Planisware Author White Papers

Using Formulas in Planisware

By Planisware

085_USING_FORMULAS_AUTH_EN_6.1.3_A

Contents

Using Formulas in Planisware

About this Guide

This white paper contains information about how formulas can be built in Planisware, with a particular focus on how the available functions can be used.

Introduction to Formulas in Planisware on page 2 introduces the important concepts about how formulas can be built in Planisware.

Buildings Formulas with the Help Dialog on page 11 introduces how you can use the formula help dialog to build formulas from the most commonly used elements and explains the concepts surrounding these elements.

Special Information for Date and Duration Formulas on page 16 then introduces the concepts you will need to be aware of in particular for the use of date and duration type formulas.

A number of sections then describe the various functions you can use within Planisware formulas. These are organized into types and within these types they are sorted alphabetically. The sections for these types are:

- Date and Duration Functions on page 19.
- Breakdown Structure Functions on page 28.
- Type Conversion Functions on page 30.
- String Treatment Functions on page 36.
- List Treatment Functions on page 45.
- Numerical Functions on page 55.
- Data Retrieval Functions on page 58.
- Baseline (Reference) Treatment Functions on page 68.
- Formula Evaluation Functions on page 73.

Finally, Global Variables on page 76 describes the global variables that can be added to Planisware formulas.

To navigate through the document, you can:

- Use the table of contents or navigation pane in the PDF to search by title. The functions are organized into categories that describe how the functions can be used correctly.
- Search for a function by using Ctrl + F on its English name, French name, description, and so on.

Version Information

The version of this document is: 085_USING_FORMULAS_AUTH_EN_6.1.3_A. The release date for this document is 5th of April 2016. For any information on the guide, any questions about the available versions, or for any feedback on the contents of this guide, please contact support@Planisware.com. Please use the version number of this guide as a reference when contacting Planisware.

Planisware reserves the right to modify the information contained in this document without prior notification. Any modifications to the documentation will be announced in a newsletter from Planisware documentation. To receive the newsletter, send an email requesting subscription to the above address.

Note: This document was written for Planisware version 6.1.3.

Introduction to Formulas in Planisware

Formulas in Planisware can be used to return different types of values using object attributes, logical operators, functions, and constant values (such as strings or numbers).

They are used to:

- Filter objects displayed in list tables.
- Filter objects treated by macros, alerts, and so on.
- Create new values from a combination of other values, to be displayed like other attributes, or to modify existing values.
- Convert values of one data type into others for further calculations.
- Produce, sort, and find data in lists of data.

Types of Data

Formulas can contain and return values of the following types:

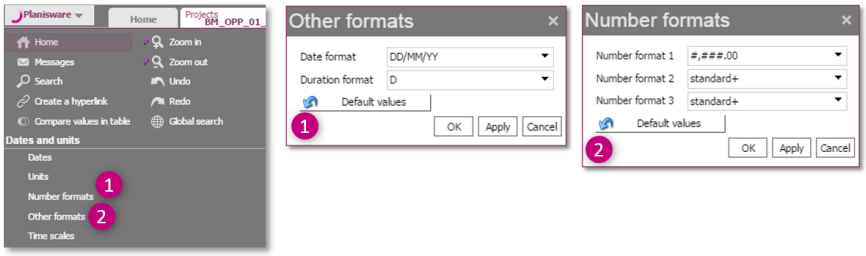
Type	Value	Default value	Encapsulation
String	"0-9 a-z & @ <>, * ?"	""	""
Number	One or several figures, eg; 345	0	
Date	'DD/MM/YY ' if DD/MM/YY is the default format for dates, eg '09/12/08'	-1	'
Duration	'DDdHHhMMm' if DDjdHhMMm is the default format for durations; eg '30d7h30m'	0	'
Boolean	YES, NO, TRUE, FALSE	FALSE	

Note: "Filter" is the word used to describe a formula that returns a Boolean value. If an objects returns YES to a filter, it will be displayed, if not, it will be hidden.

Setting Default Formats

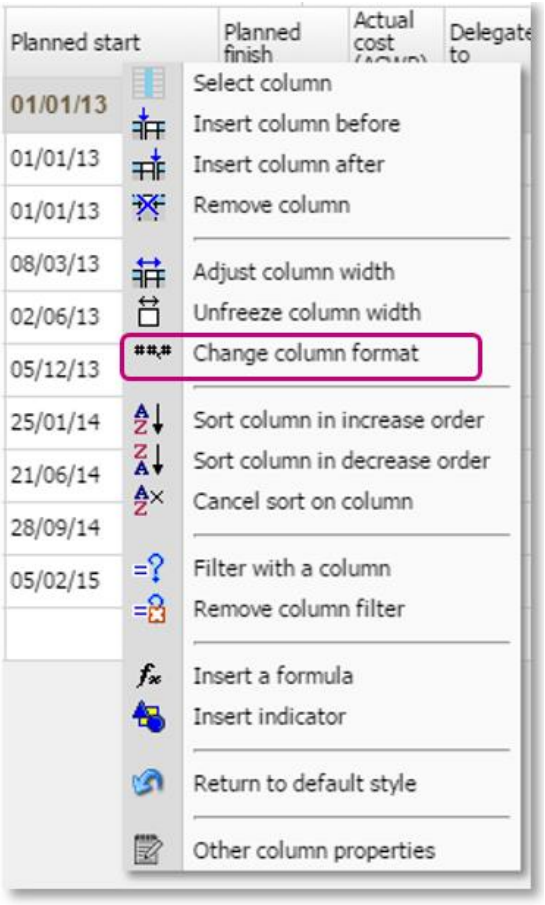
Different types of data in Planisware can be shown in different ways on the screen as you could do on a standard spreadsheet application. When manipulating formats, **always keep in mind that the displayed value does not necessarily match the value stored in the system.**

To customize the default displayed formats, use the dialogs available from the **Number formats** and **Other formats** commands on the **Planisware** main menu.



Dialogs for changing formats

You can also choose a specific format for one column by right-clicking it:



Change column format command

As an example, you may choose the following date format: `DD/MM/YYYY HH:MM:SS`.

In this case, the hour displayed (8am) does not match the stored value in the database, but comes from a parameter stating that a workday starts at 8. Calculations based on this displayed hour will therefore not work as desired.

Note: It is therefore crucial to understand how Planisware manages units before doing calculations on them. Please refer to Special Information for Date and Duration Formulas on page 16 to learn more on this topic.

Syntax Rules: Use of Quotation Marks

How arguments are written inside a formula depend on the argument type. For instance, if the first argument of a formula is of type string, it should be written with quotation marks. An example with the function `ENDSUBSTRING` (used to extract a substring from a string:

```
ENDSUBSTRING("Hello, World !",6,2)>" World"
```

However, if we replace the first argument by the name of an **attribute that points to a string**, quotation marks should not be used:

```
ENDSUBSTRING(DESC,6,2)>"Task"
```

If quotes were used in this example, the formula would truncate the string DESC and return `"DE"` instead of understanding that we point to the description attribute of the object.

Note: The same rule applies to dates and duration formats which are written between ‘’.

Case of the evaluation of a filter

Some formulas evaluate another formula inside one of their arguments. This is the case of `EVALUATE_STRING`, `LIST_COLLECT`, and many others. This formula is encapsulated as a string and must therefore be written between quotation marks. If the filter itself contains quotation marks a `\` should be inserted before each inner quotation mark, to avoid interpretation errors.

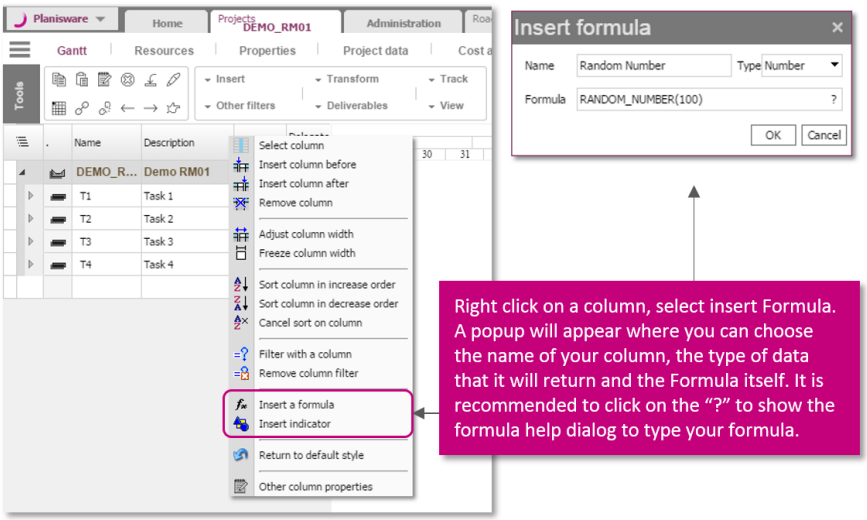
The example below evaluates a formula that returns the concatenation of two attributes separated by a string:

```
EVALUATE_STRING("DESC + \" - \" + LOCATION ")> "Consultant - JAPAN"
```

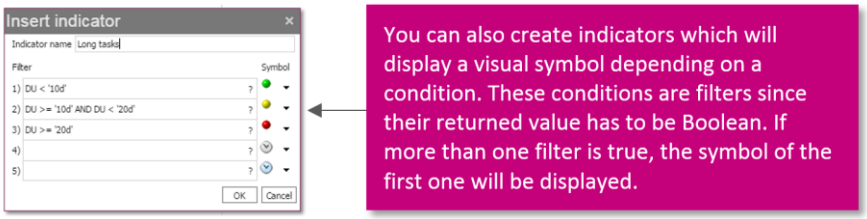
Using Formulas in Planisware Web Environment

Formulas can be used in many different contexts inside of Planisware. This section will briefly list where and how formulas can be used inside of Planisware Web environment.

From any table view of Planisware, a column can be added to insert a formula that will return a value based on the object class displayed in the table. In the following image, an activity list table is used.



Inserting formula



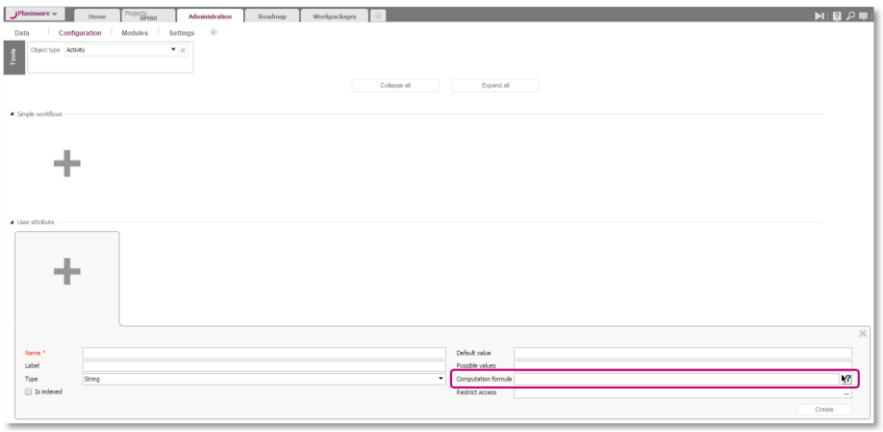
Formulas in indicators

Note: These types of formulas and indicators are not permanent and will disappear when clicking **Return to default style**. To create permanent formulas, see Creating Named Formulas on page 7 or create a user attribute in the administration mode.

In Administration Module

In the **Configuration** page, in the administration module, you can create various objects in which formulas will be used, in any fields containing the icon that links to the formula help dialog. You can refer to the E-learning content for the administration module, for more information. The following list summarizes where formulas can be used, in this module:

- **Alerts**: enable you to specify the display of messages to users when certain conditions are triggered.
- **Data management rules**: or data consistency rules enable the administrator to filter the values available in a list.
- **Locks**: enables you to block the modification of specified attributes on an object class or block the deletion of specific objects.
- **User attributes**: enables the administrator to add attributes to a specific object class. If a computation formula is entered, the user attribute will be equivalent to a named formula created in Planisware.



Computation formula on user attribute

In Planisware Explorer Module

Planisware Explorer is a powerful module that allows users to visualize data in different views and charts, such as table views, bar charts, graphs, bubble charts, and so on, by building queries. You can refer to the E-learning content for this module for more information.

Formulas can be used in the following situations inside of Planisware Explorer to build queries and dashboards.

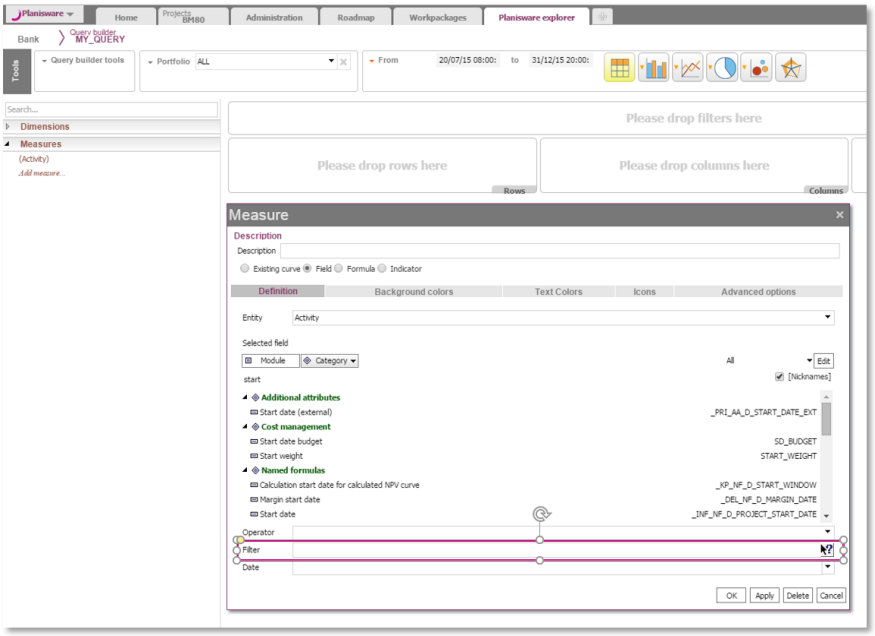
In the query builder:

- **Measures**: if you create a measure based on a field, you can filter this field based on a formula.

