they are rounded to two decimal places. Up to now, this has always been done using mathematical rounding (round-to-even). Now the commercial rounding rules are observed. This applies to the following attributes: `WertIntFW`, `MWSTBetragEKFW`, `WertExt`, `WertKosten`, `WertIntFWBrutto` and `MWSTBetrag`.

---

**Further Information** For more information about rounding, see the article **Rounding in Vertec** in the Online Knowledge Base at https://www.vertec.com/ch/kb/runden.

---

## 13.7 Overriding Windows region settings for fr-ch

Line: Standard, Expert | Module: PSA | Version: 6.5.0.18
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Windows region settings for the language French (Switzerland) are overridden because they are delivered incorrectly by Microsoft. Decimal numbers are separated with `.` and thousand numbers with `'`.

## 13.8 System setting "Service type for vaction" removed

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Since the entry of vacations via services causes performance problems, among other things, the associated system setting Project > `Service type for vacation` is removed. The system setting is also removed from existing databases if no service type is set.

---

**Further Information** For more information on **recording vacations**, visit the Online Knowledge Base at https://www.vertec.com/kb/ferienerfassung/.

---

# 14 Translations

## 14.1 Translation entries in Vertec

Line: Standard, Expert | Module: PSA | Version: 6.5.0.9
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Translations (`TranslationItems`) can now be created and managed directly in the **Settings > Customizing > Translations** folder in Vertec:

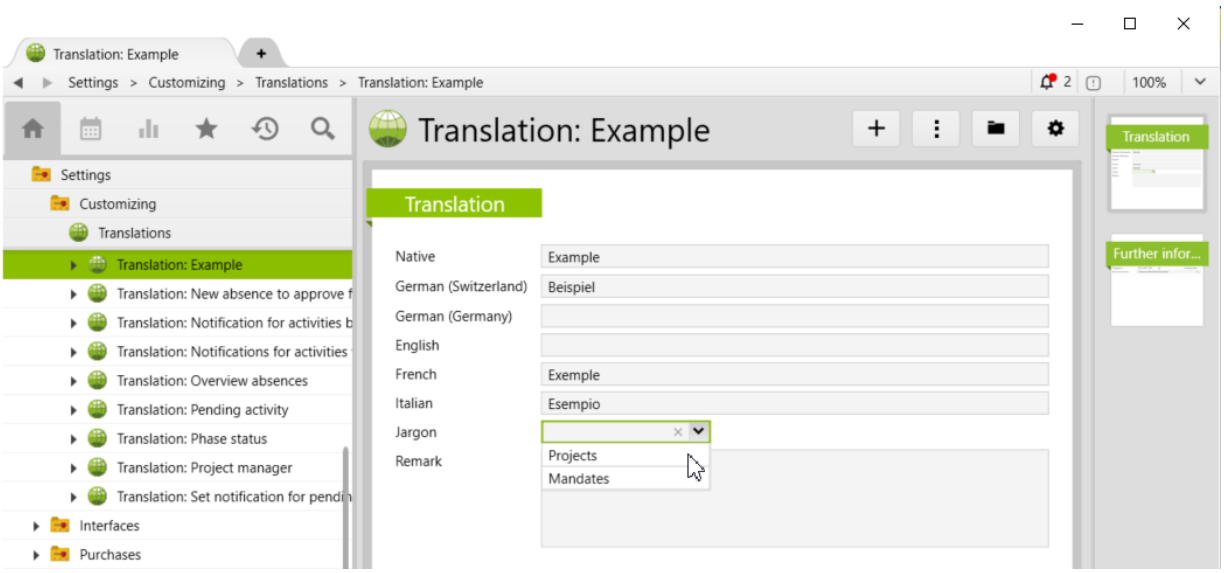



Figure 37 Create a translation

The **native** term is the original term from Vertec that is to be translated.

For each available language there is a field in which the translation can be entered.

Translations are considered as follows:

- If a translation text is defined for the current language, it will be displayed.
- German (Germany) and German (Switzerland) inherit from each other.
- If the translation text is empty for the current language, the original translation of the Vertec translation system is displayed, if available.
- If a jargon is defined (optional), the translation is only considered if this jargon is active.

## 14.2 German (Germany) language support in ML strings

Line: Standard, Expert | Module: PSA | Version: 6.5.0.9
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Multi-Language-Strings (**MLStrings**) can receive the language `DD` (German Germany) in addition to the language `DE` (German Switzerland). To ensure backwards compatibility, when a German language is entered, the term is automatically used for the other German language before the native language comes into play. In case of an empty `DD` translation, the `DE` text will be used and vice versa.

---

**Further Information** For more information, see the article **Multilingualism with Vertec** in the Online Knowledge Base at https://www.vertec.com/kb/mehrsprachigkeit-mit-vertec.

---

## 14.3 Translation of additional selection fields

Line: Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The strings in additional selection fields are translated if the additional field definition (additional field class) has an **Entry Id**.

---

**Further Information** For more information, see the **Additional Fields** article in the Online Knowledge Base at https://www.vertec.com/kb/zusatzfelder.

---

# 15 Interfaces

## 15.1 Supplied extensions as built-in code

Line: Expert | Module: PSA | Version: 6.5.0.16
Mode of Operation: On-Premises | Apps: Full-featured

The extensions supplied with Vertec are no longer stored on the file system, but are supplied directly as built-in code. The code is therefore directly visible on the extension object for all interfaces:

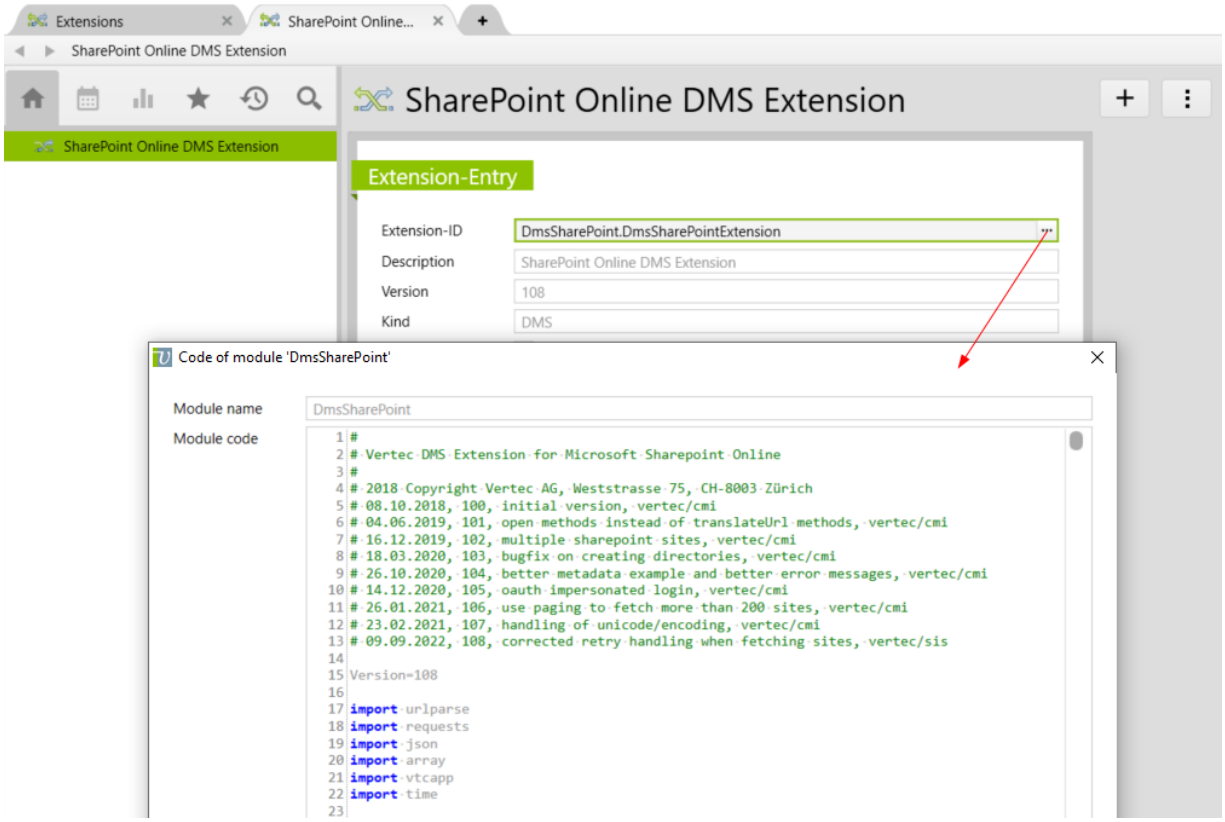



Figure 38 Built-in Code

A change arises with the sample extensions **DmsExampleExtension** and **FibuExampleExtension**. These must be registered manually (create extension object and specify the extension ID as described):

– DmsExampleExtension: `DmsExampleExtension.DmsExampleExtension`
– FibuExampleExtension: `FibuExampleExtension.ExampleOfflineDebiExtension` / `FibuExampleExtension.ExampleOnlineDebiExtension`

The modules FibuBase and ExtensionBase are still importable.

## 15.2 Importing customer-specific extensions as script modules

Line: Expert | Module: PSA | Version: 6.5.0.16
Mode of Operation: On-Premises | Apps: Full-featured

Custom extensions whose Python file is located in the Extensions folder and which are registered and **installed** as an extension are imported as a script entry after updating to the latest version. The Python modules used are also imported from the extensions folder as scripts. Care is taken to ensure that multiple script entries are not created for multiple interfaces from the same module. If a used module is not found in the Extensions folder, a corresponding warning is written to the `Vertec.Desktop.log`.

**Backwards Compatibility**
If you have customer-specific extensions in use, which are based on a copy of a previously existing extension, these must be adapted beforehand if necessary. This concerns the interfaces **FibuAbacus.py**, **FibuSesamText.py** and **Outlook.py** as well as customer-specific interfaces which use the attribute `__file__` in their code. This attribute is no longer available. As an alternative we provide the method `self.Host.getinstallationpath()`, which returns the path of the Vertec installation. Based on this, `os.join(instpath, 'Extensions')` can be used to construct the path to the Extensions folder. Example for Abacus:

```
installationPath=self.Host.getinstallationpath()
self.Adapter.dataPath=os.path.join(installationPath, 'Extensions', 'AbacusData')
```

## 15.3 Support of the EZ procedure in the AbaConnect XML accounts receivable extension

Line: Standard, Expert | Module: PSA | Version: 6.5.0.5
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the AbaConnect XML accounts receivable extension, support for posting QR-Debitor invoices has been retrofitted. Vertec supplies the EZ procedure (`PaymentOrderProcedure`) based on the Ver-tec payment type used. The assignment is made on the following basis:

- A suitable **EZ procedure** must be defined in Abacus, whose deposit account must correspond to the payment account (`VESR participant number` for ESR, `IBAN` for QR).
- The code of the payment type in Vertec must correspond to the number of the EZ procedure in Abacus. If the code is not numeric, no EZ procedure is transferred.
- Vertec determines the payment type based on the payment account. If an IBAN is defined, we search for the payment type based on this IBAN. If a payment type is found, the EZ procedure is set in Abacus.