- **Filters**: the data displayed in the query can be filtered by a formula applied to a dimension.

In the dashboard builder:

- **Widgets**: you can insert a formula widget that will directly print the formula's returned value in the dashboard.
- **Filters**: in addition to the filters created inside of the query, filters can be added in the dashboard to reduce data to be displayed.

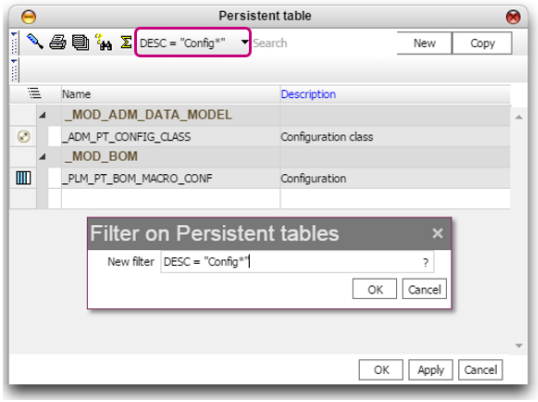


Example of filter in measure

Using Formulas in Pro Web

Administrators who have access to Pro Web can use formulas for multiple purposes (in configuration level 2). Following is a non-exhaustive list of major uses:

- **Symbolic fields**: equivalent to indicators, symbolic fields are persistent. They can be created via **Data > Data model configuration > Symbolic field**.
- In reports as **attributes**: reports are environment objects used to create the body of the intranet graphical interface. They can be used as headers, main body reports, and popup dialogs. Cells with input mode can have a filter applied to the class displayed in the drop-down list. To do this you can double-click the attribute cell, and go to value choice filter, under **Input mode** in the dialog.
- In reports, in **Lines**: can have a filter under **Print criteria** to hide a whole line of attributes under certain conditions.
- In reports, in **Blocs**: can have a filter under **Iteration block definition** to filter the data that is displayed for the class.
- In reports, as **Display condition**: a report itself can have a display condition added (using the icon available under **Input mode**) to specify that the whole report is hidden under some circumstances.
- **Filters**: any table displaying an object class can be filtered by clicking on the button highlighted in the following image. Here is an example on persistent tables.



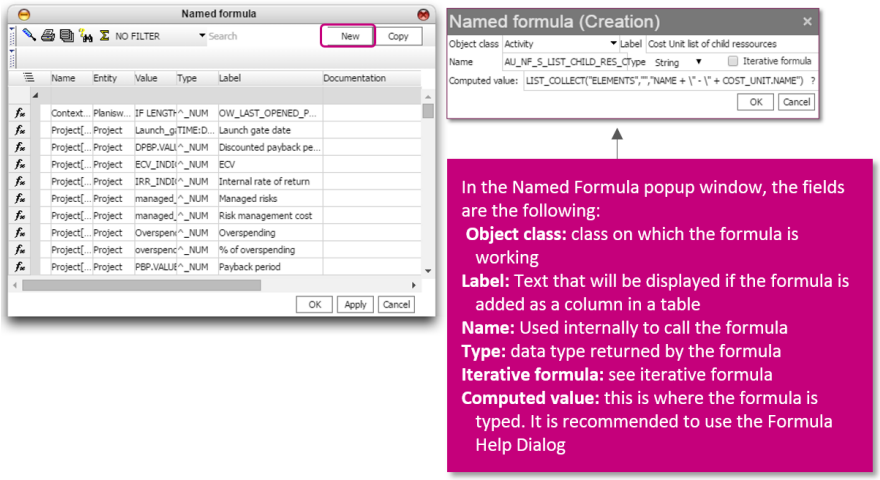
Example of filter on list table

- **Named Formulas**: equivalent to inserting a formula in a table or creating user attributes with computed value, they are persistent. See Creating Named Formulas on page 7 for more information.

Creating Named Formulas

Named formulas enable you to store the formulas that you create as environment objects so that they can be reused anywhere within the interface. You can also create iterative named formulas as seen in Creating Iterative Named Formulas on page 7.

To create a new named formula or to view existing ones, you can go to **Data > Data model configuration > Named Formulae** (in Pro Web).

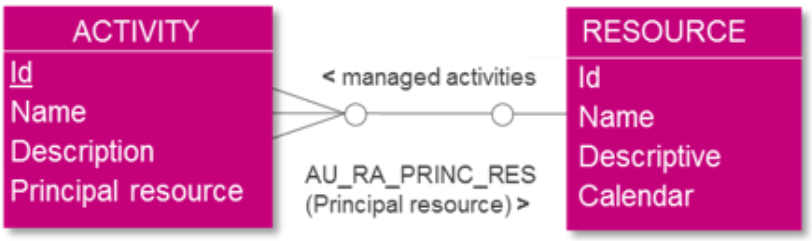


Creating named formula

Creating Iterative Named Formulas

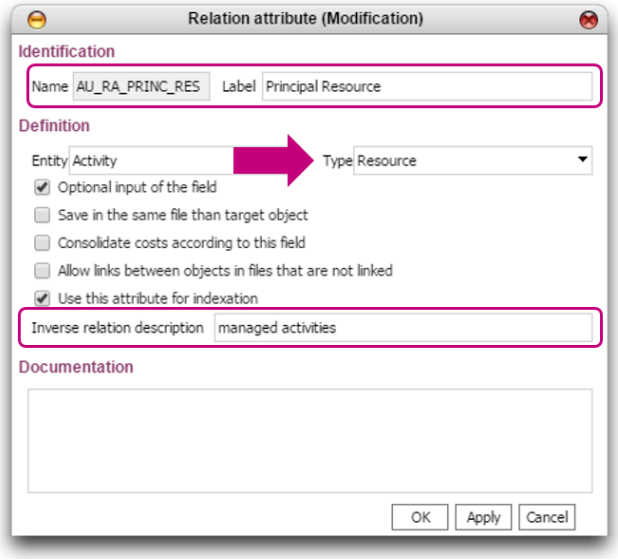
Iterative formulas enable you to return values according to relation attributes between the origin object class and the related object class, in a 1-N relation. Their use implies the knowledge of Planisware data model. To explain this concept, an example will be used.

Suppose that the business needs to be able to assign a resource to any activity and call it `Principal resource`. As the data model diagram below shows, a 1 – N relation attribute is created to link the class ACTIVITY to the class RESOURCE. An activity can have only one principal resource while a resource can manage multiple activities.



Relation created between activity and resource

The following image shows the attribute sheet for the definition of the relation attribute. It defines a directional relation from activities (Primary object) to resources (secondary object). The **Inverse relation description** is the name that will be used to navigate the relation in the opposite direction.

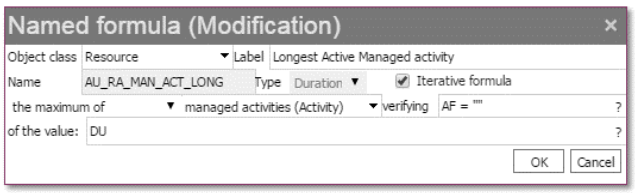


Relation created between activity and resource

You can add this relation attribute to an activity list table as a column named **Principal resource**.

The relation being set, now suppose that you want to recuperate the duration of the longest active (not finished) task managed by any resource. This means adding a formula on the resource class that goes to activities, keeping in mind that one resource can manage multiple activities.

The following iterative named formula is created:



Iterative named formula example

The fields are filled in as such:

- **Object class**: the object on which the formula works. In this example, resource.
- **Label**: text that will be displayed if the formula is added as a column in a table.
- **Name**: used internally to call the formula.
- **Type**: data type returned by the formula.
- **Iterative formula**: means that the formula will be scanning all objects linked to the object class. In the example, it will scan all activities linked to the resource.

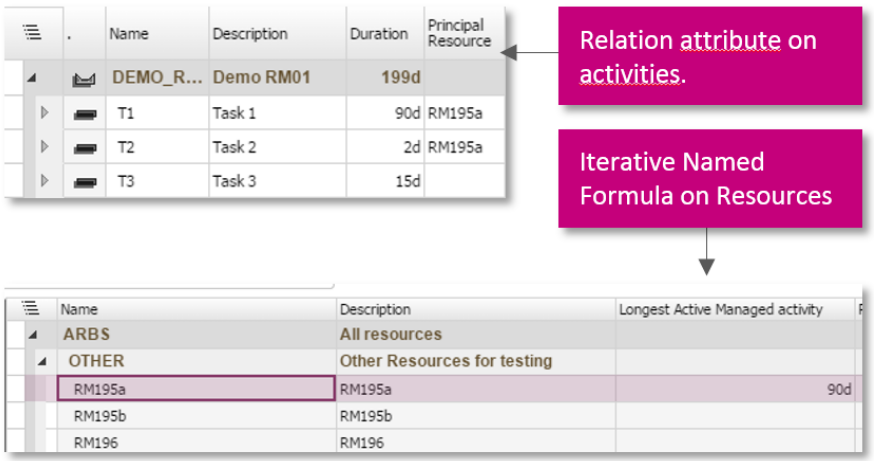
- **The maximum of**: this is a drop-down menu that displays options depending on the type of formula. The example formula is designed to return the maximum of durations of linked activities. Other possible values for this drop-down menu are listed in a table below.
- **Verifying**: filter condition applied to each scanned object. The example formula is designed to only include activities that are not finished yet (AF or Actual finish not filled).
- **Of the value**: formula defining the value that will be returned by the iterative formula.

In summary, this iterative formula will operate on resources, and display for each of them, the duration of the longest ongoing activity (no actual finish entered) linked to it by the inverse relation to `Principal resource`.

Other possible operators based on formula type:

Type	Possible Operators
Boolean	"there are at least one" "There are no" "All the"
String	"The concatenation of the"
Date, End date	"The maximum of" "The minimum of"
Duration & Number	"The sum of" "The average of" "The maximum of" "The minimum of" "The quadratic average of" "The variance of"

The following image shows an example of the result of this iterative formula.



Iterative named formula result

Calling Named Formulas

Named formulas can be displayed in list tables exactly the same as other types of attributes.

Best Practices for Performance

Problem

One of the major risks to performance in Planisware is linked to actions that iteratively scan all objects of an entity (class) when doing actions such as filtering or looking for an object.

These full scans will generate a progressive loss of performances when the number of objects increases.

As an example, the use of the function LIST_COLLECT is not recommended on a class containing a great number of objects.

Typing the following formula anywhere in the application:
`LIST_COLLECT("TASK","DU<'10d'","ID")` will result in the system scanning every single activity in the database to check if the condition `"DU<'10d'"` is verified in order to create a list. This could cause severe performance loss on large database whenever the formula is executed and is therefore not recommended.

Functions Using Indexes

All standard Planisware entities (classes) are indexed by their identifier. Some functions use these indexes to look for an object instead of scanning all the objects from an entity to filter them, which results in much better performances.

For example, the function ?OBJECT_EXISTS (see ?OBJECT_EXISTS on page 58 enables you to check if an object exists using its class and identifier. This function uses indexes.

Similarly, the following functions return the value of an object's attribute when its identifier is known:

- STRING_VALUE(entity, identifier, attribute of type string) (see STRING_VALUE on page 67 for more information).
- DATE_VALUE(entity, identifier, attribute of type date) (see DATE_VALUE on page 63 for more information).
- DURATION_VALUE(entity, identifier, attribute of type duration) (see DURATION_VALUE on page 64 for more information).
- NUMBER_VALUE(entity, identifier, attribute of type number) (see NUMBER_VALUE on page 65 for more information).

Using Conditions on Boolean Values

Using a filter condition on a Boolean attribute is much faster than on a string.

For instance, you might have internally set a rule stating that all WBS elements of a project should contain the letters “WBS” in their description field.

If you later want to create a pie chart in Planisware Explorer that counts the number of WBS elements per project, you have two options:

1. Write a condition that reads the description field of activities such as `DESC="*WBS*"`.
2. Write a condition that reads the Boolean attribute **Is a WBS Element?** such as `?WBS_ELEMENT = TRUE`. (**?WBS_ELEMENT** being the technical name of the attribute **Is a WBS Element?**).

In the first case, the system will have to read the whole description of all activities of all projects in the database in order to find these three letters. In the second case, the system will only have to check if the value of the attribute is true or false, which is a lot less information to process and there is smaller chance of mistakes.

Note: It is always recommended to filter on Boolean values when possible, by searching through the object attributes list to see if one matches the requirement. One should always be aware of the amount and frequency of calculation involved when writing a formula.

Optimizing with FROM and BELONGS

The functions FROM (see FROM on page 30 for more information) and BELONGS (see BELONGS on page 28 for more information) enable you to limit the evaluation of a formula so that it is only evaluated for objects verifying the conditions given in the function, instead of scanning a whole list of objects. This can be used in:

- Filters in Planisware tables.
- Filters in blocs, in reports.
- Filters on load curves, load arrays, crossing matrices.
- Filters on cost tables.
- Filters on cost fields.

Planisware optimizes the scanning by using:

- The identifiers and the associated indexes.
- The relations predefined between objects of the breakdown structures and their children.
- The relations to cost elements (planned hours, actual hours, planned expenditures, actual expenditures, budget lines).

For example, the filter `FROM("SQ/CRP/TLE")`, in a bloc iterating on resources, in a report, will only scan resources that are part of the branch of the breakdown structure `SQ/CRP/TLE`.

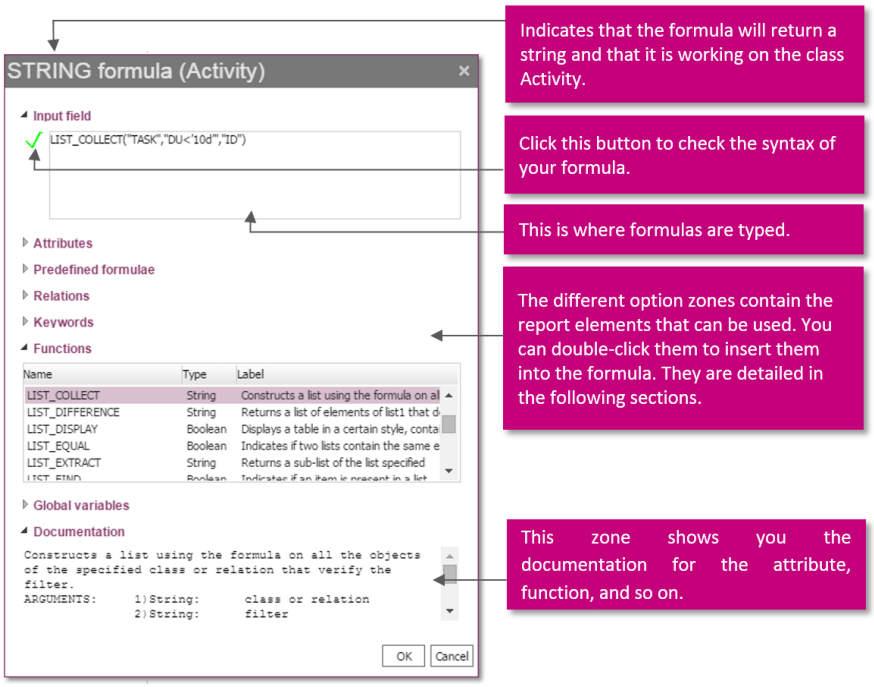
It is possible to use a list of identifiers as arguments for the FROM and BELONGS functions.

Note: Be careful, the optimization will not work in the following cases:

- The formula contains at least a conditional expression (OR) at the same level as the FROM or the BELONGS functions.
- The formula uses IF ... THEN... ELSE with test conditions that are not on the OPX2 Context.
- A formula in the filter calls a named formula containing FROM or BELONGS.
- You must check the option **Consolidate costs according to this field** in a relation attribute, in order to make the BELONGS work (for example, in a relation attribute between project (**Entity**) and Additional_Table_XXX (**Type**).

Buildings Formulas with the Help Dialog

Wherever a formula can be entered in the interface, you will be able to use the formula help dialog. This is shown in the following image:



Formula help dialog

It is important to note that the option zones named **attributes** and **Predefined formulae** will have different content depending on the object class on which the formula is working. The other sections will remain the same no matter what the object class is.

The type of formula (String, Boolean, and so on) will not affect the content of the dialog box but will cause errors if there is a mismatch between the formula returned type and what it should be.

Tip for long formulas: You can use as many spaces and line breaks as you want to write your formula. In the case of a long formula, for example, one with multiple IF statements (see Operators (Keywords) on page 14 for more information on IF clauses) a line break between each statement will help you to read the sentence and identify errors since the error message will indicate the position and line of the error.

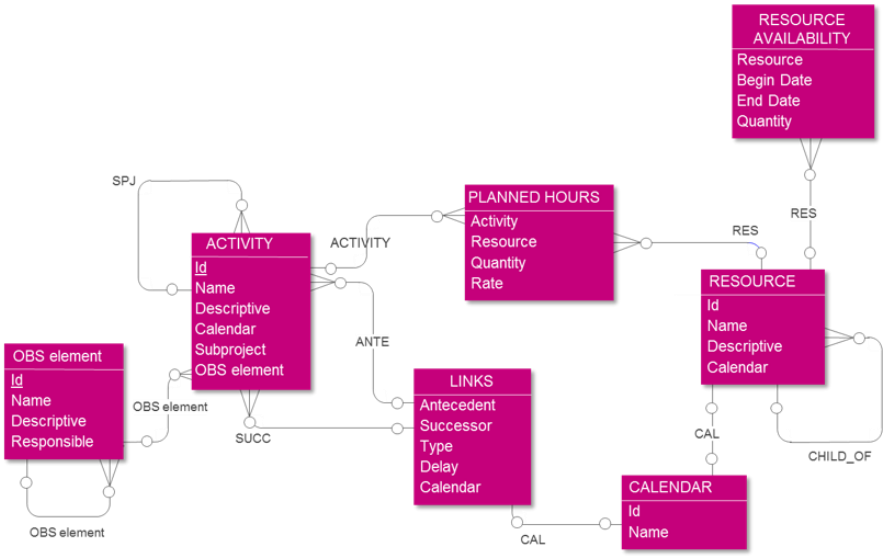
Attributes

Formulas work on a specific object class. For example, when they are used to filter resources, they work on the resource object class. Attributes for this object class can therefore be called within the formulas.

- A formula working on a specific class will also work on its subclasses (for example activity class will work on tasks and WBS elements).
- The attributes that can be used depend on their data type and the data type of the formula.
- Attributes of all types can be used in Boolean fields as long as the complete formula returns a YES or NO value.
- All data types can be used in string formulas.
- Data of one type can be transformed into another type using conversion functions (see Type Conversion Functions on page 30 for more information).

Calling Attributes of Related Objects

The data model can be used in formulas to call attributes of related objects. The diagram below shows a simplified version of the data model.



Simplified data model

You can use the syntax:
`RELATED_OBJECT_CLASS_NAME.ATTRIBUTE_OF_RELATED_OBJECT_CLASS` to call the value of an attribute on the related class. For example, a formula working on planned hours could use the syntax: `ACTIVITY.OBS_ELEMENT` to use the name of the OBS element for the activity, in the formula.

Other examples:

- You are on an activity and you want to retrieve the value of a project attribute project owner. You can use: `PROJECT.OWNER`.
- You are on a planned hours object and you want to retrieve the resource rate. You can use: `RESOURCE.RATE`.

Note: The syntax will only work on relations from the origin object N to 1. For instance, `PROJECT.OWNER` works because the origin object (activity) has a relation to just one project. For the opposite way round, you need to use iterative formulas, as explained in Creating Iterative Named Formulas on page 7.

The list of relations available for an entity can be found in the **Relations** option zone of the formula help dialog.

Predefined Formulae

This zone enables you to call an already existing named formula inside of your formula. The displayed list will contain all the named formula working on the object class in question.

Imagine for example that you want to use an existing Boolean formula to return its value as a full sentence in a string, which can be used in a popup message. The formula would take the following syntax:

```
IF MY_BOOLEAN_FORMULA THEN "The value is true" ELSE "The value is false"
                                                FI
```

This would return the string "The value is true", if the formula `MY_BOOLEAN_FORMULA` is returning YES and "The value is false" if it is returning NO.

Operators (Keywords)

Operators can be used to build the necessary structure into the formula syntax. There are mathematical operators, logical operators, and two special types of operators, used to create IF clauses and IN statements. These are described, in turn, in the following sections.

Available operators

Mathematical Operators

Operator	Description	Used with Data type
+	Sum	Strings, Numbers, Lists, Dates, Durations
-	Subtract	Numbers, Dates, Durations
*	Multiply	Durations and numbers
/	Divide	Durations and numbers
<>	Different than	Strings, dates, durations and numbers
=	Equal	Strings, dates, durations and numbers
>	Greater than	Dates, durations and numbers
>=	Greater than or equal to	Dates, durations and numbers
<	Smaller than	Dates, durations and numbers
<=	Smaller than or equal to	Dates, durations and numbers

Filtering a table display with a formula will result in the concerned column(s) header(s) being displayed in blue to remind the user that data has been filtered. To avoid this, for formulas that should be applied at all times, write the statement between brackets, for example: `[DESC ="*Coa*"]`.

Note: The operator * can also be used in string to verify a partial match of string, for example, `Name="Task*"`, will be used in a filter to return all tasks for which the name starts by "Task".

To write the statement `a<b<c`, you need to write `a<b AND b<c`, since these operators can only return true or false.

Logical Operators

These are joining operators. They are used in Boolean formulas to specify conditions and join different conditions.

Operator	Description
AND	Returns true if both operands are true; otherwise, returns false.
OR	Returns true if at least one operand is true; otherwise, returns false.
XOR	Returns true if only when both operands differ (one is true, the other is false).
NOT	Returns the Negation of a statement

Possible Boolean values:

Operator	Description
TRUE, YES	Boolean True
FALSE, NO	Boolean False

Special Case: IF Clause

The IF clause enables you to return two possible values depending on a condition. The syntax is the following:

```
IF condition THEN"value if condition is true"ELSE"value if condition is
                                    false"FI
```

Here is an example with a single IF clause:

Description	We want to know for each activity of our project the delay between the budget end date and the planned finish of the task. If the budget end date is empty, the formula will return 0. To achieve this, we write the following formula of type Duration:
Class	`Activity`
Formula	`IF ED_BUDGET ="" THEN 0 ELSE` `DIFF_DATE(ED_BUDGET,PF,CAL) FI`
Result	`'4d'`
Note	`ED_BUDGET: It is the technical name of the attribute that is tracking the budget end date of an activity` `CAL: It is the technical name of the field tracking the calendar of an activity.`

The **IF**starts the IF clause. **ED_BUDGET =""** is the condition that we want to test. If it is true, the formula will return the statement after **THEN** (0). If it is false, the formula will return the statement after **ELSE** (in the example, this is a function that calculates the difference between two dates). The **FI** closes the IF clause.

Here is an example using multiple IF clauses:

Description	We want to create a ranking formula (of type number) that will give a score to projects depending on their cash flow value. This formula can later be used in Planisware Explorer to build charts.

Formula	`IF _SC_NF_N_CASH_FLOW <= 0 THEN 0 ELSE`
	`IF _SC_NF_N_CASH_FLOW > 0 AND _SC_NF_N_CASH_FLOW <=10000 THEN 1 ELSE`
	`IF _SC_NF_N_CASH_FLOW > 10000 AND _SC_NF_N_CASH_FLOW <=20000 THEN 2 ELSE`
	`IF _SC_NF_N_CASH_FLOW > 20000 THEN 3`
	`FI FIFIFI`

Result	3

Special Case: IN Statement

The IN statement allows you to test if the value of an attribute belongs to a list of values instead of writing multiple "OR" conditions.

The syntax is the following:

```
attributeIN(Value1,Value2,Value3)
```

Here is an example:

Description	Formula will return TRUE if the product (object class defined in breakdown structure 1) linked to a project is in the list of values: Product A or Product B or product C.
Class	`Project`
Formula	`IF BREAKDOWN_1 IN("PROD-A", "PROD-B", "PROD-C") THEN TRUE ELSE FALSE FI`
Result	`TRUE`
Note	BREAKDOWN_1 is the technical name of the relation field that links a project to a product. PROD-A,PROD-B, PROD-C are, in this example, possible values for objects in that breakdown structure.

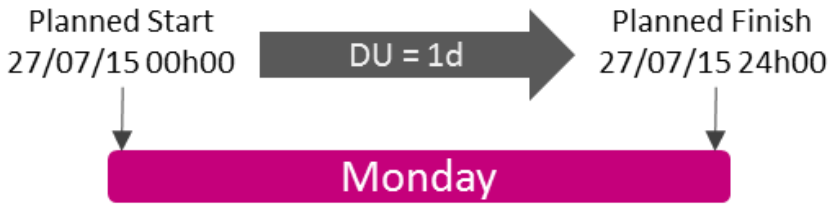
Special Information for Date and Duration Formulas

Date and duration calculations are a central point of Planisware. Before being able to use formulas correctly on these types of data, a few main concepts must be understood.

End Dates at Midnight

You might have noticed that activities with a duration of 0 days (thus milestones) seems to end before they started: the planned finish is the day before the planned start.