## 15.4 Dialog for cancellation date in the Abacus Web Debtor extension

Line: Standard, Expert | Module: PSA | Version: 6.5.0.12
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

When canceling bookings, a dialog is now called up via Internal credit note, with which a cancellation date can be entered. The default value is today's date.

## 15.5 Multiple QR-IBAN per supplier in the Abacus Web Creditor extension

Line: Standard, Expert | Module: Fremdkosten | Version: 6.5.0.16
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

When posting accounts payable in the **Abacus Web Creditor extension,** multiple QR-IBANs are supported by determining the `BeneficiaryAccount` matching the current QR-IBAN. If none exists, a new one will be created.

## 15.6 Separation of street and house number in Abacus

Line: Standard, Expert | Module: PSA | Version: 6.5.0.23
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

As of version 2021, Abacus has introduced a separation of street and house number in the addresses. Therefore, the Abacus Web as well as the AbaConnect XML extensions have been adapted so that the addresses of the contacts in Vertec are exported to Abacus with separated street and house number.

## 15.7 Posting QR Invoices in the SAGE 200 creditors accounting extension

Line: Standard, Expert | Module: Fremdkosten | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The SAGE 200 creditors accounting extension now supports the posting of QR invoices.

## 15.8 Payment reconciliation in the DATEV Debtor Extension

Line: Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: On-Premises | Apps: Full-featured

If 2 payments with the same date and text were read in, one of the two payments was ignored. This problem has been fixed and both payments appear in Vertec.

**Note:** For backwards compatibility, in cases where the problem already occurs, the already im-ported payment with the same date and text must be manually deleted in Vertec so that the document numbers can be re-entered and thus both payments can be imported.

## 15.9 Increase of invoice number length for DATEV extensions

Line: Expert | Module: Leistung & CRM, Fremdkosten | Version: 6.5.0.15
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The DATEV Debtor and Creditor Extensions the invoice number (`document field1` in DATEV) to 12 characters. However, this field may be up to 36 characters long. The interfaces have been adapted accordingly.

# 16 List Controller

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

A List Controller is a Python class that is defined to influence the behavior of a list view.

The main current use of List Controllers is in **Resource planning views** (see 2.5). However, List Controllers can also be set in the properties of **folders** as well as on **link types** to define the lists that are displayed.

Functionality of a List Controller:

- A List Controller is a Python class that can (optionally) be used with a list (folder, link type or the new Resource planning views (see 2.5)).
- The List Controller is specified via `<Modulename>.<Classname>`.
- If you do not use a List Controller, the lists behave as before.
- The List Controller can preload objects and data, which can provide significant performance benefits.
- A List Controller can also precompute data, which can then be used by Custom Renderers (see 3) in the columns. This is also performance relevant, as certain calculations only need to be done once instead of per cell as before.
- A List Controller can manage dynamic columns. Columns can be defined as Dynamic and are repeated against the right. How many times and what the heading is called can be controlled via List Controller. The exact procedure is described in section 16.1.
- The List Controller can, but does not have to, calculate the objects that are in the list. Calculating the objects (via `get_row_objects`, see below) is currently only supported for Resource planning views.

**Methods**
Eine List Controller Klasse kann folgende Methoden definieren:

| | |
|---|---|
| initialize(self, subscriber) | Is called when the list is calculated. In the initialize method data can be precalculated, which must be available for the cells renderer of the list. |
| get_dynamic_column_count(self, subscriber) | Returns the number of repetitions of the columns marked as `dynamic` in the list. Each repetition is displayed in the list as a group with a group title.<br>In case of lists that should display one column for each date interval to be displayed, this value corresponds to the number of intervals to be displayed (e.g. months).<br>For more information on dynamic columns, see Chapter 16.1 below. |
| get_dynamic_column_title(self, column_index, subscriber) | Calculates the group title of a dynamic column group. The passed column_index can take values from 0 (first group) to number of groups-1 (last group).<br>In case of date column groups the date interval (e.g. month name) is returned as string.<br>More information about dynamic columns and column groups can be found in chapter 16.1 below. |
| And additionally, when using the List Controller in Resource planning views: | |
| initialize_with_period(self, start, end, interval, subscriber) | Special variant of initialize for Resource planning, which additionally receives the start and end of the currently displayed period as well as the configured planning interval (`0` for days, `1` for weeks, `2` for months). In the case of Resource planning lists, the data provider for the planning data is initialized |

| | |
|---|---|
| | here. For Resource planning views, initialize_with_period is preferably called instead of initialize if the List Controller implements this method. |
| get_row_objects(self, subscriber) | Calculates the row objects that are displayed in the list.<br>For Resource planning views, this determines all rows for which a plan value is entered. |
| get_row_objects_type(self) | Calculates the type of the row objects. If this is constant, it can also be specified as `row_objects_type` attribute.<br>Only useful in combination with `get_row_objects`. Supported only for resource plan views. |
| add_row_object(self, obj) | Used for the asterisk line functionality to add more entries to be scheduled to the list.<br>This method must be present to display an asterisk line for lists with List Controller and Show Add Row option turned on. Supported for Resource planning views only. |

The object instance (`self`) of a List Controller offers the following attributes and methods:

| | |
|---|---|
| self.evalocl(expression, rootobj=None) | Allows the evaluation of OCL on the self OCL Evaluator. When called, the appropriate subscriber is automatically used. If rootobj is optionally specified, the evaluation is performed on this object, otherwise globally. |
| self.context | Is the context object of the list. For folders and link containers this is the Container object (folder or link container).<br>In case of lists defined via Resource planning view for single objects, the context is the single (user entry) object. |
| self.get_context_entries() | Returns a list of the following objects depending on self.context:<br>If self.context is a container object, then the entries of the container are returned. In this case, the entries of the container are subscribed to automatically.<br>Otherwise, a list is returned that contains only the self.context object. |
| self.owner | Returns the owner (can be of type `LinkRolle`, `ViewType` (Resource planning view) or `AbfrageOrdner` (QueryFolder)) of the List Controller. |

In a list with configured List Controller, a Custom Renderer (see Chapter 3) has access to the List Controller instance via the `self.controller` attribute.

As soon as a List Controller is specified in a List Controller field, the button with the three dots appears. Clicking on it opens the code for viewing:

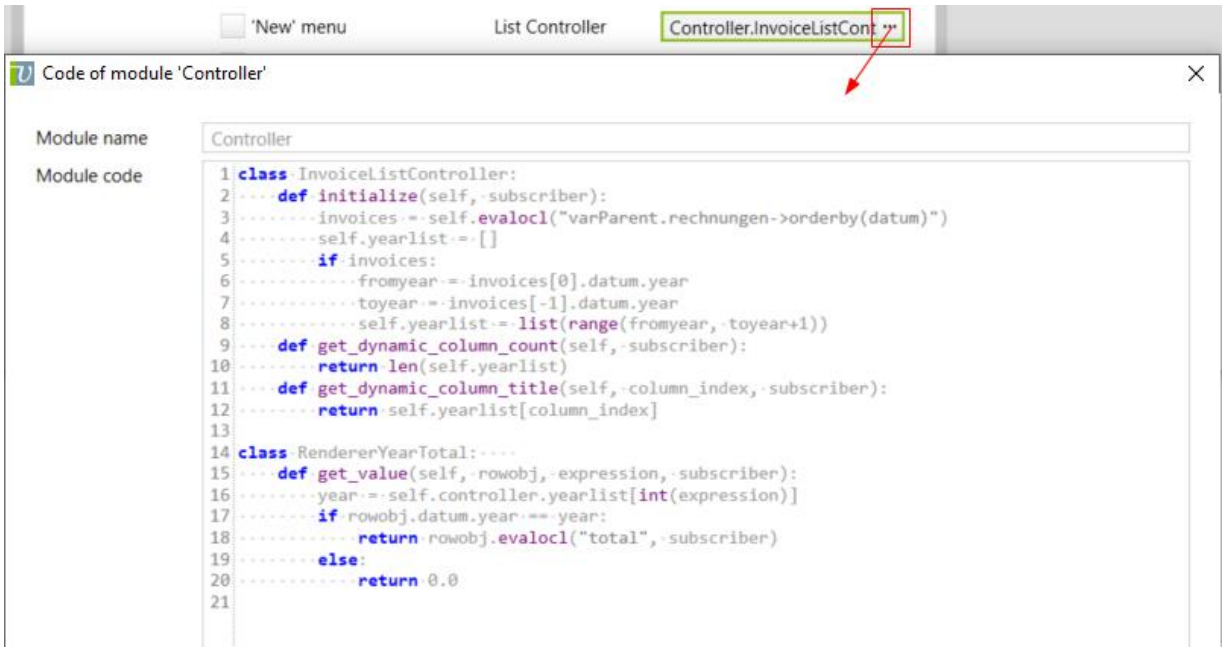



Figure 39 Clicking the button with the three dots in the List Controller field opens the corresponding code for viewing

A list of the List Controllers supplied as standard can be found in the chapter 18.2.

## 16.1 Dynamic Columns

Columns in the list settings can have the `Dynamic` checkbox set as of Vertec 6.6.