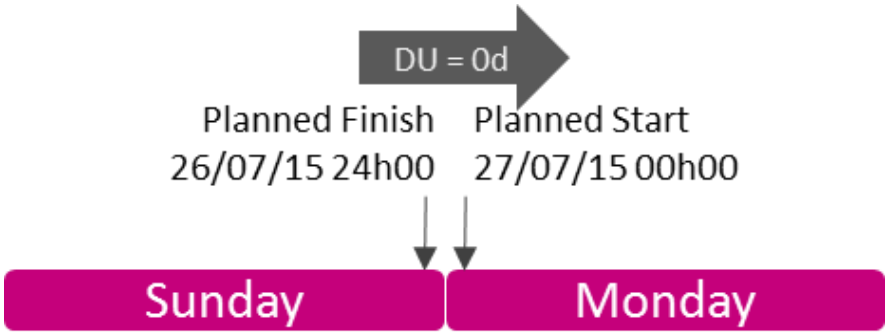
This is because of the default positioning of start and end type dates within a period. Suppose that you create a task with a duration of 1 day, starting on Monday:



Task duration and dates example

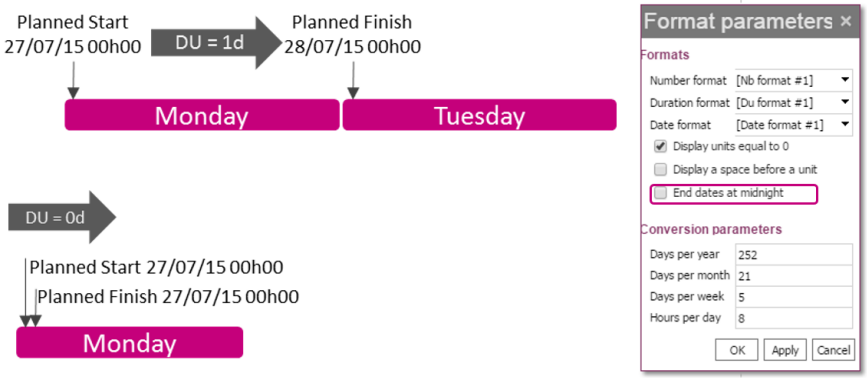
When you set a duration on activities, Planisware automatically decides that start dates should be at the beginning of periods, and end dates should be at the end of periods, without you having to manually set hours.

Now, if the activity is a milestone (DU = 0) Planisware will automatically set the dates as follows:



Milestone task duration and dates example

You can change this behavior in Pro Web, is necessary, by setting the parameter **End dates at midnight** to false. This is found in the parameters list table accessed via **Data > Parameters sets > Default formats**. The behavior will then be the following:



Milestone task duration and dates with no dates end at midnight

Task Type

There is alternative method you can use to define how the dates and times on tasks will be treated.

An attribute has been created for tasks called task type (TASK_TYPE). This attribute enables users to specify that a task should behave differently, in terms of date, if it is a milestone. This attribute can have one of the three following values:

1. **Task**, which is the regular behavior, requiring a planned start and planned finish date for the task.
2. **Start milestone**, which only requires a planned start date for the task.
3. **End milestone**, which only requires a planned finish date for the task.

This means that any tasks set as type start milestone or end milestone will no longer have two different dates (planned start and planned finish), but only one. The other date will be empty or hidden.

Restricting the Period of a Day

A day can start and stop at midnight. This can be modified, in Pro Web, using two parameters found in the **Time management** grouping, in the parameters list table (accessed via **Data > parameter sets > Other parameters**. These parameters are:

1. **Start hour for Day time unit**
2. **Finish hour for Day time unit**

If these parameters are set, respectively, to `8h` and `20h`, this will change the start and end times of dates from `00h00 > 24h00` to `08h00 > 20h00`.

The consequences of this change are:

- Start and end dates for activities will be set according to the hours defined in these two parameters.
- The **End dates at midnight** parameter is no longer relevant.
- Some values returned from formulas will be different. For example, the function DAY which should round a date to midnight will return 8am if the parameter **Start hour for day time unit** is set to 8.

Time Unit

The default time unit is a crucial parameter that decides with which level of precision Planisware calculates time. It is a choice that is made by your administrator while Planisware is implemented in your company and it cannot be modified afterwards. In fact, all projects must have the same time unit in order to have a consistent database. This choice has a large impact on the system's performance, which might explain why the most precise time unit has not been set.

Note: To check which time unit is used by your projects, you can add the **Time unit** attribute to a project list table, as a column.

It is possible to have a larger time unit for the project scheduling than the one used to measure resource load. For example, the projects and the calendars can have day as a time unit, and the resources can be measured in hours.

There are major consequences of what time units are used on named formulas. If the time unit of your projects is set to days, any formula trying to return data on hours based on a date format will not work properly.

For example, if the time unit is day:

- `HOUR_NUMBER('15/07/15 16:43:00')` returns nothing.
- `HOUR_NUMBER(PF)` returns nothing.

This is because the input date of the formula has the maximum precision of one day in the system. Even though the system might display hour data on any planned finish attributes, if you modify the date format used, the value is not stored in the system.

For example, if the time unit is day and the **Start hour for Day time unit** parameter is set to 8:

- `DAY('10/11/15 11:45:00')` returns `10/11/15 08:00:00`.
- `DAY('10/11/15 13:45:00')` returns `11/11/15 08:00:00`.

In this case, the function is supposed to round the day to midnight. However, because the time unit is day, hours are therefore interpreted as decimals, and the system will round to the closest day. Noon will therefore be the separator. The returned hour is 8am because of the **Start hour for Day time unit** parameter.

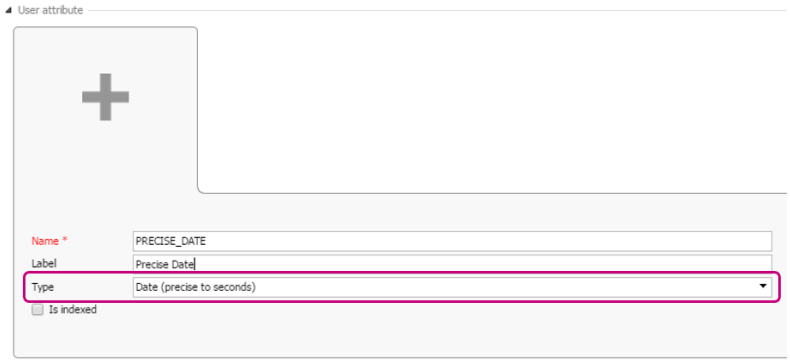
To work with hours and minutes without modifying the time unit, see the following section.

Date (Precise to Seconds) Type

Formula functions such as DAY, HOUR_NUMBER, and HOUR can be used without having to modify the time unit. This can be done by taking advantage of the data type called **Date (precise to seconds)**. Any data stored with this data type will be stored as a full date (for example `'10/11/15 11:45:00'`), without needing to modify the time unit.

In order to use this, you can:

- Go to the **Configuration** page in the administration module (in Planisware Web).
- Select activity for **Object type**.
- Create a user attribute as described in the following image.



Creating user attribute

Note: Although the field itself is precise to seconds, Planisware formulas language has a maximum precision of minutes. Any date such as `'10/11/15 11:45:31'` will therefore be rounded up to `'10/11/15 11:46:00'` if evaluated in a formula.

Date and Duration Functions

ADD_DURATION

Presentation

Description	Allows to add a duration to a date taking into account a calendar.
Type	DATE
French name	PLUS_DUREE
Treatment type	Date and Duration
See also	SUB_DURATION

Arguments

N°	Type	Description
1	Date	Origin date
2	Duration	Duration to add
3	String	Name of the calendar

Example 1

Description	We want to know the date of a task whose planned start (originally 06/07/15) might slip of 10 day, based on a 5 days per week calendar.
Class	Activity
Formula	`ADD_DURATION(PS,'10d',"CAL5D")`
Result	`18/07/15`
Note	PS is the technical name of the Planned Start field on the activity class "CAL5D" is the name of a calendar. Each implementation might have different calendar. Make sure to point toward an existing calendar.

Example 2

Description	Same example using the activity calendar.
Class	Activity
Formula	`ADD_DURATION(PS,'10d',CAL)`
Result	`18/07/15`
Note	This time, CAL refers to the name of the field on the activity class. This field might contain the value "CAL5D".

Note: Remember to verify that your dates format matches this example.

BEG_OF_MONTH

Presentation

Description	Rounds a date down to the beginning of the month
Type	DATE
French name	DEBUT_MOIS
Treatment type	Date and Duration

Arguments

N°	Type	Description
1	Date	Date to round

Example 1

Description	We want to retrieve the annotations made on the duration field of tasks.
Formula	`BEG_OF_MONTH('10/05/15')`
Result	`01/05/15`

Example 2

Description	Same example with US date format
Formula	`BEG_OF_MONTH('06/15/15')`
Result	`06/01/15`

Example 3

Description	We want to retrieve the beginning month for an activity.
Class	Activity
Formula	`BEG_OF_MONTH(PS)`
Result	`01/05/15`

Note: As shown in examples 1 and 2, the date format may vary. Note that the system will interpret the date format based on the default format of the system upon formula creation. If the default format is **DD/MM/YY** and the user tries to enter `06/13/15`, an error message will be displayed.

DAY

Presentation

Description	Rounds a date down to midnight*.
Type	DATE
French name	JOUR
Treatment type	Date and duration
See also	HOUR

Arguments

N°	Type	Description
1	Date (Precise to seconds)	Date to round

Example

Description	To return the date rounded to midnight of a date field on the class activity (precise to seconds) PRECISE_DATE = '16/07/15 17:29:00':
Class	Activity
Formula	`DAY(PRECISE_DATE)`
Result	`16/07/15 00:00:00`

Note: See Special Information for Date and Duration Formulas on page 16 to understand how dates, durations, time units, and periods work in Planisware. This will explain why certain formulas might not work in some cases.

DAY_OF_WEEK

Presentation

Description	Rounds a date down to a day of week
Type	DATE
French name	JOUR_SEMAINE
Treatment type	Date and Duration

Arguments

N°	Type	Description
1	Date	date to round down
2	Number	Desired day (1 for Monday, 7 for Sunday)

Example

Description	To return the first day of a week:
Formula	`DAY_OF_WEEK('08/07/15',1)`
Result	`11/11/15 08:00:00`

DIFF_DATE

Presentation

Description	Subtracts date 1 from date 2 and returns the duration between the two, taking into consideration the calendar.
Type	DURATION
French name	DIFF_DATE
Treatment type	Date and Duration

Arguments

N°	Type	Description
1	Date	Start date
2	Date	End date
3	String	Calendar name

Example 1

Description	From Wednesday 1st of July to Wednesday 8th of July, taking into account a five days' workweek (CAL5D):
Formula	`DIFF_DATE('01/07/15','08/07/15',"CAL5D")`
Result	`5d`

Example 2

Description	Same dates, this time without any calendar.
Formula	`DIFF_DATE('01/07/15','08/07/15',"")`
Result	`7d`

Example 3

Description	Activity Planned Finish Slippage in working days
Class	Activity
Formula	`DIFF_DATE(PF,ED_BUDGET,CAL)`
Result	`Gap between Planned Finish date and the budget finish date.`
Note	PF is the technical name of attribute "Planned Finish" on the activity class. ED_BUDGET is the technical name of the attribute tracking the budget end date of an activity.

HOUR

Presentation

Description	Returns the hour relative to a date in a duration output format.
Type	DURATION
French name	HEURE
Treatment type	Date and duration
See also	DAY

Arguments

N°	Type	Description
1	Date	Date

Example

Description	Suppose that we have a field on the class activity called PRECISE_DATE = '10/11/16 12:30:00'
Class	Activity
Formula	`HOUR(PRECISE_DATE)`
Result	'12h30m00s'

Note: If the parameter **Hours per day** seen in Pro Web via **Data > Parameters sets > Default formats** is set to less than 24 hours, the returned duration will be divided. Therefore, if there are 12 hours per day, the upper formula will return `'06h15m00s'`.

See Special Information for Date and Duration Formulas on page 16 to understand how dates, durations, time units, and periods work in Planisware. This will explain why certain formulas might not work in some cases.

HOUR_NUMBER

Presentation

Description	Returns the hour corresponding to a given date in a number format.
Type	NUMBER
French name	NUMERO_HEURE
Treatment type	Date and duration
See also	WEEK_NUMBER, YEAR_NUMBER, HOUR_NUMBER, MONTH_NUMBER

Arguments

N°	Type	Description
1	Date (Precise in seconds)	Date from which the hour number will be extracted

Example

Description	To return the hour corresponding to a precise date field on the class activity`PRECISE_DATE ='22/07/15 17:59:00':`
Class	Activity
Formula	`HOUR_NUMBER(PRECISE_DATE)`
Result	`17`

Note: See Special Information for Date and Duration Formulas on page 16 to understand how dates, durations, time units, and periods work in Planisware. This will explain why certain formulas might not work in some cases.

MONTH_LENGTH

Presentation

Description	Computes the length of the month of a given date
Type	DURATION
French name	LONGUEUR_MOIS
Treatment type	Date and duration

Arguments

N°	Type	Description
1	Date	Date whose month length is to be measured

Example

Description	To retrieve the number of days in a specific month
Formula	`MONTH_LENGTH('10-NOV-95')`
Result	`'30d'`

MONTH_NUMBER

Presentation

Description	Returns the month corresponding to a given date.
Type	NUMBER
French name	NUMERO_MOIS
Treatment type	Date and duration
See also	WEEK_NUMBER, YEAR_NUMBER, HOUR_NUMBER, MONTH_NUMBER

Arguments

N°	Type	Description
1	Date	Date

Example

Description	To return the number of a month within a given date:
Formula	MONTH_NUMBER('10-NOV-95')
Result	`11`

PERIOD_START

Presentation

Description	Returns the beginning date of a period based on a given date with a possible offset value.
Type	DATE
French name	DEBUT_PERIODE
Treatment type	Date and duration

Arguments

N°	Type	Description
1	Date	Start date
2	String	Time unit of the period – e.g.: year, quarter, month, week
3	Number	Number representing period offset

Example 1

Description	Suppose that we have an activity starting (PS) the 13-SEPT-15 (Sunday). We want to know the date corresponding to the beginning of the week (Monday):
Class	Activity
Formula	`PERIOD_START(PS,"week",0)`
Result	`07-SEPT-15`

Example 2

Description	Same than previous example, but we want to know the date of the beginning of the next week. We use an offset value of 1. (Note that negative values also work).
Class	Activity
Formula	`PERIOD_START(PS,"week",1)`
Result	`14-SEPT-15`

Example 3

Description	Same than previous example, but we want to know the date of the beginning of the next quarter. We use an offset value of 1. (Note that negative values also work).
Class	Activity
Formula	`PERIOD_START(PS,"quarter",1)`
Result	`01-OCT-15`

PRINT_PERIOD

Presentation

Description	Returns a period from a given date into the specified format
Type	STRING
French name	AFFICHE_PERIODE
Treatment type	Date and duration
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Date	Date whose period is searched
2	String	Destination format

Example

Description	To retrieve the year of a date in the format "19XX":
Formula	`PRINT_PERIOD('10-NOV-95',"19XX")`
Result	`"1955"`

Format List

Hours	Days	Weeks	Months	Years
"0,1,2...24"	"1,2...31"	"1,2...52"	"1,2...12"	"19XX"
"0,12,0,12"	"L,M,M...D"		"J,F,M..."	"92,93..."
"0,2,4...24"	"MONDAY,TUESDAY..."		"JAN,FEB..."	
"0,4,8,12,16,20"	"1,8,15..." (rounded up to the date of the Monday of the week)		"JANUARY,FEBUARY..."	
"0,6,12,18"			"S1,S2,..."	
"0h,0h05...23h55"			"T1,T2,..."	
"0h,0h15,0h30...23h45"				
"0h,0h30...23h30"				

Note: The formats will be precise to the minute only if the time unit of your project is set to minute.

RELATIVE_DATE

Presentation

Description	Returns the number of days of a date relative to the beginning of a sliding time scale.
Type	STRING
French name	DATE_RELATIVE
Treatment type	Date and duration

Arguments

N°	Type	Description
1	String	Sliding time scale name
2	Date	Date to compare with the scale

Example 1

Description	Suppose that we have set a time scale called TIME_SCALE whose origin date is the 01/01/01. We want to compare a date to this time scale:
Formula	`RELATIVE_DATE("TIME_SCALE",'15/01/01')`
Result	`D 14`

Example 2

Description	Same with a date anterior to the origin date of the time scale:
Formula	`RELATIVE_DATE("TIME_SCALE",'15/12/00')`
Result	`D-17`

Note: Sliding time scales can be managed in Pro Web in planning mode, from the menu **Scheduling** > **Sliding time scales**.

SUB_DURATION

Presentation

Description	Subtracts a duration from a specified date, taking into account the specified calendar
Type	DATE
French name	MOINS_DUREE
Treatment type	Date and Duration Treatments
See also	ADD_DURATION

Arguments

N°	Type	Description
1	Date	Origin date
2	Duration	Duration to add
3	String	Name of the calendar

Example

Description	We want to know the date of a task whose planned start (originally 06/07/15) might be anticipated of 10 day, based on a 5 days per week calendar.
Class	Activity
Formula	`SUB_DURATION(PS,'10d',"CAL5D")`
Result	`22/06/15`
Note	Technical name of Planned Start field on the activity class. '10d' is a hard-coded duration "CAL5" is a hard-coded calendar name. Depending on your implementation, the calendar names may vary. Make sure to use an existing calendar. You can also use CAL, without quotes, to refer to the activity calendar.

WEEK_NUMBER

Presentation

Description	Returns the week number (1 to 52) corresponding to a given date.
Type	NUMBER
French name	NUMERO_SEMAINE
Treatment type	Date and duration
See also	WEEK_NUMBER, YEAR_NUMBER, HOUR_NUMBER, MONTH_NUMBER

Arguments

N°	Type	Description
1	Date	Date

Example

Description	
Formula	WEEK_NUMBER('10-NOV-95')
Result	45

YEAR_NUMBER

Presentation

Description	Returns the year of a given date.
Type	NUMBER
French name	NUMERO_ANNEE
Treatment type	Date and duration
See also	WEEK_NUMBER, YEAR_NUMBER, HOUR_NUMBER, MONTH_NUMBER

Arguments

N°	Type	Description
1	Date	Date

Example

Description	To return the number of a year within a given date:
Formula	YEAR_NUMBER('10-NOV-95')
Result	1995

Breakdown Structure Functions

BELONGS

Presentation

Description	Returns TRUE if the specified object belongs to the specified work breakdown structure.
Type	BOOLEAN
French name	APPARTIENT
Treatment type	Breakdown Structure
See Also	FROM, NEW_CODE

Arguments

N°	Type	Description
1	String	Object class or relation
2	String	ID of the object

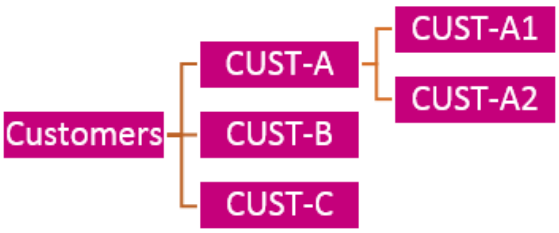
Example 1 - Simple

Description	Check whether a given resource belongs to a department called “Department A”. Resource is a breakdown structure
Class	Resource
Formula	`BELONGS("RESOURCE","Department A")`
Result	`TRUE`

Example 2 - On Relation

Description	Checks if a project customer belongs to the family “CUST-A”. More details below.
Class	Activity
Formula	`BELONGS("BREAKDOWN_0","CUST-A")`
Result	`TRUE`

Take the example where the breakdown BS0 is used to identify the different categories and names of customers. The structure could look like the following:



Example customer breakdown structure

Each project has in standard a 1-N relation to the customer breakdown structure, which means that a project can be linked to a customer, and a customer can be linked many projects.

This relation is materialized by the relation attribute "BREAKDOWN_0" on the project class.

Now, look at the result of the above formula in different cases:

PROJECT	CUSTOMER (BREAKDOWN_0)	BELONGS("BREAKDOWN_0","CUST-A")
PROJ1	CUST-A	TRUE
PROJ2	CUST-B	FALSE
PROJ3	CUST-A1	TRUE
PROJ4	CUST-C	

The system browses the reverse relation from customer to projects. In this cases it searches for customer A and all children in the list of projects for which the value matches.

Note: The BELONGS formula only works on breakdown structures that consolidate.

Refer to Best Practices for Performance on page 10 to see how this formula can be used to improve formulas performance.

FROM

Presentation

Description	Indicates if an object has a father in a breakdown structure.
Type	BOOLEAN
French name	A_PARTIR_DE
Treatment type	Breakdown Structure
See Also	BELONGS, NEW_CODE

Arguments

N°	Type	Description
1	String	ID of the parent

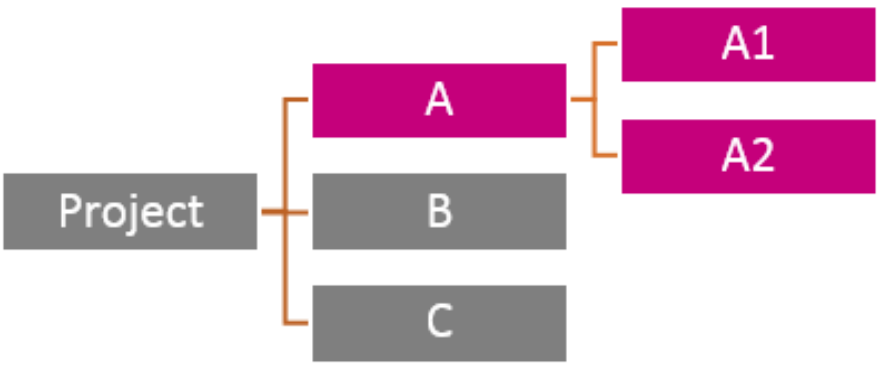
Example 1

Description	Considering a project with tasks and WBS elements represented by the following diagram. The following formula applied to all activities (as an added column) will create a Boolean for each of them.
Formula	`FROM("A")`
Result	`TRUE for A, A1 and A2`

Example 2

Description	Check whether a given resource belongs to a department called "Department A"
Class	Resource
Formula	`FROM("Department A")`
Result	`TRUE`

The following image shows a project work breakdown structure with activity IDS:



Example work breakdown structure

Note: Refer to Best Practices for Performance on page 10 to see how this formula can be used to improve formulas performance.

Type Conversion Functions

See Setting Default Formats on page 2 to understand how formats work in Planisware.

DATE

Presentation

Description	Converts a string to a date, in a given input format.
Type	DATE
French name	DATE
Treatment type	Conversion of Types

Arguments

N°	Type	Description
1	String	String Date
2	String	Input Format name

Example

Description	To convert the string "25/01/11" into a real date, type the following:
Formula	`DATE("25/01/11","DD/MM/YY")`
Result	`'25/01/11'`

Note: The returned date has to be inside the currently defined time window to be displayed.

DURATION

Presentation

Description	Converts a string into a duration
Type	DURATION
French name	DUREE
Treatment type	Conversion of Types

Arguments

N°	Type	Description
1	String	Duration string

Example

Description	If you have generated the string “3d4h” by using other formulas and need to recover a duration to use it into another function:
Formula	`DURATION("3d4h")`
Result	`'3d'`

Note: The result will be in “d” if the default format for durations is set to days.

To know whether you should use ““ or not within your formulas, you can refer to Syntax Rules: Use of Quotation Marks on page 4.

NUMBER

Presentation

Description	Converts a string into a number using the specified format
Type	NUMBER
French name	NOMBRE
Treatment type	Conversion of Types

Arguments

N°	Type	Description
1	String	number string
2	String	Format name

Example

Description	To convert a number returned as a string into a real number:
Formula	`NUMBER("355,02","####")`
Result	`355.02`

PRINT_COST

Presentation

Description	Convert a cost to a given format
Type	STRING
French name	AFFICHE_COUT
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Number	number
2	String	Format name

Example

Description	To change the separator used to display a cost, use the following formula.
Formula	`PRINT_COST(200.5,"####,00")`
Result	`"200,5"`

PRINT_DATE

Presentation

Description	Converts a date into the specified format.
Type	STRING
French name	AFFICHE_DATE
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Date	Date to convert
2	String	Destination format

Example

Description	To convert '10-NOV-95' into "95/11/10 12:00":
Formula	`PRINT_DATE('10-NOV-95',"YY/MM/DD HH:MM")`
Result	`"95/11/10 12:00"`

PRINT_DATE_LANGUAGE

Presentation

Description	Convert a date to a given format in a given language
Type	STRING
French name	AFFICHE_DATE_LANGAGE
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Date	Date to convert
2	String	Destination format
3	String	Language

Example

Description	To print a date with months in full letters and in French:
Formula	`PRINT_DATE_LANGUAGE(PS,"MMMM/JJ/AA","FRENCH")`
Result	"JUILLET/16/15"

PRINT_DURATION

Presentation

Description	Converts a duration into the specified format
Type	STRING
French name	AFFICHE_DUREE
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Duration	Duration input field
2	String	Format of the destination duration

Example

Description	To change the displayed duration of a field into another format:
Formula	`PRINT_DURATION('1d',"hm")`
Result	`"24h"`

PRINT_END_DATE

Presentation

Description	Displays an end date with the provided format
Type	STRING
French name	AFFICHE_DATE_FIN
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Date	number which is an end date
2	String	

Example

Description	Converts the time format of a task whose planned start is 23/07/15 into 15/07/22. The day change from 23 to 22 is because it is considered by Planisware that the end of the previous day is equal to the beginning of the current day. Refer to End dates at midnight for more information.
Class	Activity
Formula	`PRINT_END_DATE(PS,"YY/MM/DD HH:MM")`
Result	`15/07/22 20:00`

Comparison with PRINT_DATE

Description	Comparison with print date
Formula	`PRINT_DATE(PS,"YY/MM/DD HH:MM")`
Result	`15/07/23 00:00`

Note: The hour where one day becomes the next is not necessarily midnight. In the examples, it is set to 20:00. This is specified using the **Finish hour for day time unit** parameter that can be found in Pro Web. More information on this parameter can be found in Restricting the Period of a Day on page 17.

PRINT_NUMBER

Presentation

Description	Converts a number into the specified format.
Type	STRING
French name	AFFICHE_NOMBRE
Treatment type	Conversion of Types
See also	PRINT_COST, PRINT_DATE, PRINT_DATE_LANGUAGE, PRINT_DURATION, PRINT_END_DATE, PRINT_NUMBER, PRINT_PERIOD

Arguments

N°	Type	Description
1	Number	Number to convert
2	String	Destination format

Example

Description	To convert a number using the "." Separator to a number using the "," separator:
Formula	`PRINT_NUMBER(200.5,"####,00")`
Result	`"200,50"`

Note: To obtain a list of the formats that can be used, by default, for your conversion, you can refer to the fields available in the dates and units dialog in Planisware Web (Accessed via **Planisware** > **Dates and units**.

String Treatment Functions

ENDSUBSTRING

Presentation

Description	Extracts a sub-string from a string, given a number of characters to extract and their relative position to the end of the string.
Type	STRING
French name	FINSOUSCHAINE
Treatment type	String Treatments
See also	SUBSTRING, SUBSTITUTE

Arguments

N°	Type	Description
1	String	String to be treated
2	Number	number of characters in the extracted substring
3	Number	Starting point from the right.