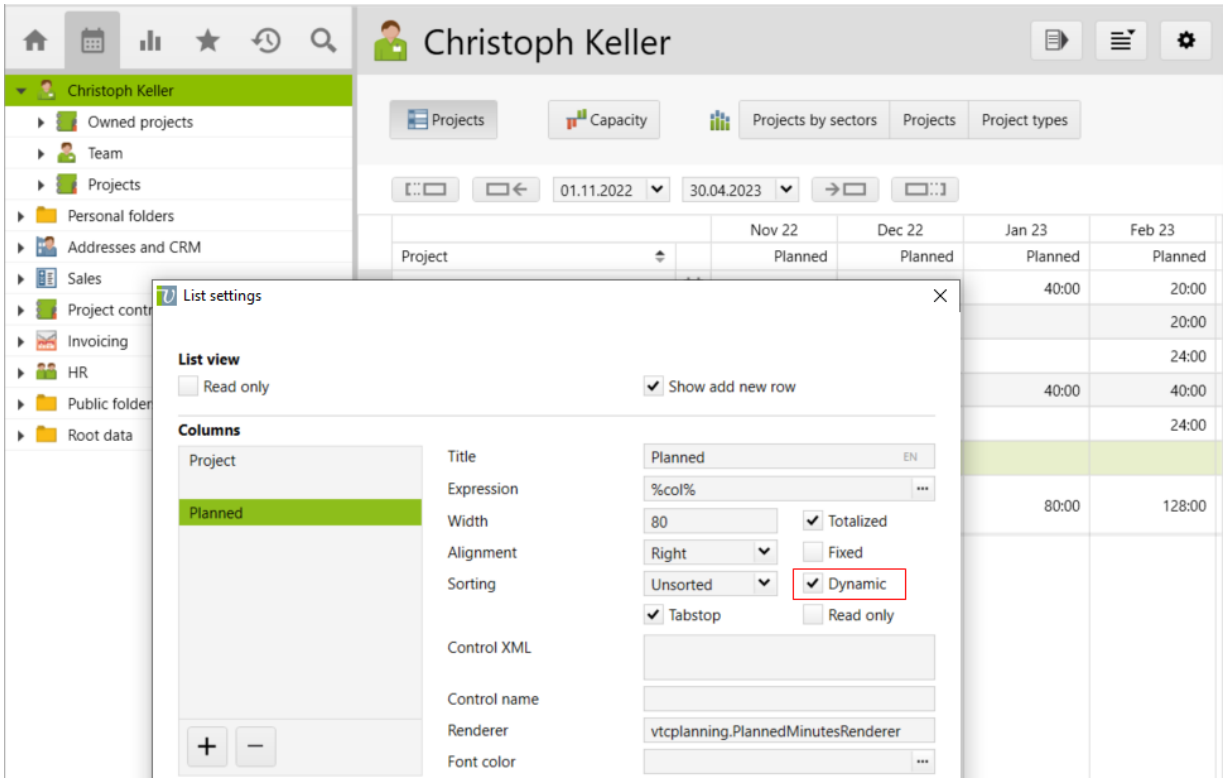



Figure 40 A column marked as Dynamic in the Time table User - Projects

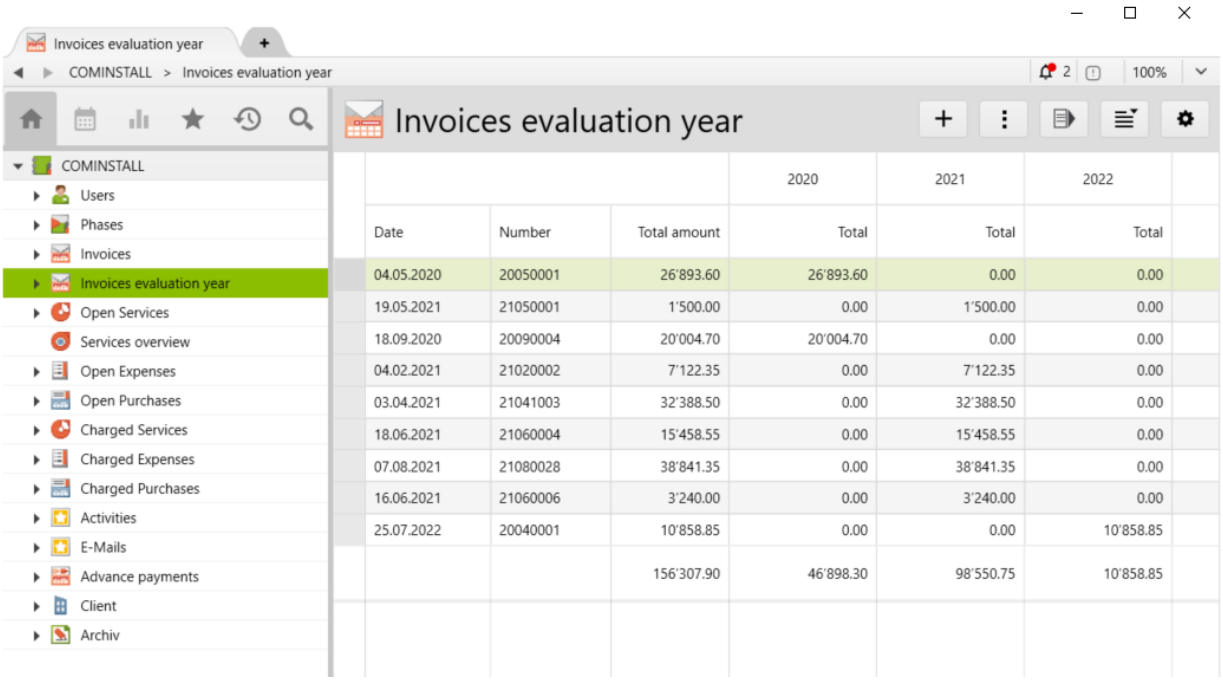
Appropriately marked columns are repeated as many times as the method `get_dynamic_column_count()` (see previous section) returns.

Several dynamic columns must be defined one after the other and then duplicated as a whole. The first group is given index 0, the second 1, and so on.

This index is accessed with the expression `%col%`. This can be referred to in the renderer and the data output accordingly (see example in Section 16.2).

## 16.2 Example

In the example, invoice evaluation is made on project. It is a normal invoice list, but the amounts are sorted by year.




| | | | 2020 | 2021 | 2022 |
|---|---|---|---|---|---|
| Date | Number | Total amount | Total | Total | Total |
| 04.05.2020 | 20050001 | 26'893.60 | 26'893.60 | 0.00 | 0.00 |
| 19.05.2021 | 21050001 | 1'500.00 | 0.00 | 1'500.00 | 0.00 |
| 18.09.2020 | 20090004 | 20'004.70 | 20'004.70 | 0.00 | 0.00 |
| 04.02.2021 | 21020002 | 7'122.35 | 0.00 | 7'122.35 | 0.00 |
| 03.04.2021 | 21041003 | 32'388.50 | 0.00 | 32'388.50 | 0.00 |
| 18.06.2021 | 21060004 | 15'458.55 | 0.00 | 15'458.55 | 0.00 |
| 07.08.2021 | 21080028 | 38'841.35 | 0.00 | 38'841.35 | 0.00 |
| 16.06.2021 | 21060006 | 3'240.00 | 0.00 | 3'240.00 | 0.00 |
| 25.07.2022 | 20040001 | 10'858.85 | 0.00 | 0.00 | 10'858.85 |
| | | 156'307.90 | 46'898.30 | 98'550.75 | 10'858.85 |

Figure 41 Example of an invoice list with amounts sorted by year

The calculation of the data is done in the List Controller. The display in the list is provided by Custom Renderer (see Chapter 3).

For this purpose, the following script is registered, in our example under the name `Controller`.

It contains the List Controller and the Total Renderer:

```
class InvoiceListController:
    def initialize(self, subscriber):
        invoices = self.evalocl("varParent.rechnungen->orderby(datum)")
        self.yearlist = []
        if invoices:
            fromyear = invoices[0].datum.year
            toyear = invoices[-1].datum.year
            self.yearlist = list(range(fromyear, toyear+1))
    def get_dynamic_column_count(self, subscriber):
        return len(self.yearlist)
    def get_dynamic_column_title(self, column_index, subscriber):
        return self.yearlist[column_index]

class RendererYearTotal:
    def get_value(self, rowobj, expression, subscriber):
        year = self.controller.yearlist[int(expression)]
        if rowobj.datum.year == year:
            return rowobj.evalocl("total", subscriber)
        else:
            return 0.0
```

A wrapper link type was created for the list and the List Controller was specified in it in the form `<Modulename>.<Classname>`:

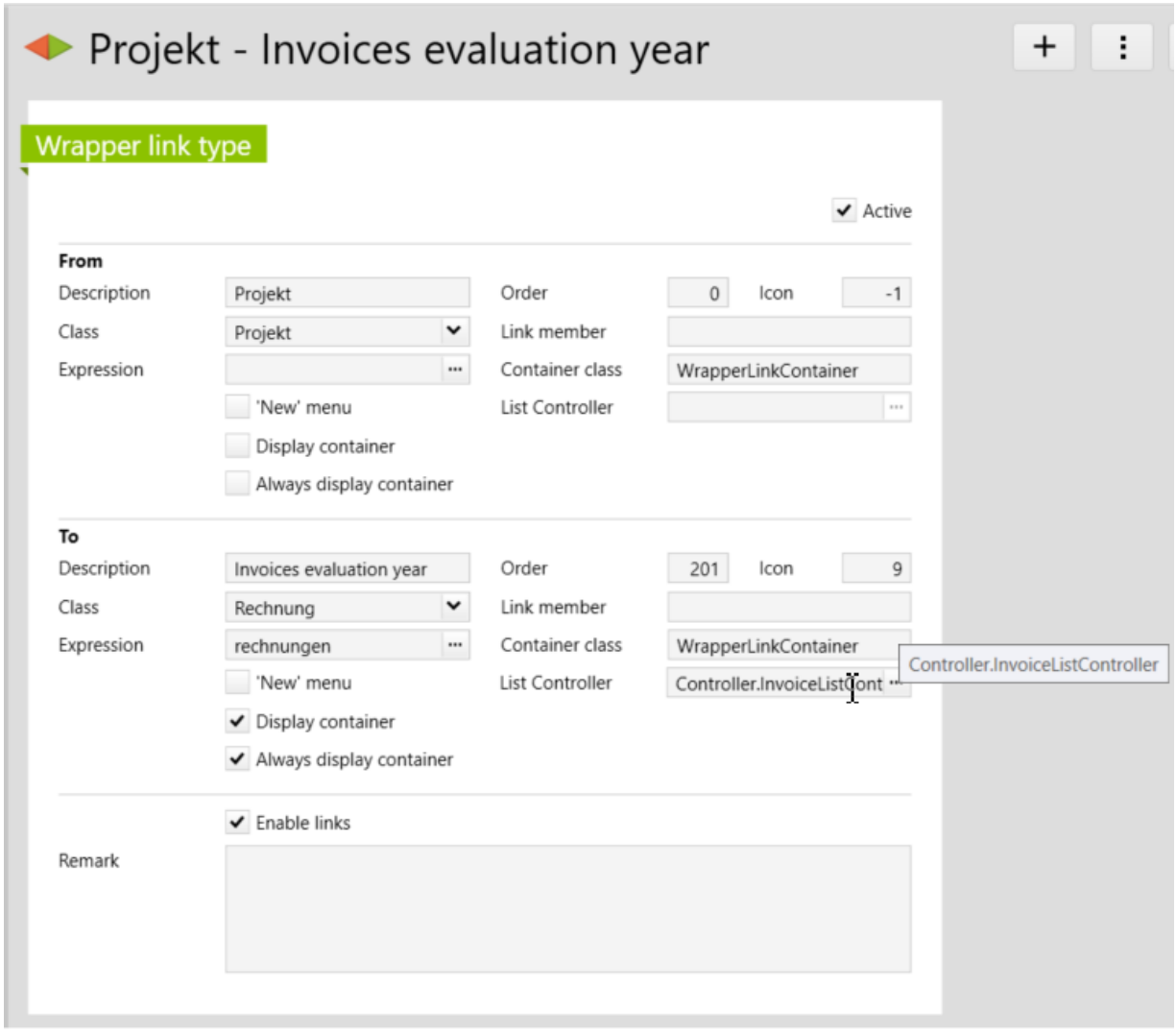



Figure 42 The wrapper link type with the List Controller, which calculates the page of invoices