Example

Description	To extract the word " World" (with a space) from a specific string:
Formula	`ENDSUBSTRING("Hello, World !",6,2)`
Result	`" World"`

Note: The last character of the string, or the first starting from the right is numbered 0.

FORMAT_STRING

Presentation

Description	Creates a dynamic string based on a skeleton and a list of fill words that can be added to specific positions of the skeleton.
Type	STRING
French name	FORMAT_CHAINE
Treatment type	String Treatments

Arguments

N°	Type	Description
1	String	Skeleton of the string. Use %1 to specify the position of the first element to be placed from the list.
2	String	List of elements to be placed into the skeleton.
3	String	List delimiter

Example

Description	Simple example
Formula	`FORMAT_STRING("Hello %1, how %2 you?","theworld;are",";")`
Result	`"Hello the world, how are you?"`

GET_ASCII

Presentation

Description	Get character ASCII code from a character
Type	NUMBER
French name	DONNE_ASCII
Treatment type	String treatments
See also	GET_CHAR

Arguments

N°	Type	Description
1	String	Character string

Example

Description	To get the ASCII of the character “A”
Formula	`GET_ASCII("A")`
Result	`65`

Note: You can find a list of ASCII characters at http://www.asciitable.com/.

GET_CHAR

Presentation

Description	Returns an ASCII character from its code
Type	STRING
French name	DONNE_CARACTERE
Treatment type	String treatments
See also	GET_ASCII, RICHTEXT

Arguments

N°	Type	Description
1	Number	Code of the ASCII

Example

Description	To add a line feed to a string composed of two attributes
Formula	`NAME + GET_CHAR(10) + Desc`
Result	`"T1` `Task 1"`

Note: You can find a list of ASCII characters at http://www.asciitable.com/.

LENGTH

Presentation

Description	Returns the length of a string.
Type	NUMBER
French name	LONG
Treatment type	String Treatments

Arguments

N°	Type	Description
1	String	String to measure

Example

Description	To add a line feed to a string composed of two attributes
Formula	`LENGTH("Hello, World !")`
Result	`14`

LOWERCASE

Presentation

Description	Transforms a string into lower case characters
Type	STRING
French name	MINUSCULES
Treatment type	String Treatments
See also	UPPERCASE

Arguments

N°	Type	Description
1	String	String to transform

Example

Description	
Formula	`LOWERCASE("Hello, World !")`
Result	`"hello, world !"`

MATCH_STRING

Presentation

Description	Filters objects that contains a specific string in any of their attributes.
Type	BOOLEAN
French name	CONTIENT_CHAINE
Treatment type	String treatments

Arguments

N°	Type	Description
1	String	string used to filter

Example

Description	To filter the list of all tasks that contain the string "Task" in any of their string type fields (Name, description, notepad, etc.):
Formula	`MATCH_STRING("Task")`
Result	`TRUE`

Note: The use of a wildcard "*" is not necessary since the function already searches for the presence of the string passed as an argument inside any field string. In the example, the result will be TRUE for "TASK 1", "MyTASK", and so on.

NTH

Presentation

Description	Extracts the Nth element from a string separated by a specified separator.
Type	STRING
French name	NIEME
Treatment type	String Treatments

Arguments

N°	Type	Description
1	String	string
2	Number	Element number to extract
3	String	String separator

Example

Description	To retrieve the third element of the list, considering that the spaces between words are part of the separators and not the elements:
Formula	`NTH("3h, 2h, 1h",2,", ")`
Result	`"1h"`

POSITION

Presentation

Description	Returns the position of a string inside another string.
Type	NUMBER
French name	POSITION
Treatment type	String Treatments
See also	POSITION_END

Arguments

N°	Type	Description
1	String	Searched string
3	String	String in which the search is done

Example

Description	To retrieve the third element of the list, considering that the spaces between words are part of the separators and not the elements:
Formula	`POSITION("ab","azertyabaerty")`
Result	6

Note: Planisware returns the position of the first character of the argument 1. If the string is not found, the default displayed value will be -1.

RICHTEXT

Presentation

Description	Returns the rich text associated to a field if it exists, otherwise the raw text is returned.
Type	STRING
French name	RICHTEXT
Treatment type	String treatments
See also	GET_CHAR, RICHTEXT_COLLECT

Arguments

N°	Type	Description
1	String	string defining the field name

Example

Description	Displaying an attribute that contains rich text formatting will not be displayed by default in Planisware. For example, adding the column NOTE_PAD to a list of tasks will only show raw text. To display rich text, type the following:
Formula	RICHTEXT("NOTE_PAD")
Result	"ContentOF THE note Pad"

RICHTEXT_COLLECT

Presentation

Description	Builds a valid rich text using the field on all objects of the specified class or relation that verify the filter. The result is not a rich text if and only if none of the field of the objects is a rich text.
Type	STRING
French name	RICHTEXT_COLLECTER
Treatment type	String treatments
See also	GET_CHAR, RICHTEXT

Arguments

N°	Type	Description
1	String	class or relation
2	String	Filter applied to the class
3	String	Field

Example 1

Description	To recover rich text for the field Note Pad for tasks whose name is T1:
Formula	`RICHTEXT_COLLECT("TASK","NAME=\"T1\"","NOTE_PAD")`
*Result**	"Note PadFor Task 1"

Example 2

Description	To recover rich text for the field Note Pad for tasks whose name starts by T:
Formula	`RICHTEXT_COLLECT("TASK","NAME=\"T*\"","NOTE_PAD")`
*Result**	"Note Pad for Task 2Note PadFor Task 1"

Note: The displayed text is what is shown in examples. The real text returned by Planisware is HTML text. For example:

```
<html>
<p>Project1 non rich text content</p>
<p>Project2 <b>rich text content</b></p>
</html>
```

HTML example

Refer to Syntax Rules: Use of Quotation Marks on page 4 for information on the use of \.

STRING

Presentation

Description	Automatically generates a string based on a pattern
Type	STRING
French name	CHAINE
Treatment type	String Treatments

Arguments

N°	Type	Description
1	String	String pattern to generate
2	Number	Number of occurrences

Example 1

Description	To generate a simple string:
Formula	`STRING("Hello",4)`
Result	`"HelloHelloHelloHello"`

Example 2

Description	To generate a complex string based on task attributes:
Formula	`STRING(Name,1) + "-" + STRING(DESC,1)`
Result	"PHASE1 - Development"

STRING_VALUE_LANGUAGE

Presentation

Description	Returns the value contained in the specified string field, in the specified language, of the specified object. The function returns a list of values if a list of object index is given.
Type	STRING
French name	VALEUR_CHAINE_LANGAGE
Treatment type	String Treatments
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object
3	String	Name of the field
4	String	Language

Example

Description	To get the French value of a project state (“closed” in English)
Class	Activity
Formula	`STRING_VALUE_LANGUAGE("PROJECT","TESTPROJ","STATE","FRENCH")`
Result	`"Cloturé"`

SUBSTITUTE

Presentation

Description	Replaces one character chain with another
Type	STRING
French name	SUBSTITUE
Treatment type	String Treatments
See also	SUBSTRING, ENDSUBSTRING

Arguments

N°	Type	Description
1	String	New sub-string
2	String	Old sub-string
3	String	Entire string

Example 1

Description	To replace "Main subproject" by "Main WBS element":
Formula	`SUBSTITUTE("WBS element","subproject","Main subproject")`
Result	`"Main WBS element"`

Example 2 (Applied to a List)

Description	To replace “T” by “Task in the list composed of T1, T2 and T3 :
Formula	`SUBSTITUE("Task","T","T1,T2,T3")`
Result	`"Task1,Task2,Task3"`

SUBSTRING

Presentation

Description	Extracts a sub-string from a string, given a number of characters to extract and their relative position to the beginning of the string.
Type	STRING
French name	SOUSCHAINE
Treatment type	String Treatments
See also	ENDSUBSTRING, SUBSTITUTE

Arguments

N°	Type	Description
1	String	string
2	Number	beginning of the extracted substring
3	Number	End of the extracted substring

Example

Description	To retrieve the duration(DU) of My_Task:
Formula	`SUBSTRING("Hello, World !",2,6)`
Result	`"llo,"`

TRUNCATE

Presentation

Description	Transforms a decimal number into an integer
Type	NUMBER
French name	TRONQUE
Treatment type	Numerical

Arguments

N°	Type	Description
1	Number	Number to truncate

Example

Description	To recover an integer from a decimal number
Formula	`TRUNCATE(27.385)`
Result	`27`

UNDERLINE

Presentation

Description	Extracts a line from a string.
Type	STRING
French name	SOUSLIGNE
Treatment type	String Treatment
See also	FORMATTED_SUBLINE

Arguments

N°	Type	Description
1	String	string
2	Number	Line number

Example

Description	To show the second line of a string:
Formula	`UNDERLINE("Hello,` `World !",1)`
Result	`"World !"`

UPPERCASE

Presentation

Description	Transforms a string into upper case characters
Type	STRING
French name	MAJUSCULES
Treatment type	String Treatments
See also	LOWERCASE

Arguments

N°	Type	Description
1	String	String to transform

Example

Description	
Formula	UPPERCASE("Hello, World !")
Result	"HELLO, WORLD !"

List Treatment Functions

The first element of a list has the number 0. Consequently, a list of 5 elements will have them numbered from 0 to 4.

Elements of a list are only separated by a comma. Any spaces are part of the list elements. There are exceptions for functions where the separator can be configured.

LIST_COLLECT

Presentation

Description	Constructs a list by scanning all the objects of the specified class or relation that verify a filter.
Type	STRING
French name	LISTE_COLLECTER
Treatment type	List Treatments
See also	LIST_MAKE

Arguments

N°	Type	Description
1	String	class or relation of objects constituting the list
2	String	Filter / condition to verify on the objects
3	String	Formula defining the string that will be displayed in each list element

Example 1

Description	To build a list containing the ID of all projects whose owner is Maxence:
Class	Project
Formula	`LIST_COLLECT("PROJECT","OWNER=\"maxence\"","ID")`
Result	`"Project A,ProjectB,ProjectC,Project D"`

Note: This formula may cause performance issues. Please refer to Best Practices for Performance on page 10 for more information. It is highly advised not to use LIST_COLLECT functions in features built into an Intranet launched on Planisware Server (the Planisware Web solution, for example).

Refer to Syntax Rules: Use of Quotation Marks on page 4 for information on the use of \.

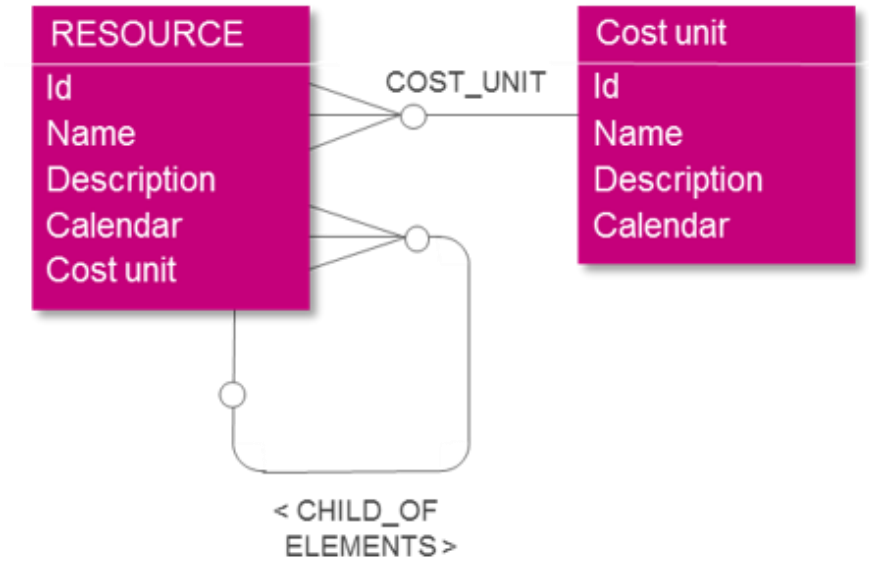
Example 2 - On a Relation

Description	In this example, we are working on the Resource Class, and we want to add a column that displays for each resource its list of child resources and their associated cost unit. (See the diagram on next page)
Class	Resource
Formula	`LIST_COLLECT("ELEMENTS","","NAME + \" - \" + COST_UNIT.NAME")`
Result	(See screenshot on next page)"RM179 - Default cost unit[EUR],RM181 - Default cost unit[EUR],RM180 - Default cost unit[EUR],RM178 - Default cost unit[EUR]"

Name	Element of	=LIST_COLLECT("ELEMENTS","","NAME + \" - \" + COST_UNIT.NAME")	Cost unit
ARBS		OTHER - Default cost unit[EUR],RBSD-BM - Default cost unit[EUR],RBSD-A...	Default cost unit...
OTHER	ARBS	RM196 - Default cost unit[EUR],RM195a - Default cost unit[EUR],RM195b - D...	Default cost unit...
RBSD-AVAIL	ARBS	RM179 - Default cost unit[EUR],RM181 - Default cost unit[EUR],RM180 - Def...	Default cost unit...
RM178	RBSD-AVAIL		Default cost unit[EUR]
RM179	RBSD-AVAIL		Default cost unit[EUR]
RM180	RBSD-AVAIL		Default cost unit[EUR]
RM181	RBSD-AVAIL		Default cost unit[EUR]

Result of LIST_COLLECT function

Example 2 - Data Model Explanations



Result of LIST_COLLECT function

The starting point for the example is the resource class. This is defined by the fact that the formula is created on a resource table.

From this point, the goal is to collect a list of the children of each resource. This is a reflexive 1 – N relation. A resource is the child of one other resource. This can be identified by the attribute CHILD_OF. In the other direction, a resource can have multiple children. This can be identified by the inverse relation attribute ELEMENTS.

The example uses ELEMENTS as the first argument of our LIST_COLLECT function to navigate downwards from a resource to its children.

The second argument of the function is empty because a filter is not needed.

The third argument is the formula that will contain both the NAME of the child and its cost unit. Since a resource can only have one cost unit, we can use the syntax COST_UNIT.NAME to retrieve the name of the cost unit linked to the resource.

LIST_DIFFERENCE

Presentation

Description	Returns a list of elements of list 1 that do not appear in list 2.
Type	STRING
French name	LISTE_DIFFERENCE
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List 1
2	String	List 2

Example

Description	To compare the difference between list 1 : "A,B,D,E,G" and list 2 :"A,B,C,D,F"
Formula	`LIST_DIFFERENCE("A,B,D,E,G","A,B,C,D,F")`
Result	`"E,G"`

LIST_EQUAL

Presentation

Description	Indicates if two lists contain the same elements. The order is not important.
Type	BOOLEAN
French name	LISTE_EGALITE
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List 1
2	String	List 2

Example

Description	Example 1	Example 2
Formula	`LIST_EQUAL("D,A,B","B,D,A")`	`LIST_EQUAL("X","A,C,D,E")`
Result	`YES`	`NO`

LIST_EXTRACT

Presentation

Description	Returns a sub-list of the specified list using the position of first and last elements to be extracted.
Type	STRING
French name	LISTE_EXTRAIRE
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	list
2	String	Position of first element to extract
3	String	Position of last element to extract

Example

Description	Example 1	Example 2
Formula	`LIST_EXTRACT("A,B,C",0,1)`	`LIST_EXTRACT("A,B,C",1,3)`
Result	`"A"`	`"B,C"`

LIST_FIND

Presentation

Description	Indicates if an item is present in a list.
Type	BOOLEAN
French name	LISTE_DANS
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	Item to find
2	String	list

Example

Description	We search for "B" in "B,D":	We search for "X" in "A,C,D,E"
Formula	`LIST_FIND("B","B,D")`	`LIST_FIND("X","A,C,D,E")`
Result	`YES`	`NO`

Note: The first argument cannot be a list. For instance, the expression `LIST_FIND("B,D","B,D")` will not work.

LIST_INTERSECT

Presentation

Description	Returns a list containing all the elements that appear in BOTH of the two lists that are passed as arguments
Type	STRING
French name	LISTE_INTERSECTION
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List 1
2	String	List 2

Example

Description	Example 1	Example 2
Formula	`LIST_INTERSECT("A,B,C","B,D")`	`LIST_INTERSECT("A,B,C","A,C,D,E")`
Result	`"B"`	`"A,C"`

LIST_LENGTH

Presentation

Description	Returns the number of items contained in the list.
Type	INTEGER
French name	LISTE_LONGUEUR
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List

Example

Description	To measure the length of a list (number of items):
Formula	`LIST_LENGTH("TaskA,TaskB,TaskC")`
Result	3

LIST_MAKE

Presentation

Description	Constructs a list containing N occurrences of the element passed as an argument.
Type	STRING
French name	LISTE_CONSTRUIRE
Treatment type	List Treatments
See also	LIST_COLLECT

Arguments

N°	Type	Description
1	String	Length of the list to construct
2	String	Element to instantiate

Example

Description	To construct a list of four instances of “A”
Formula	`LIST_MAKE(4,"A")`
Result	`"A,A,A,A"`

Note: The element can neither be a class nor an attribute. If it needs to be, you will need to use `LIST_COLLECT` see LIST_COLLECT on page 45 for more information.

LIST_MERGE

Presentation

Description	Merges the two lists passed as arguments while eliminating duplicates.
Type	STRING
French name	LISTE_UNION
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List 1
2	String	List 2

Example

Description		
Formula	`LIST_MERGE("A,B,C","B,D")`	`LIST_MERGE("A,B,C","A,B")`
Result	`"A,B,C,D"`	`"A,B,C"`

Note: The resulting list is not sorted by default.

LIST_MODIFY

Presentation

Description	Replaces the Nth element of a list by a value passed as an argument
Type	STRING
French name	LISTE_MODIFIER
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List (by ID)
2	String	Position of the element to be replaced
3	String	Replacement value

Example

Description	To replace the element "A" of the list "A,B,C" by "X", we write the following statement:
Formula	`LIST_MODIFY("A,B,C",0,"X")`
Result	`"X,B,C"`

Note: Remember that the first element of a list is 0 and not 1.

LIST_NOTEXIST

Presentation

Description	Creates a list from an existing one using all its elements who do not verify a certain filter.
Type	STRING
French name	LISTE_NONEXISTE
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	list
2	String	class
3	String	Filter applied to the list

Example

Description	In this example, only task B has a duration of 10 days. This function will therefore extract all but B into a new list.
Formula	`LIST_NOTEXIST("A,B,C","TASK","DU='10d'")`
Result	`"A,C"`

LIST_POSITION

Presentation

Description	Returns the position of an element in a list
Type	NUMBER
French name	LISTE_POSITION
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	item
2	String	

Example

Description	Example 1	Example 2
Formula	`LIST_POSITION("D","B,D")`	`LIST_POSITION("X","A,C,D,E")`
Result	`1`	`-1`

Note: The formula will return -1 if the element is not in the list.

LIST_REMOVE

Presentation

Description	Removes all occurrences of the specified item from the specified list
Type	STRING
French name	LISTE_SUPPRIMER
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	Item to remove
2	String	List

Example

Description	Example 1	Example 2
Formula	`LIST_REMOVE("A","A,B,C")`	`LIST_REMOVE("A","A,B,A,C,A")`
Result	`"B,C"`	`"B,C"`

LIST_REMOVE_DUPLICATES

Presentation

Description	Removes all duplications of the specified item from the specified list
Type	STRING
French name	LISTE_SUPPRIMER_DUPLICATA
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List

Example

Description	To remove all duplicates from a list:
Formula	`LIST_REMOVE_DUPLICATES("A,B,A,A,A,C")`
Result	`"A,B,C"`

LIST_SORT

Presentation

Description	Sorts a list into alphabetical order
Type	STRING
French name	LISTE_TRI
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List to sort

Example

Description	Example 1	Example 2
Formula	`LIST_SORT("D,A,B")`	`LIST_SORT(“1,5,4,3,5”)`
Result	`"A,B,D"`	`“1,2,3,4,5”`

LIST_SUBSTITUTE

Presentation

Description	Replaces all occurrences of a specified element in a list with a new value
Type	STRING
French name	LISTE_SUBSTITUER
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List
2	String	Old value
3	String	New Value

Example

Description	Example 1	Example 2
Formula	`LIST_SUBSTITUTE("A,B,C",","A","X")`	`LIST_SUBSTITUTE("A,B,C","B","Y")`
Result	`"X,B,C"`	`"A,Y,C"`

LIST_SUM

Presentation

Description	Compute the sum of all elements of the list
Type	NUMBER
French name	LISTE_TOTAL
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List to sum

Example

Description	Example 1	Example 2
Formula	`LIST_SUM("A,B,C")`	`LIST_SUM("1,2,3")`
Result	0	6

LIST_THEREIS

Presentation

Description	Searches the specified list and returns TRUE if at least one element in the list verifies the filter passed as an argument
Type	BOOLEAN
French name	LISTE_ILYA
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List of objects on which the condition will be verified.
2	String	Class of the objects in the list.
3	String	Filter or condition applied to the objects.

Example

Description	To verify if at least one task in the defined list has a duration of 10 days, we write the following statement:
Class	Activity
Formula	`LIST_THEREIS("A,B,C","TASK","DU='10d'")`
Result	`YES`

LIST_VALUE

Presentation

Description	Returns the value Nth element of a list
Type	STRING
French name	LISTE_VALEUR
Treatment type	List Treatments

Arguments

N°	Type	Description
1	String	List
2	Number	Position of the element to extract

Example

Description	Example 1	Example 2
Formula	`LIST_VALUE("A,B,C",0)`	`LIST_VALUE("A,B,C",1)`
Result	`"A"`	`"B"`

Numerical Functions

ABS

Presentation

Description	Returns the absolute value of a number
Type	NUMBER
French name	ABS
Treatment type	Numerical

Arguments

N°	Type	Description
1	Number	Number to evaluate

Example

Description	Absolute Value of -1
Formula	`ABS(-1)`
Result	1

MAX

Presentation

Description	Returns the maximum of the two number passed as arguments
Type	NUMBER
French name	MAX
Treatment type	Numerical
See also	MAX, MIN, MOD

Arguments

N°	Type	Description
1	Number	value1
2	Number	value2

Example

Description	Example 1	Example 2
Formula	`MAX(10,3)`	`MAX(100,60)`
Result	`10`	`100`

MIN

Presentation

Description	Returns the minimum of the two number passed as arguments
Type	NUMBER
French name	MIN
Treatment type	Numerical
See also	MAX, MIN, MOD

Arguments

N°	Type	Description
1	Number	value1
2	Number	value2

Example

Description	Example 1	Example 2
Formula	`MIN(10,3)`	`MIN(100,60)`
Result	`3`	`60`

MOD

Presentation

Description	Returns the modulo of the two arguments.
Type	NUMBER
French name	MOD
Treatment type	Numerical
See also	MAX, MIN,ROUND_NUMBER, QUO

Arguments

N°	Type	Description
1	Number	Number
2	Number	

Example

Description	The modulo operation finds the remainder after division of one number by another. Eg: 10/3 = 3 + **1**
Formula	`MOD(10,3)`
Result	1

RANDOM_NUMBER

Presentation

Description	Returns a random number between 0 and the number passed as an argument
Type	NUMBER
French name	NUMERO_ALEATOIRE
Treatment type	Numerical

Arguments

N°	Type	Description
1	Number	Upper bound for the random range. Lower bound is always 0

Example

Description	Example 1	Example 2: rising the lower bound
Formula	`RANDOM_NUMBER(100)`	`RANDOM_NUMBER(100)+1000`
Result	23	1030

Note: This number might be recalculated during the session. It is therefore not saved in Planisware and will be constantly evolving.

ROUND_NUMBER

Presentation

Description	Round the number resulting from the division of two numbers (arg1/arg2).
Type	NUMBER
French name	NUMERO_ARRONDI
Treatment type	Numerical
See also	MOD, QUO

Arguments

N°	Type	Description
1	Number	dividend
2	Number	Divisor

Example

Description	10/3 = 3.3333	11/3 = 3.666
Formula	`ROUND_NUMBER(10,3)`	`ROUND_NUMBER(11,3)`
Result	3	4

Data Retrieval Functions

Functions in this section enable users to retrieve data linked to an object.

?OBJECT_EXISTS

Presentation

Description	Searches for an object using its class and ID. Returns TRUE if it exists, FALSE if not.
Type	BOOLEAN
French name	?OBJET_EXISTE
Treatment type	Retrieve Data

Arguments

N°	Type	Description
1	String	Class of the searched object
2	String	ID of the object

Example

Description	If we are searching for the existence of a task with the ID “MyTask” that does not exist:
Formula	`?OBJECT_EXISTS("Task","MyTask")`
Result	NO

ANNOTATION

Presentation

Description	Returns the annotation associated to an attribute
Type	STRING
French name	ANNOTATION
Treatment type	Retrieve Data

Arguments

N°	Type	Description
1	String	Attribute whose annotation should be returned.

Example

Description	We want to retrieve the annotations made on the duration field of tasks.
Formula	`ANNOTATION("DU")`
Result	`"* 06/07/15 ALIX This duration is too long!"`

ATTRIBUTE_TYPE_COMMENT

Presentation

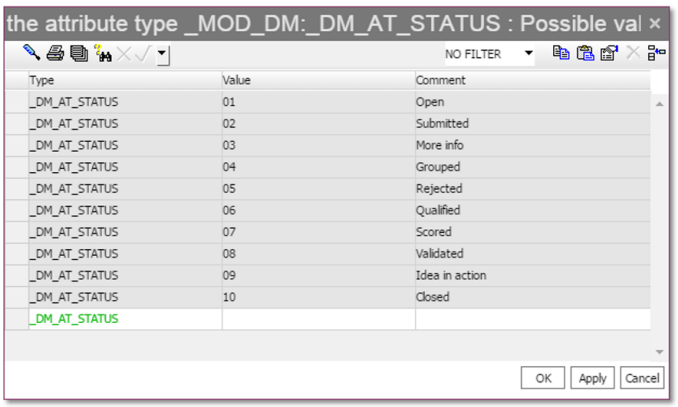
Description	Retrieves the name of a possible value relative to a new attribute type. Common attribute types are “String”, “Boolean”, “Date”, etc. Administrators can create new attribute types using Planisware Pro and add a list of possible values for this attribute type. For example, the possible values for the attribute type “Boolean” are “TRUE” and “FALSE”.
Type	STRING
French name	DESC_TYPE_ATTRIBUT
Treatment type	Retrieve Data
See Also	ATTRIBUTE_TYPE_VALUE

Arguments

N°	Type	Description
1	String	Name of the attribute type
2	String	Possible value of which the name is searched

Example

Description	Let’s suppose that we want to retrieve the name of the second possible value for the attribute type “Request status” – “_DM_AT_STATUS”. The following screenshot illustrates the formula behavior.
Formula	`ATTRIBUTE_TYPE_COMMENT("_DM_AT_STATUS","02")`
Result	`Submitted`



To access this popup, from Planisware Pro, Go to Data > Data Model Configuration > Attribute Type. Select an object from the list and click “Possible Values”.