In turn, a dynamic column is created in the list settings and the renderer is specified, also in the form `<Scriptname>.<Klassenname>`:

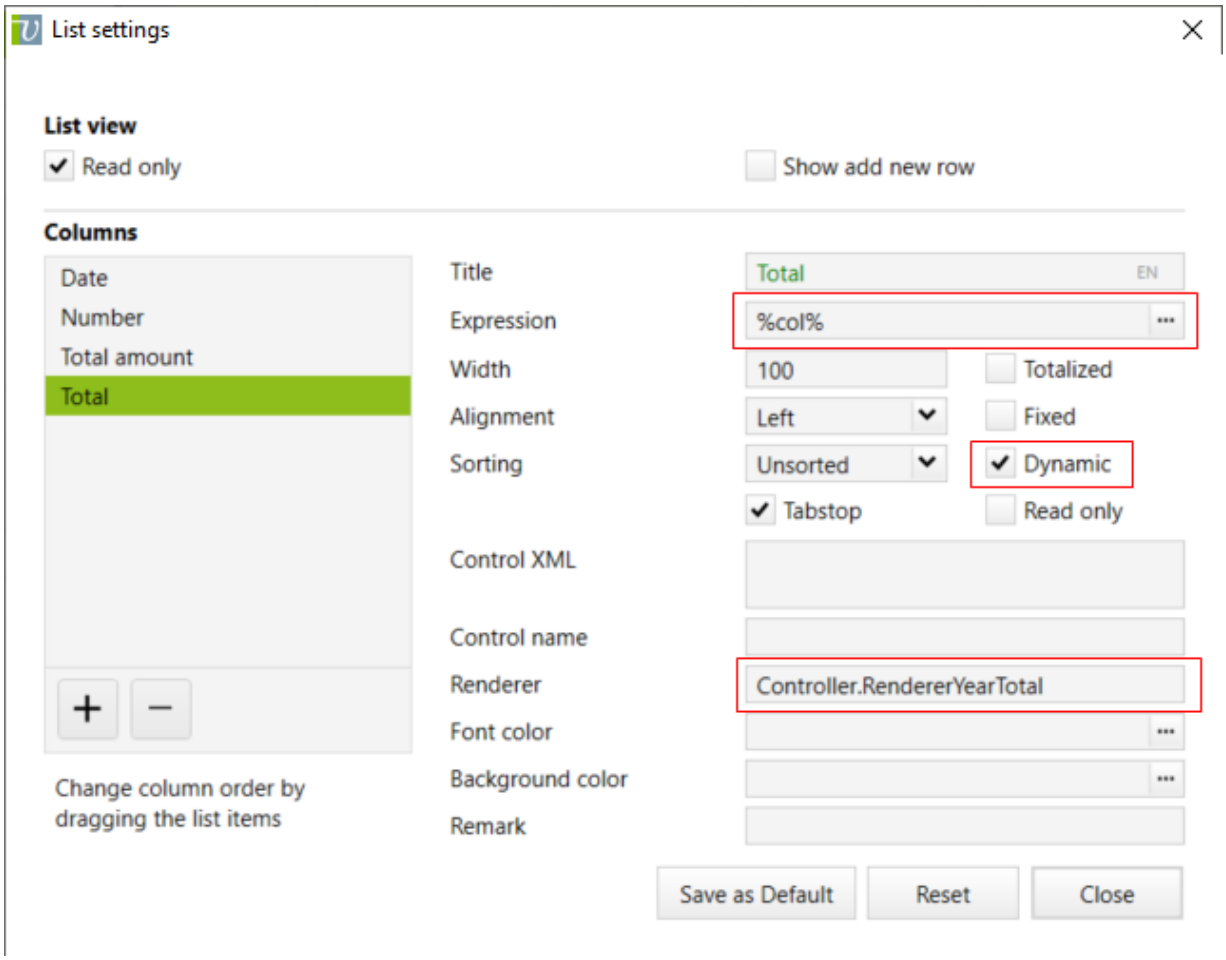



Figure 43 List settings of the dynamic column with specification of the renderer

# 17 Technical

## 17.1 New Firebird Version

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: On-Premises | Apps: Full-featured

The Vertec setup provides the Firebird version 4.0.1.2694. If an older version is installed, the setup suggests an update, performs it automatically and sets the original DB password.

## 17.2 License dialog in the Cloud and Web App

Line: Standard, Expert | Module: PSA | Version: 6.5.0.9
Mode of Operation: On-Premises | Apps: Cloud App, Web App

Licenses can also be entered and changed in the Cloud and Web App. To do this, the `Modify license...` option appears in the **Settings** menu, which opens a dialog in which the license name and license code can be entered. The license is immediately checked for validity. With the setting `Show License Dialog = False` in the **[CloudServer] section** of the Vertec.ini file, the display of this option can be deactivated. The default setting is `True`.

---

**Further information** For more information on entering licenses, see the article **Licensing / Initializing Vertec** in the Online Knowledge Base at www.vertec.com/kb/lizenzieren/.

---

## 17.3 Removed "Open in new window" option in the Web App

Line: Standard, Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Web App

The `Open in new window` option is no longer offered in the Web App, because web browsers cannot distinguish between opening a new tab or window in this case. Opening in a new window can be enabled by dragging out a new tab.

## 17.4 Cloud App Installer checks TLS certificate more restrictively

Line: Standard, Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: On-Premises | Apps: Cloud App

The Cloud Installer only accepts valid certificates when downloading files from the Vertec Cloud Server if it is a TLS connection. For non-trusted (self-signed) certificates, this means that the certificate must be present in the Windows certificate store under "Trusted Root Certification Authorities" as trusted.

Declaring the certificate as trusted in the browser is no longer sufficient. The certificate must be imported manually on all clients in Windows to be accepted as "secure".

The name of the certificate must match the host name of the server. For example, if the server and cloud installer are running locally, a localhost certificate is required in the trusted root certificate authorities of the computer user. If the certificate name does not match, an error is displayed.

---

**Further information** For more information about TLS operation via a self-signed certificate, see the article Cloud Server: Deployment and Security in the Online Knowledge Base at www.vertec.com/kb/cloudserver.

---

## 17.5 New command line parameters for Cloud Installer

Line: Standard, Expert | Module: PSA | Version: 6.5.0.20
Mode of Operation: On-Premises | Apps: Cloud App

The Cloud Installer, which installs the Cloud App locally, can additionally be started with the following parameters:

– The `/dir` parameter can be used to specify an alternative folder for the installation. If the directory does not yet exist, an attempt will be made to create it.

  – `Vertec.CloudInstaller.exe /install /dir=<my directory>`

– With the `/silent` parameter, no user interface or dialogs are displayed. In case of error the return value is `!=0`. In the console the error message is displayed.

## 17.6 Command line parameters for the Cloud App

Line: Standard, Expert | Module: PSA | Version: 6.5.0.22
Mode of Operation: On-Premises | Apps: Cloud App

The Cloud App supports the following command line parameters:

– /noevents
– /script
– /super
– /paysync
– /batch

---

**Further Information** For more information on what these parameters do exactly, see the **Command Line Parameters** article in the Online Knowledge Base at www.vertec.com/kb/parameter.

---

## 17.7 Cloud Server in maintenance mode

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Cloud App and Phone App

If the Cloud Subscription System or the Cloud Server is in maintenance mode, corresponding messages are now displayed in the Cloud App and in the Phone App.

WIth Cloud Abo:

```
The Vertec Cloud Subscription System is currently in maintenance mode. Please try again after the maintenance is completed.
```

On-Premises:

```
The Cloud Server is currently in maintenance mode. Please try again after the maintenance is completed.
```

---

**On-Premises** For information on the maintenance mode of Cloud Servers, see the article on the Cloud Server Management Console at www.vertec.com/kb/cloudserver/.

---

## 17.8 Phone App: QR code for server address

Line: Standard, Expert | Module: PSA | Version: 6.5.0.9
Mode of Operation: Cloud Subscription / On-Premises | Apps: Phone App

In the App Portal, a QR code with the corresponding server address appears under Phone App. In the Phone App settings, this QR code can be scanned to enter the server address:

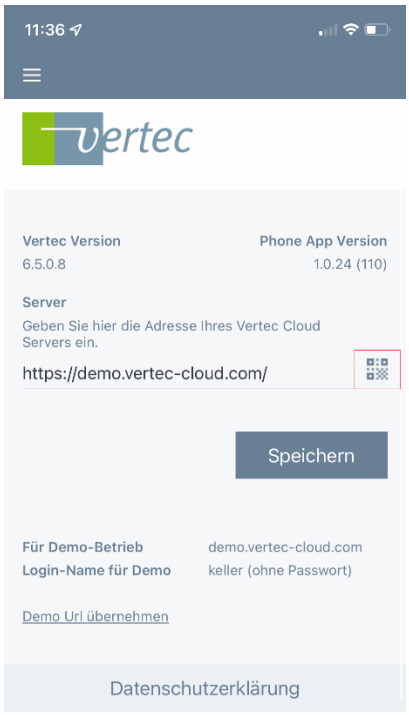



Figure 44 Entering the server address via QR code

## 17.9 Only full-featured apps delete invalid objects

Line: Standard, Expert | Module: PSA | Version: 6.6
Mode of Operation: On-Premises / Cloud Subscription | Apps: full-featured

Deleting invalid objects when exiting Vertec sessions could lead to unexpected data loss for **non**-full-featured apps (such as the Outlook App) due to the session timeout. Therefore, as of Vertec version 6.6, invalid objects are only deleted by sessions of full-featured apps (Desktop App, Cloud App and Web App).

**Further information** For more information about invalid entries, see the article www.vertec.com/kb/gueltigkeit/ in the Online Knowledge Base.

## 17.10 Improvements database convert

Line: Standard, Expert | Module: PSA | Version: 6.5.0.20
Mode of Operation: On-Premises | Apps: Full-featured

Until now the database convert failed if system tables were defined which Vertec does not know. As of version 6.5.0.20, only Vertec tables are considered during the database convert and the others are ignored, so that the convert no longer fails.

## 17.11 Demo DB removed from setup

Line: Standard, Expert | Module: PSA | Version: 6.5.0.22
Mode of Operation: On-Premises | Apps: Full-featured

The Demo DB has been removed from the Vertec setup. The following applies:

– No shortcut is created for the Vertec Desktop App (Demo DB).
– The `VertecDemo.fbk` file is no longer supplied.
– No [DemoDB] section in the Vertec.ini file for new installations.
– Existing Demo DB's are not deleted from the file system.

## 17.12 Empty database delivers Firebird stored procedure

Line: Standard, Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: On-Premises | Apps: Full-featured

During the database convert, a stored procedure for updating the index statistics is created in the Firebird database. Now the empty database is also delivered including this procedure.

---

**More information** For more information about Firebird and index statistics, see the **Database Performance and Index Statistics** article in the Online Knowledge Base at www.vertec.com/kb/performance-und-index/.

---

## 17.13 Support for additional MS-SQL driver

Line: Standard, Expert | Module: PSA | Version: 6.5.0.9
Mode of Operation: On-Premises | Apps: Full-featured

The **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** driver is supported. However, the default provider in a new installation remains `SQLOLEDB`.

---

**Further Information** For more information, see the article **The Vertec.ini** - File in the Online Knowledge Base at www.vertec.com/kb/ini.

---

# 18 Python Functions

## 18.1 Python module vtcplanningcore for Resource planning

Line: Expert | Module: Ressourcenplanung | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the **vtcplanningcore** Python module there are the following methods:

| | |
|---|---|
| increment_interval_date(datum, increment): date | Function for incrementing and decrementing period start dates. Increments the specified date by the number of intervals specified in `increment`. A negative number of intervals can be specified for decrementing. |
| get_planning_level(): string | Returns the planning level as a string.<br>Possible results: `'Projekt'` or `'ProjektPhase'`. |
| get_planning_interval_type(): string | Returns the planning interval as a string.<br>Possible results: `'day'`, `'week'` or `'month'`. |
| show_bulk_planning_dialog(left_obj, right_obj) | Shows a dialog for efficiently setting multiple planning values for the specified objects.<br>The order in which the objects are provided does not matter, but the caller must ensure that one of them is an object of type AbstractWorker, i.e. either a ProjectWorker or a PlanningWorker, and the other is either a Project or a ProjectPhase. |

**The ResourcePlanningProvider helper object**

For the implementation of Resource planning lists, a `ResourcePlanningProvider` class is also available in the module vtcplanningcore, which allows the reading and setting of planning data and which can be used in a List Controller.

A ResourcePlanningProvider object is created for a list of entries (workers, projects or phases) and a time period. The provider then loads all planning data for these source objects and for the specified period. Subsequently, the planned values can be retrieved or set.

A planned value in the ResourcePlanningProvider is always valid for 2 entries and one date, e.g. for the user "Christoph Keller", the project "AZZ2" and the month starting with 01.08.2022.

One entry (`sourceEntry`) must come from the list of entries specified during initialization. We call the other entry "opposite" (`othersideEntry`).

The following methods are available on a ResourcePlanningProvider object:

| | |
|---|---|
| ResourcePlanningProvider(sourceEntries, start, end) | Constructor, creates a planning provider and loads the planning data for the specified objects and period. |
| add_entry(obj) | Adds a new object to the source list (sourceEntries) of the provider. |
| add_otherside_entry(obj) | Temporarily adds a new object to the list of otherside entries. This allows scheduled values to be entered for this object with `set_planned_minutes()`. |
| get_planned_minutes(date, sourceEntry, othersideEntry, subscriber): int | Returns the planned time in minutes for the planning cell with interval `date` and the two specified entries. If no planning data is available, None is returned. |

| | |
|---|---|
| set_planned_minutes(date, sourceEntry, othersideEntry, value) | Sets the planned time in minutes for the planning cell with interval `date` and the specified entries.<br>Setting a `None` value as value removes the planning data for this cell. |
| has_write_access(sourceEntry, othersideEntry, subscriber): boolean | Checks whether write permissions exist for the two objects and the date interval. |
| get_planned_minutes_aggregated (source_entry, otherside_entry, dateFrom, dateTo=None, subscriber=None): int | Returns the aggregated scheduled time for the two specified entries in the specified time period.<br>– Both **source_entry** and **otherside_entry** can be `None`, in which case all entries known to the current ResourcePlanningProvider are considered.<br>– **dateFrom** is required, **dateTo** is optional. If no dateTo is specified, only the interval matching dateFrom is considered.<br>Returns `None` if no plan values are found with the specified criteria. |
| get_net_capacity_minutes(self, worker, dateFrom, dateTo=None, subscriber=None): int | Net capacity in minutes per worker.<br>– **worker**: AbstractWorker, PlanningWorker or ProjectWorker (see 2.4)<br>– **dateFrom**: Start date<br>– **dateTo**: Optional, end date. If None, the end date of the inter-vall of dateFrom is used.<br>Thus, for example, on a project, the workers who are already overloaded can be highlighted in color in the list. |
| get_gross_capacity_minutes(self, worker, dateFrom, dateTo=None, subscriber=None): int | Gross capacity in minutes per worker.<br>– **worker**: AbstractWorker, PlanningWorker or ProjectWorker (see 2.4).<br>– **dateFrom**: Start date<br>– **dateTo**: Optional, end date. If None, the end date of the interval of dateFrom is used.<br>Thus, for example, on a project, the workers who are already overloaded can be highlighted in color in the list. |
| get_remaining_capacity_minutes(worker, dateFrom, dateTo=None, subscriber=None): int | Remaining free capacity in minutes for a given worker and a given date range (net_capacity - planned_minutes).<br>– **worker**: AbstractWorker, PlanningWorker or ProjectWorker (see 2.4).<br>– **dateFrom**: Start date<br>– **dateTo**: Optional, end date. If None, the end date of the intervall of dateFrom is used. |
| get_otherside_entries(subscriber): list of entries | Returns the list of entries for which plan data already exists, starting from the objects `sourceEntries` specified in the constructor. |
| generate_date_range(start, end) | Returns a list of date values for planning intervals from `start` to `end`. Depends on the system-wide setting of the planning interval (months, weeks, days). |
| get_start_preceding_interval(): date | Returns the date of the interval preceding the start interval. |
| get_column_title_by_date(date): string | Returns the appropriate column title when the date value for a given planning interval is specified. |

ResourcePlanningProviders are commonly used in List Controllers and Custom Renderers for planning lists. The List Controller instantiates a planning provider, which is then used to display and set the values in plan cells.

### vtcplanningcore as Stub File

The vtcplanningcore module is also available as a Python stub file. It is called **vtcplanningcore.py** and is located in the subfolder `PythonStubs` in the Vertec installation directory like the other stub files.

**More Information** For more information about the included Python stub files, see the KB article at www.vertec.com/kb/pythonstubfiles/.

## 18.2 Python module vtcplanning for Resource planning

Line: Expert | Module: Ressourcenplanung | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

In the **vtcplanning** Python module we provide standard implementations of List Controllers for Resource planning (see also chapter 13).

These all have an `initialize_with_period(start, end, interval)` method, which is called by the Vertec core and specifies the planning period to be displayed as well as the planning interval.

To ensure that a planning table that starts on a new interval (e.g. current month) is not completely empty, the List Controllers also load planning data for an interval before the specified start date.

This means that if August - October is specified as the planning period (planning interval "month "), then row objects are also displayed which have planning data in July, but not in August - October.

### Time tables

The Time tables are based on a single object and display one row per planned opposite object. The columns correspond to the time planning intervals.

Time tables based on a list can also be displayed as read-only summary tables (see chapter 2.7 for this).

We provide the following List Controller classes in the Python module vtcplanning:

| | |
|---|---|
| vtcplanning.SingleObjectTimeTableController | Used to plan on individual objects (`Show for single objects` checkmark is set).<br>The controller itself can handle the different classes (Projekt, ProjektPhase, AbstractWorker, Projektbearbeiter or Planning-Worker), the same controller can always be specified. |
| vtcplanning.ReadonlySingleObjectTimeTableController | Based on a single object, but read-only, not for planning. Used for time tables on single objects that cannot be scheduled. |
| vtcplanning.ReadonlyProjectsSingleObjectTimeTableController | For read-only project time tables on individual objects. Used to display project totals if the planning level is phases. |
| vtcplanning.ReadonlyPhasesSingleObjectTimeTableController | For read-only phase time tables on individual objects. Used to display phase sums if the type of Resource planning is user-phase-link (see chapter 2.15). |
| vtcplanning.ReadonlyContainerTimeTableController | For read-only time tables on lists (`Show for lists` checkmark is set).<br>Used for summary time tables (see 2.7). |

| | |
|---|---|
| vtcplanning.ReadonlyOtherSideContainerTimeTableController | For read-only time tables on lists (`Show for lists` checkmark is set).<br>Used for totals time tables to display the totals of the opposite side on a list of objects (e.g. worker totals on a project list). |
| ReadonlyPhasesContainerTimeTableController | For read-only time tables on lists (`Show for lists` checkmark is set) to display totals of phases on a list of projects or workers when the planning level is project (in this case the normal List Controllers on projects would not be displayed). |

**Pivot tables**

Starting from a list of entries (AbstractWorker, Project or Phases) a Resource planning pivot table can be displayed. The pivot table shows the entries as columns and the planned opposite entries as rows. This makes it possible to enter planning data for new opposite entries via an asterisk line.

For the realization of pivot table views the following List Controller classes are available in the Python module vtcplanning:

| | |
|---|---|
| vtcplanning.RegularPivotTableController | For pivot tables with the entries in the list as rows and the opposite side as columns. Used to plan on a list (`Show for lists` checkmark is set).<br>The controller itself can handle the different classes (Project, ProjectPhase, AbstractWorker, ProjectWorker or PlanningWorker), the same controller can always be specified. |
| vtcplanning.MirroredPivotTableController | For pivot tables with the entries in the list as columns and the opposite side as rows. Used to plan on a list (`Show for lists` checkmark is set).<br>A asterisk line can be displayed in this list to insert new objects to be planned. More detailed information can be found in chapter 2.6.<br>The controller itself can handle the different classes (Projekt, ProjektPhase, AbstractWorker, Projektbearbeiter or PlanningWorker), the same controller can always be specified. |
| vtcplanning.ReadonlyRegularPivotTableController | Used to display project - worker pivot tables when the planning level is `Phases`, with the list entries as rows and the opposite side as columns. |
| vtcplanning.ReadonlyMirroredPivotTableController | Used to display project - worker pivot tables when the planning level is `Phases`, with the list entries as columns and the opposite side as rows. |
| vtcplanning.RegularSingleObjectPivotTableController | Used to plan in pivot tables even on a single project. Shows a pivot table with the single entry as row and the opposite side as columns. |
| vtcplanning.MirroredSingleObjectPivotTableController | Used to plan in pivot tables even on a single project. Shows a pivot table with the single entry as column and the opposite side as rows. |

### Available Renderers

To match this, also in the vtcplanning Python module, there are the following Custom Renderers for use in the Resource planning tables:

| | |
|---|---|
| vtcplanning.PlannedMinutesRenderer | Renderer for the planning times. Enables the input of the plan values in the individual cells.<br>If no plan values are available, 0 is returned. |
| vtcplanning.NetCapacityRenderer | Renderer to display the net availability. Only on user rows with dynamic columns (see 16.1).<br>If a capacity renderer is used in a non-dynamic column, in a table with phases or projects as rows, a corresponding error is thrown. |
| vtcplanning.GrossCapacityRenderer | Renderer to display the gross availability. Only on user rows with dynamic columns (see 16.1).<br>If a capacity renderer is used in a non-dynamic column, in a table with phases or projects as rows, a corresponding error is thrown. |
| vtcplanning.RemainingCapacityRenderer | Renderer to display the remaining availability. Only on user rows with dynamic columns (see 16.1).<br>If a capacity renderer is used in a non-dynamic column, in a table with phases or projects as rows, a corresponding error is thrown. |

These then implicitly behave correctly depending on whether you use them in static or dynamic columns.

### vtcplanning as Stub File

The vtcplanning module is also available as a Python stub file. It is called **vtcplanning.py** and is stored like the other stub files in the subfolder `PythonStubs` in the Vertec installation directory.

**More Information** For more information about the included Python stub files, see the KB article at www.vertec.com/kb/pythonstubfiles/.

## 18.3 Python functions for merging PDF documents

Line: Expert | Module: PSA | Version: 6.5.0.7
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

There are three new features for merging PDF documents:

- **pdfcombine**: This method merges 2 PDF documents. The PDF documents are passed as a byte string.
  Example: `newpdf = vtcapp.pdfcombine(pdf1, pdf2)`
- **pdfextract**: This method extracts one or more pages. The parameters pagefrom and pagetill correspond to the first and last page respectively and must be specified.
  Example: `newpdf = vtcapp.pdfextract(pdf1, pagefrom, pagetill)`
- **pdfpagecount**: This method specifies the page count of the document.
  Example: `pages = vtcapp.pdfpagecount(pdf1)`

## 18.4 Python method for converting .msg files to MIME format

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python method `vtcapp.msgtomime()` converts `.msg` files to **MIME format** so that e-mails (activities) can be displayed directly in Vertec.

Example Code:

```
import os

for act in argobject.eintraege:
    if act.effpfad and os.path.exists(act.effpfad):
        try:
            f = open(act.effpfad, 'rb')
            act.content = vtcapp.msgtomime(f.read())
        except Exception as e:
            print('error on activity %s: %s' % (act, e))
        else:
            print('path is empty or doesn\'t exist on activity %s' % act)
```

**Further information** For more information about this Python method, see the **Vertec Python Functions** article in the Online Knowledge Base at www.vertec.com/kb/pythonfunktionen.

## 18.5 Python function for sending e-mails from Vertec

Line: Expert | Module: PSA | Version: 6.5.0.20
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The new method `vtcapp.sendmail()` enables sending e-mails from Vertec.

In the Cloud Subscription, 1000 emails per month and customer are available free of charge via a mailgun account provided by Vertec. For on-premises customers, or if a separate SMTP server is to be used in the Cloud Subscription, this can be set up in the new **E-Mail section** in the system settings.

Example Code:

```
vtcapp.sendmail(to, subject, body, [cc, bcc, fromSender, attachments])
```

Als Body kann Text als String eingetragen oder ein Word-Dokument verwendet werden.

**Weitere Informationen** Alle Informationen zur neuen Python Methode **sendmail()** finden Sie im Artikel **Vertec Python Funktionen** in der Online Knowledge Base unter https://www.vertec.com/kb/pythonfunktionen.

## 18.6 Word document as body in the e-mail templates

Line: Expert | Module: PSA | Version: 6.5.0.21
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python functions `createoutlookmail()` and `sendmail()` of the module **vtcapp** can be used to specify a Word document instead of a string for the argument **body** as of this version:

1. The corresponding Word document must be created and read:

```
in_file = open("C:\<yourPath>\ExampleWordForEmail.docx", "rb")
dataFileData = in_file.read()
```

2. Generate the e-mail with the method:

```
vtcapp.createoutlookmail('doku@vertec.com', 'E-Mail from Word', dataFileData)
```

It is also possible to use the output of an Office Report (previously Extended Office Report) and pass it as a body:

```
templ = vtcapp.evalocl("bericht->select(eintragid='BerichtEmailBody')->first")
body, activity = vtcapp.executereport2(argobject, None, templ)
vtcapp.createoutlookmail('doku@vertec.com', 'E-Mail from Word', body)
```

**Further Information** For all information, see createoutlookmail and sendmail in the **Vertec Python Functions** article in the Online Knowledge Base at www.vertec.com/kb/pythonfunktionen/.

## 18.7 Python method for Word to PDF conversion

Line: Expert | Module: PSA | Version: 6.5.0.22
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The new Python method `convertwordtopdf()` converts a Word document of type **.docx** into a PDF.

The corresponding Word document must be created and read and can then be converted to a PDF file.

**Further information** For all information, see convertwordtopdf in the **Vertec Python Functions** article in the Online Knowledge Base at www.vertec.com/kb/pythonfunktionen/.

## 18.8 wordtopfd: Table displayed correctly

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The `wordtopdf()` method described in 18.7 had the problem that tables were not displayed correctly everywhere due to a bug in the third-party component used. For example, the QR number part of invoices was not displayed correctly (table did not go to the end of the PDF page, an additional page was printed). Also shapes in the footer were not displayed correctly.

These problems were fixed with the present version 6.6 by an update of the used third-party component.

## 18.9 Support for 'with vtcapp.SystemContext()'

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Currently in Python you have to work with `try` ... finally if you use the SystemContext:

```
try:
    vtcapp.beginsystemcontext()
    ...Code that benefits from the SystemContext.
finally:
    vtcapp.endsystemcontext()
```

Alternatively, it is now also possible to use a `with` statement:

```
with vtcapp.SystemContext():
    ... Code that benefits from the SystemContext.
```

For this purpose, there is a new method `vtcapp.SystemContext()` which returns a Python object `SystemContext`.

- SystemContext is started correctly when "entering" the with block.
- SystemContext is terminated correctly when "exiting" the with block, both when exiting properly and when errors occur.

The method `vtcapp.SystemContext()` must always be called with `with`. If the method is called without `with`, nothing happens.

**Backwards Compatibility**
The old spelling still works, but should no longer be used. We recommend using the new spelling everywhere instead.

## 18.10 Classification of scripts

Line: Expert | Module: PSA | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

For script entries there is a new field **Platform**:

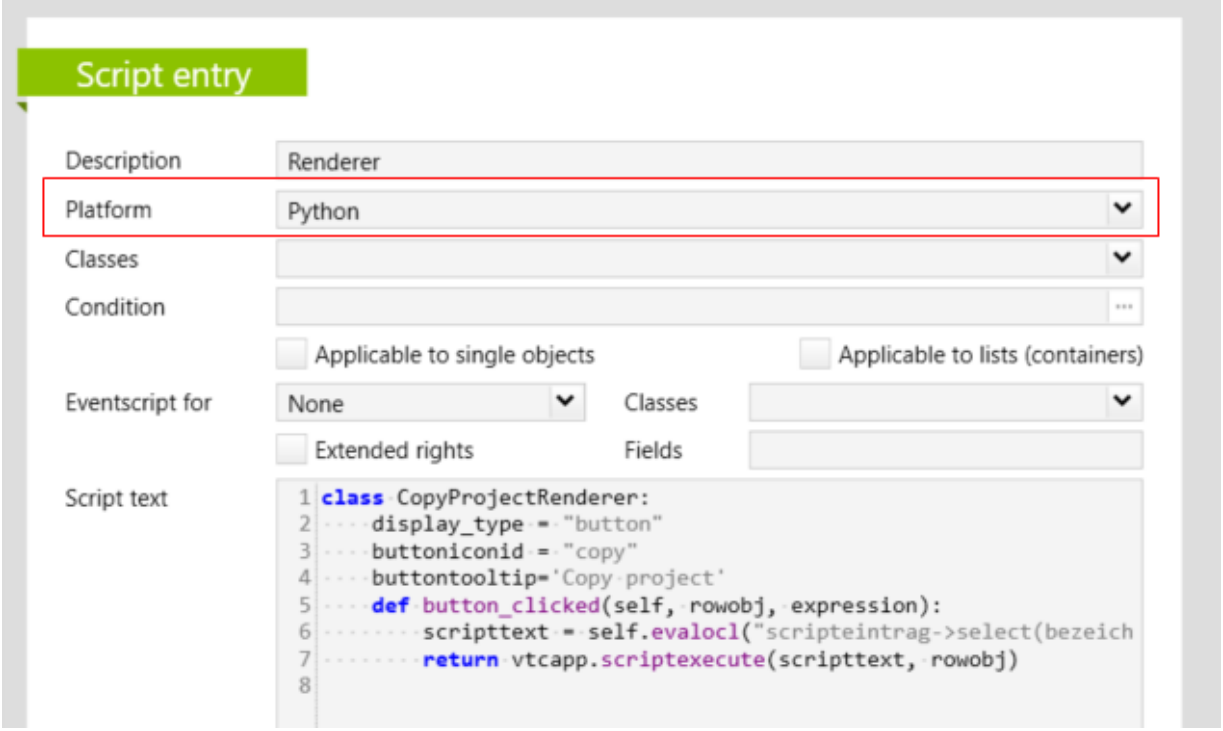



Figure 45 Registration of a Python script

This field is now used to specify whether the script is a Python script or a VBScript.

When executing the scripts, a distinction is then made according to the platform setting as to whether VB or Python is being executed.

The syntax highlighting of Python code as well as the activation of the `Script Editor...` button is now also done via this setting and no longer by the automatic recognition of the `#` as the first character in the script.

**Backwards compatibility**
The value is set automatically during the update. All scripts that start with a `#` are marked as `Python` scripts. All others as `VBScripts`.

## 18.11 Reading files via requestfilefromclient() with confirmation dialog

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Reading a file from the client via Python method `vtcapp.requestfilefromclient()` now always shows a dialog, which informs the user that Vertec wants to read this file, which can be acknowledged with **Yes** or **No**. If the user chooses **No**, the process is aborted and an exception is thrown.

## 18.12 Saving files via sendfile() with confirmation dialog

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

Saving files to the client machine via `vtcapp.sendfile()` without displaying the file selection dialog is now only possible for the file types (extensions) of the following whitelist:

.doc, .docx, .xls, .xlsx, .pdf, .csv, .txt, .zip, .png, .jpg, .jpeg, .bmp, .gif, .eml und .ics.

An error message appears for all other file types.

## 18.13 Restricting the execution of files on the client

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python method `vtcapp.executefile()` can be used to execute an existing file on the client, for example to call an app for a third-party product like a DMS from Vertec. For file types on the **Whitelist** (see above) this still works as before. For all others, a dialog appears asking whether Vertec is allowed to open the file.

– If **Yes**, the path and name of the file will be remembered and the dialogue will no longer appear when this file is called up again. This is done by an entry in the registry in the key: `HKEY_CURRENT_USER\Software\ExecuteFileWhiteList`. The whitelisting refers only to the executable file and its path, not to any arguments.
– If **No**, there is no write access to this part of the registry and the file cannot be remembered. A warning is written in the log file (Vertec.Cloud.log or Vertec.Desktop.log).

## 18.14 Optimization of data transfer via sendfile() method

Line: Expert | Module: PSA | Version: 6.5.0.11
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python function `vtcapp.sendfile()` can be used to transfer a file from the server to a client. With larger files (> 10MB) there could be interruptions due to timeout problems. This has been fixed.

## 18.15 Optimization of data transfer via requestfilefromclient()

Line: Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

With the Python function `vtcapp.requestfilefromclient()` a local file can be selected and processed in the client application. The data transfer has been optimized by transferring the data in packets.

## 18.16 Additional argument for Python method createoutlookmail()

Line: Expert | Module: PSA | Version: 6.5.0.22
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

The Python function `vtcapp.createoutlookmail` has been extended by the optional argument `onBehalfOf`. This allows to send e-mails with different sender.

The following example opens an e-mail with the sender noreply@vertec.com:

```
vtcapp.createoutlookmail("dokumentation@vertec.com", "Considering XYZ", onBehalfOf="noreply@vertec.com")
```

**Further information** For all information, see createoutlookmail in the **Vertec Python Funktions** article in the Online Knowledge Base at www.vertec.com/kb/pythonfunktionen/.

## 18.17 New functions of the Python module "ziputils

Line: Expert | Module: PSA | Version: 6.5.0.14
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

ZIP files can not only be created but also extracted. For this purpose the Python module **ziputils** has the following 3 new functions:

- `readnames(zipcontent)` returns a list of the file names contained in the ZIP file. If the ZIP file has a folder structure, the names contain the paths separated with slashes (/).
- `readbyname(zipcontent, name)` reads a file from the ZIP file based on its name.
- `readbyidx(zipcontent, idx)` reads a file based on its index in the name list.

**Further information** For more information about the "ziputils" module, see the **Vertec Python Funktions** article in the Online Knowledge Base at www.vertec.com/kb/pythonfunktionen.

# 19 Security

## 19.1 Authentication via API tokens for Web API accesses

Line: Standard, Expert | Module: Leistung & CRM, Business Intelligence | Version: 6.6
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

With version 6.6, **API Tokens** are introduced to increase the login security of Web API accesses. The API tokens increase the login security of the Vertec XML interface and the BI API interface.

---

Note For backwards compatibility, the old system (authentication via username and password) is still supported. However, as of **Vertec version 6.7 it is no longer supported**. The changeover should therefore still take place soon.

---

### Generating API Token

API tokens can be generated on the Further Information page of a user:

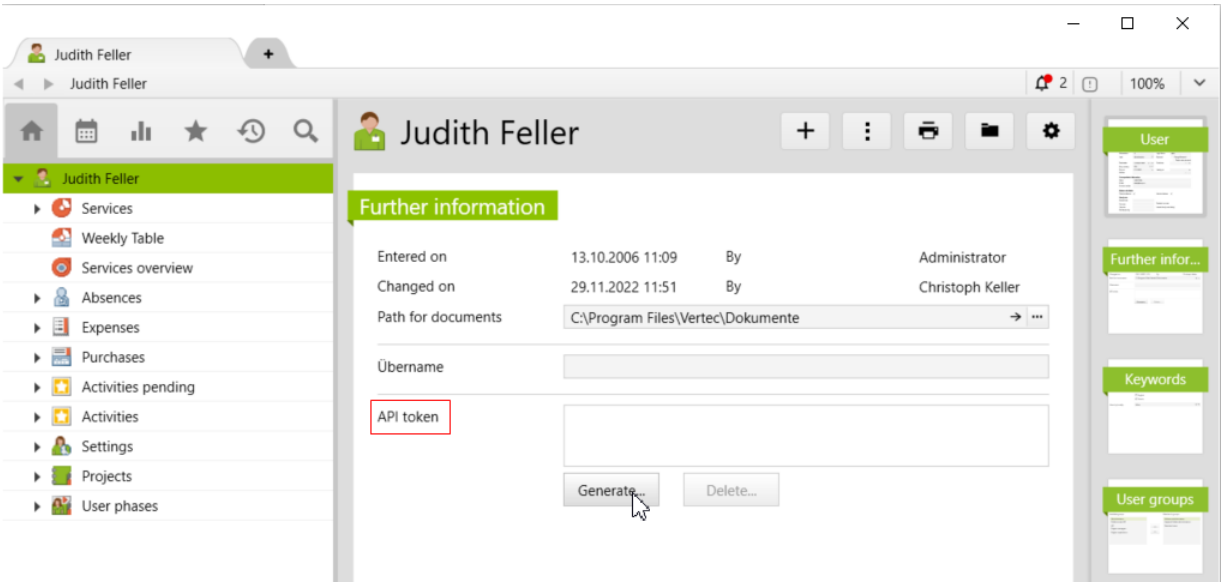



Figure 46 Generating API tokens

With a click on the `Generate...` button, the API token is generated and displayed in the corresponding field:

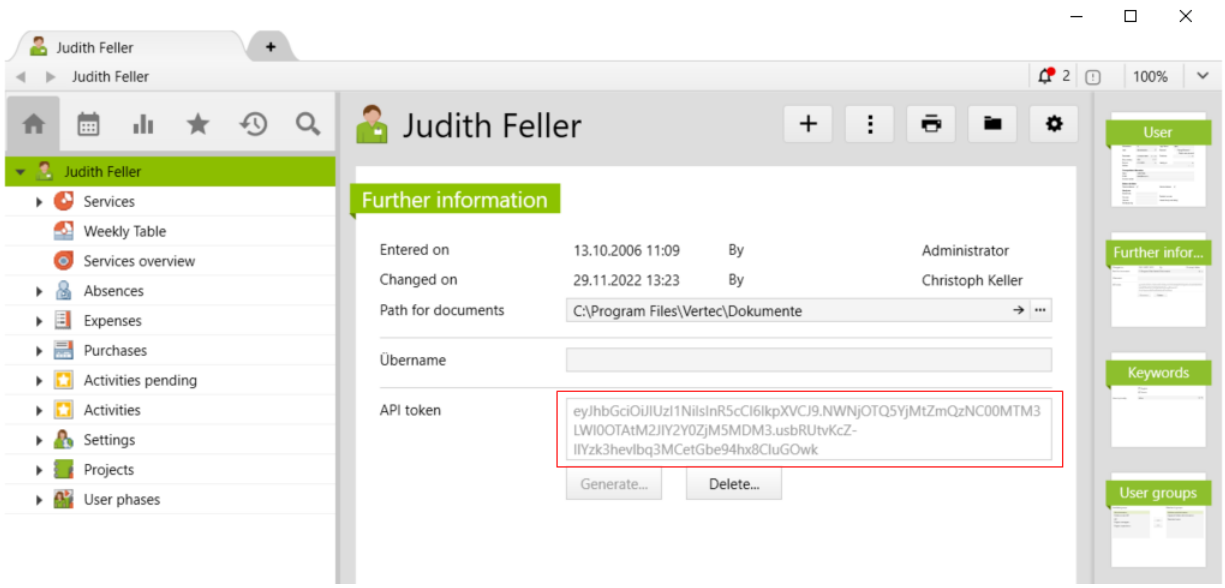



Figure 47 The API token can be copied

This must be copied immediately, because it will no longer be displayed the next time the page is called. Instead of the token the following text is displayed and the token can only be deleted:

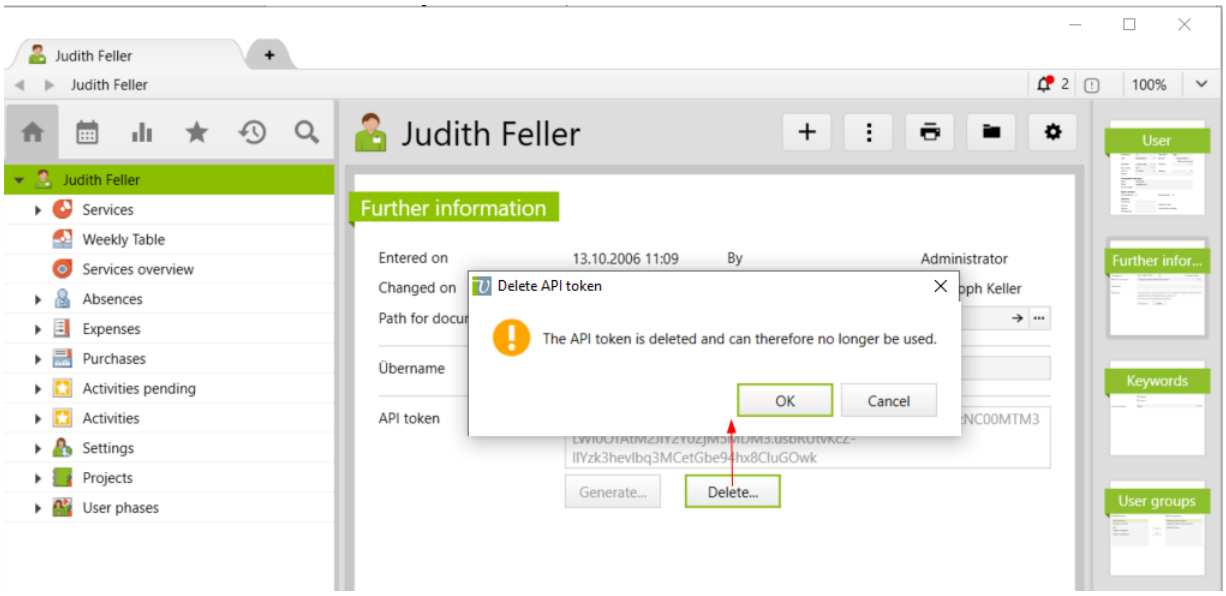



Figure 48 An API token is deleted

If the token is deleted, it cannot be used afterwards and instead of the `Delete...` button the `Generate...` button is available again.

### Authentication via API token at the XML server

Authentication via API token is done at the XML server by passing the parameter `api_token` instead of `vertec_username` and `password`. Example:

```
import requests
url = 'http://localhost:8081/xml'
api_token =
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.MGVhZmUzMzYtNmVhMi00MDdhLTgxNjQtZDYxZmI0NzU2M
WZi._r16YlvWmZCMJ3qdDX3bK5_DJHwcczTYaWoKUYUNZuk'
headers = {'Authorization': 'Bearer %s' % api_token}

xmlquery = """
<Envelope>
  <Body>
    <Query>
      <Selection>
        <ocl>projektbearbeiter</ocl>
      </Selection>
    </Query>
  </Body>
</Envelope>
"""

response = requests.post(url, headers=headers, data=xmlquery)
print(response.text)
```

There is no need to call `/xml/auth` anymore, access is directly to `/xml`.

Also, new sessions are no longer continuously created. The same session is always used within the timeout interval.

### Starting Multiple Sessions

If a new (parallel) session is to be used explicitly, an HTTP header `VertecSessionTag` with an arbitrary value (e.g. 1) can optionally be included. A maximum of 36 characters is allowed. Per `VertecSessionTag` a new session is started. Thus, the session affinity is based on the combination of API token and session id.

Example:

```
headers = {'Authorization': 'Bearer %s' % api_token, 'VertecSessionTag': '1'}
```

### Authentication via API Token at BI API

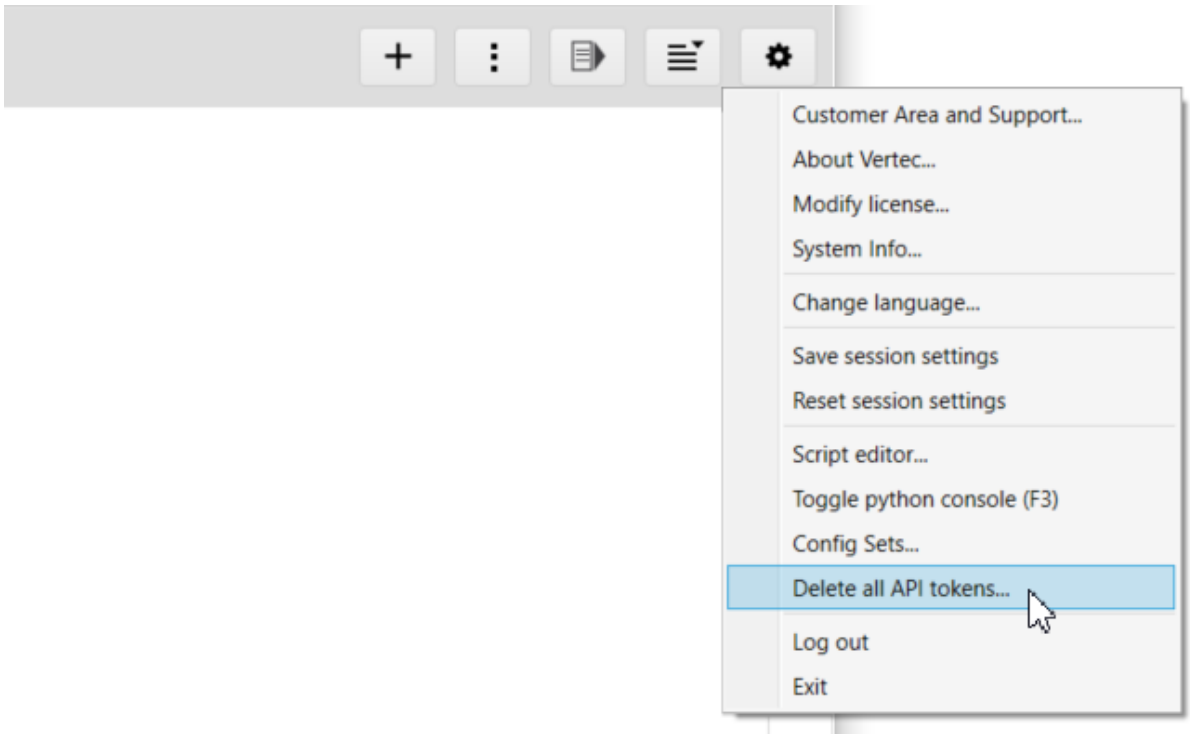
For the BI API, the API token must be specified as a `bearer` token in an `authorization` header:

```
import requests
url = 'http://localhost:8081/api/bi/measures'
api_token =
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.MGVhZmUzMzYtNmVhMi00MDdhLTgxNjQtZDYxZmI0NzU2M
WZi._r16YlvWmZCMJ3qdDX3bK5_DJHwcczTYaWoKUYUNZuk'
headers = {'Authorization': 'Bearer %s' % api_token}
r = requests.get(url, headers=headers)
measures = r.text
```

If both API token and username/password are specified in a request, only the API token is considered. If it is invalid, a login attempt with username/password will not be made automatically.

### Delete all API tokens

Users with administrator privileges can delete the generated API tokens of all users in the Settings menu using the `Delete all API tokens...` option at once:




First, a warning query appears asking again whether all API tokens should really be deleted.

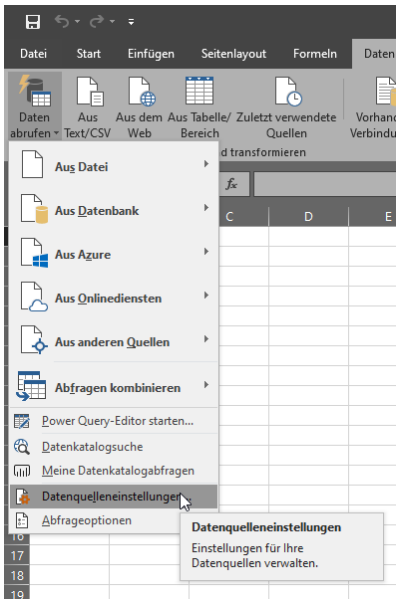
### Authorizations

The **API Token** section on the Further Information page of the user is displayed only if the logged-in user has the **administrator privilege** or is granted the **write privilege** to `Projektbarbeiter.ApiToken`.

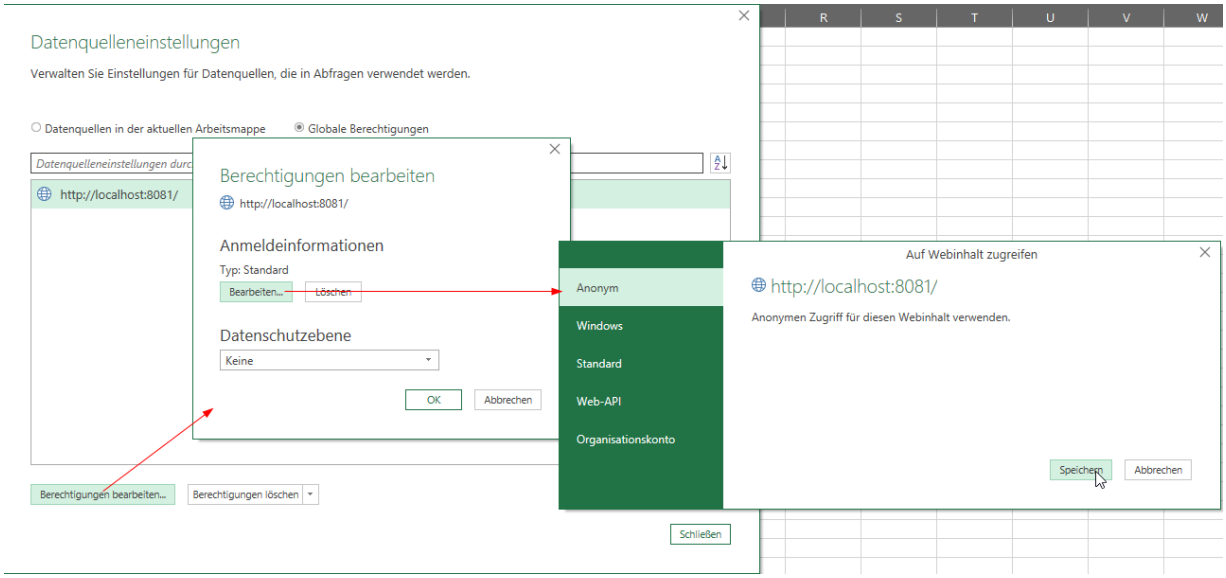
### Authorization-Header in Excel

If the data is loaded into Excel as in the sample application on the Knowledge Base (www.vertec.com/kb/bi-api-beispielanwendung-excel), then the credentials are set as follows:

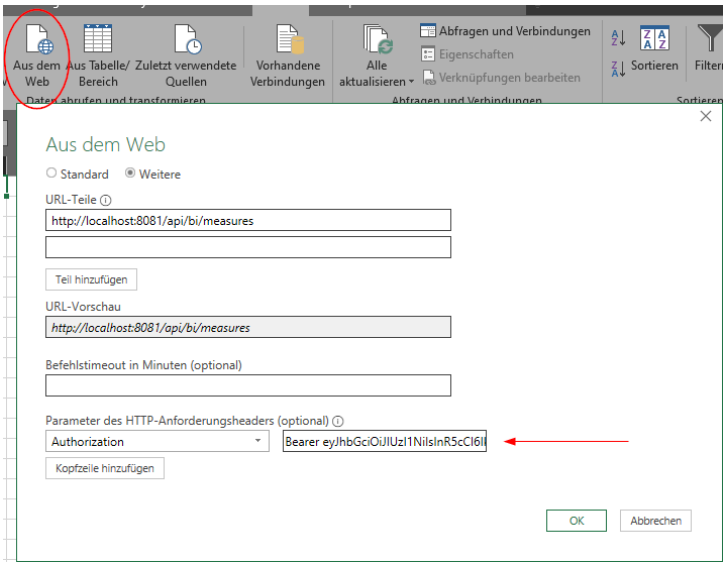
1. Open the data source settings:




2. Select data source > edit permissions > edit credentials > anonymous:




3. Data from the Web > Other > Set parameters of the HTTM request header:




## 19.2 2FA Secrets are regenerated after deactivation

Line: Standard, Expert | Module: PSA | Version: 6.5.0.5
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

When changing the 2FA system setting `Use 2 factors for cloud clients` to `False`, all existing secrets on users are deleted. When the setting is changed back to `True`, users will be prompted to re-set up with QR code.

## 19.3 Waiver of thumbprint for secure LDAP connections

Line: Standard, Expert | Module: PSA | Version: 6.5.0.16
Mode of Operation: Cloud Subscription / On-Premises | Apps: Full-featured

A thumbprint is only necessary for self-signed certificates. For secure connections like LDAP servers, which use a regular security certificate, the thumbprint (LDAPS certificate fingerprint) does not have to be entered anymore.

---

**Further information** For more information about LDAPS certificate fingerprinting, see the **System Settings Authentication** article in the Online Knowledge Base at www.vertec.com/kb/sektionauthentisierung.

---

## 19.4 COM login method removed

Line: Expert | Module: PSA | Version: 6.5.0.15
Mode of Operation: On-Premises | Apps: Full-featured

The COM method `.login()` has been removed for security reasons, as it cannot take 2FA into account.

# 20 Updating Vertec On-Premises Installations

## 20.1 New installation of Vertec

For a new installation, use the VertecSetup-6.6.0.exe installation program available for download. For more information on installation, see our Online Knowledge Base at www.vertec.com/kb/neuinstallation.

## 20.2 Before the update to 6.6

Please refer to the backward compatibility sections in the chapters as well as section 1.2.

## 20.3 Update von Vertec

To update a Vertec installation, also use the installation program. For larger customer installations with many customer-specific reports and list settings, we recommend that you first set up a test installation and run through the release change before performing the update on the active system. For more information, see the **Test Installations** article in our Online Knowledge Base at www.vertec.com/kb/testinstallation.

## 20.4 The first startup after conversion

After the data conversion, the Vertec Desktop App must be started. This first start after the conversion is important and is also part of the conversion, as various adjustments to data and structures, etc, are only carried out then. It is important that this first start runs smoothly. Under no circumstances should the first startup process be aborted after a conversion.

**Restarting the Cloud Server**

After the conversion and the initial start of the Vertec Desktop App, the Cloud Server must be restarted. This completes the update of Vertec.