Possible value list in Pro Web

ATTRIBUTE_TYPE_VALUE

Presentation

Description	Retrieves the value of a possible value relative to a new attribute type based on its comment. See ATTRIBUTE_TYPE_COMMENT for more information on this formula.
Type	STRING
French name	VALEUR_TYPE_ATTRIBUT
Treatment type	Retrieve Data
See Also	ATTRIBUTE_TYPE_COMMENT

Arguments

N°	Type	Description
1	String	Name of the attribute type
2	String	Comment

Example

Description	Let’s suppose that we want to retrieve the value of the second possible value for the attribute type “Request status” – “_DM_AT_STATUS”.
Formula	`ATTRIBUTE_TYPE_VALUE("_DM_AT_STATUS","Submitted")`
Result	`02`

BELONGS

Presentation

Description	Returns TRUE if the specified object belongs to the specified work breakdown structure.
Type	BOOLEAN
French name	APPARTIENT
Treatment type	Breakdown Structure
See Also	FROM, NEW_CODE

Arguments

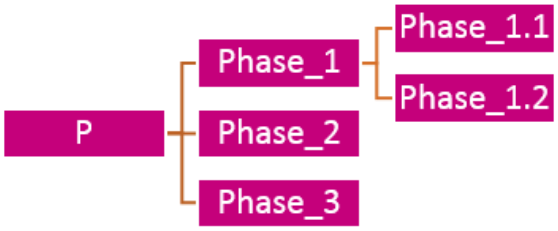
N°	Type	Description
1	String	Object class
2	String	ID of the object

Example 1

Description	If there is a task named T1 in currently opened files:
Formula	`BELONGS("TASK","T1")`
Result	`TRUE`

Example 2

Description	To verify if a list of activities belongs to the Phase_1 of the project P within the tree called "SUB_PROJECT". This will also be true for all activities who belong to children of Phase_1: eg Phase_1.2.
Formula	`BELONGS("SUB_PROJECT","P/Phase_1")`
Result	`TRUE`



Example WBS

Note: Breakdown structures such as this one can be managed in the **Data** page, in the administration module in Planisware Web. You can use the **Object type** field to select the right structure.

BOOLEAN_VALUE

Presentation

Description	Returns the value contained in the specified Boolean field, of the specified object
Type	BOOLEAN
French name	VALEUR_BOOLEEN
Treatment type	Retrieve Data
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object
3	String	Name of the Boolean field

Example

Description	We want to know if the object of type "Task" named "Task1" is finished, meaning that the attribute ?FINISHED is true (This field is calculated and depends on the field Actual finish, AF). To retrieve the value of this attribute for this specific object, we write the following statement.
Formula	`BOOLEAN_VALUE("Task","Task4","?FINISHED")`
Result	`TRUE`

CLASS_COMMENT

Presentation

Description	Returns a class comment from its name. (CLASS_DESC is returned.)
Type	STRING
French name	DESC_CLASSE
Treatment type	Retrieve data
See also	CLASS_PLURAL

Arguments

N°	Type	Description
1	String	Name of the class

Example 1

Description	Example 1
Formula	`CLASS_COMMENT("TASK")`
Result	`"Task"`

Example 2

Description	Example 2
Formula	`CLASS_COMMENT("SUBPROJECT")`
Result	`"WBS Element"`

Example 3

Description	Example 3
Formula	`CLASS_COMMENT("3BS")`
Result	`"My Breakdown Structure"`

CLASS_PLURAL

Presentation

Description	Returns a class plural from its name
Type	STRING
French name	PLURIEL_CLASSE
Treatment type	Retrieve data
See also	CLASS_COMMENT

Arguments

N°	Type	Description
1	String	Name of the class

Example

Description	
Formula	`CLASS_PLURAL("TASK")`
Result	`"Tasks"`

DATE_VALUE

Presentation

Description	Returns the value contained in the specified date field, of the specified object
Type	DATE
French name	VALEUR_DATE
Treatment type	Retrieve Data
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object
3	String	Name of the Date field

Example

Description	To return the Planned Start (PS is the technical name of the field) for project P204:
Class	Project
Formula	`DATE_VALUE("PROJECT","P204","PS")`
Result	`15/08/15`

DURATION_VALUE

Presentation

Description	Returns the value contained in the specified duration field, of the specified object
Type	DURATION
French name	VALEUR_DURATION
Treatment type	Retrieve Data
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object
3	String	Name of the Duration field

Example

Description	To return the duration of the activity named “board meeting for SGC”.
Class	Activity
Formula	`DURATION_VALUE("ACTIVITY","board meeting for SGC","DU")`
Result	`'8d'`

GET_COST_UNIT_VALUE

Presentation

Description	Returns the value of a specified cost unit at a given date.
Type	NUMBER
French name	GET_COST_UNIT_VALUE
Treatment type	Retrieve Data

Arguments

N°	Type	Description
1	String	ID of unit
2	Date	Date to find

Example

Description	To get the value of US dollars for the 1st of July 2000:
Formula	`GET_COST_UNIT_VALUE("USD",'1/7/00')`
Result	`0,69516`
Note	`Cost Unit "USD" is defined as a cost unit in the cost unit breakdown structure.`

Note: In this case the value is expressed relatively to the Euro whose value is 1. This is defined in Pro Web using the object class escalation.

NUMBER_VALUE

Presentation

Description	Returns the value contained in the specified number field, of the specified object
Type	NUMBER
French name	VALEUR_NOMBRE
Treatment type	Retrieve Data
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object
3	String	Name of the number field

Example

Description	To retrieve the actual cost (ACTUAL_COST) of My_Task:
Class	Activity
Formula	`NUMBER_VALUE("Task","My_Task","ACTUAL_COST")`
Result	`120000`

SEARCH_OBJECTS

Presentation

Description	Returns a list of object ID that match a list of multiple attribute values.
Type	STRING
French name	RECHERCHER_OBJETS
Treatment type	Objects & files manipulation

Arguments

N°	Type	Description
1	String	Class of the objects
2	String	List of fields used to filter
3	String	Values of fields

Example

Description	We are searching the list of tasks on a specified project that match value criteria on the attributes Name and description. (See table below)
Class	Activity
Formula	`SEARCH_OBJECTS("Activity","NAME,DESC,PROJECT","T*,Milestone,Project1")`
Result	`"T1,T2,T3,M1"`

Project Data

ID	Name	DESC	Project
T1	T1	Task 1	Project1
T2	T2	Task 2	Project1
T3	T3	Task 3	Project1
M1	M1	Milestone	Project1
T1	T1	Task 1	Project2
T2	T2	Task 2	Project2

Note: This function iterates on a whole class. It may cause performance issues. Please refer to Best Practices for Performance on page 10 for more information.

SELECT_DATA

Presentation

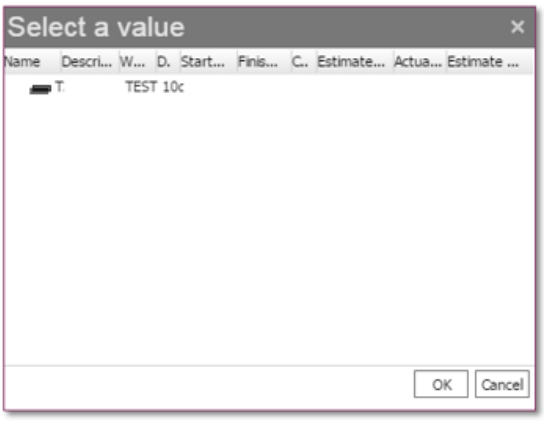
Description	Displays a dialog containing all the objects that belong to the specified class and verify the specified filter. You can then select an object from the list in the dialog.
Type	STRING
French name	SELECTION_DONNEE
Treatment type	User interaction
See also	SELECT_LIST

Arguments

N°	Type	Description
1	String	class
2	String	Filter formula

Example

Description	To show a popup that lists tasks with a duration of 10 days, type the following formula in a report cell for example. The returned value of the formula will be the ID of the object selected in the Popup window.
Class	Activity
Formula	`SELECT_DATA("TASK","DU='10d'")`
Result	`T1`



Select value dialog

STRING_VALUE

Presentation

Description	Returns the value contained in the specified string field, of the specified object, the function returns a list of values if a list of object index is given.
Type	STRING
French name	VALEUR_CHAINE
Treatment type	Retrieve Data
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	ID of the object / List of objects
3	String	Name of the field

Example 1 - Simple

Description	To Return the description of user GALMERAS.
Class	user
Formula	`STRING_VALUE("USER","GALMERAS","DESC")`
Result	`"Gilles Almeras"`

Example 2 - Using Lists

Description	The formula can also be applied to a list of objects
Class	Activity
Formula	`STRING_VALUE("ACTIVITY","A,B,C,D","DESC")`
Result	`"Task A,TaskB,TaskC,Task D"`

USER_IN_GROUP

Presentation

Description	Return TRUE if the user belongs to a group.
Type	BOOLEAN
French name	UTILISATEUR_DANS_GROUPE
Treatment type	Retrieve Data

Arguments

N°	Type	Description
1	String	User ID
2	String	User Groups

Example 1

Description	If the user whose name is Maxence belongs to the group PM, then:
Class	User
Formula	`USER_IN_GROUP("MAXENCE","PM")`
Result	`TRUE`
note	User groups are defined by your administrators. User profiles can also be used as argument of this function.

Example 2

Description	If the user whose name is Maxence belongs to the group PM but does not belong to user group RM, then:
Class	User
Formula	`USER_IN_GROUP("MAXENCE","PM,RM")`
Result	`TRUE`

Example 3

Description	If the user whose name is Maxence belongs to the group PM but does not belong to user group RM, then:
Class	User
Formula	`USER_IN_GROUP("MAXENCE","PM")AND` `USER_IN_GROUP("MAXENCE","RM")`
Result	`FALSE`

Baseline (Reference) Treatment Functions

These functions enable you to retrieve data that is stored in baselines (object class named references in Pro Web).

REFERENCE_BOOLEAN_VALUE

Presentation

Description	Returns the value contained in the specified Boolean field stored in a reference or project version.
Type	BOOLEAN
French name	VALEUR_BOOLEEN_REFERENCE
Treatment type	Baselines / References
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	To tell if Activity was Finished when Baseline1 was created:
Class	Activity
Formula	`REFERENCE_BOOLEAN_VALUE("ACTIVITY",id,"?FINISHED","BASELINE1")`
Result	`TRUE`
Note	Baseline names are defined by your administrator. In this example, there is a baseline named BASELINE1. Additionally, the list of attributes captured when taking a baseline is also specified by the administrator. In this example, attribute "?FINISHED" is captured during the baseline process.

Note: The use of this formula is recommended on the class project, not on activities.

Not all attributes of projects are stored in baselines. To know which ones are, use the `REFERENCE_EXISTS` function (see REFERENCE_EXISTS on page 71). The attributes saved in baselines are defined by your administrator in Pro Web via **History** > **Reference Templates**.

REFERENCE_DATE_VALUE

Presentation

Description	Returns the value contained in the specified date field stored in a reference or project version.
Type	DATE
French name	VALEUR_DATE_REFERENCE
Treatment type	Baselines / References
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	Same than REFERENCE_NUMBER_VALUE but returns a Date that is stored in the baseline.
Formula	
Result	

Note: The use of this formula is recommended on the class project, not on activities.

Not all attributes of projects are stored in baselines. To know which ones are, use the `REFERENCE_EXISTS` function (see REFERENCE_EXISTS on page 71). The attributes saved in baselines are defined by your administrator in Pro Web via **History > Reference Templates**.

REFERENCE_DURATION_VALUE

Presentation

Description	Returns the value contained in the specified duration field stored in a reference or project version.
Type	DURATION
French name	VALEUR_DURATION_REFERENCE
Treatment type	Baselines / References
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	Same than REFERENCE_NUMBER_VALUE but returns a duration that is stored in the baseline.
Formula	
Result	

Note: The use of this formula is recommended on the class project, not on activities.

Not all attributes of projects are stored in baselines. To know which ones are, use the `REFERENCE_EXISTS` function (see REFERENCE_EXISTS on page 71). The attributes saved in baselines are defined by your administrator in Pro Web via **History > Reference Templates**.

REFERENCE_EXISTS

Presentation

Description	Indicates if a field is stored in a reference for one given object, the arguments of the function are the same than for REFERENCE_STRING_VALUE
Type	BOOLEAN
French name	REFERENCE_EXISTE
Treatment type	Baselines / References
See also	REFERENCE_STRING_VALUE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	To tell if the attribute "ESTIMATE_AT_COMPLETION" is stored in "BASELINE1" of the project "PERMANENT-A", type the following.
Formula	`REFERENCE_EXISTS("PROJECT","PERMANENT-A","ESTIMATE_AT_COMPLETION","BASELINE1")`
Result	`YES`

Note: The use of this formula is recommended on the class project, not on activities.

REFERENCE_NUMBER_VALUE

Presentation

Description	Returns the value contained in the specified number field stored in a reference or project version.
Type	NUMBER
French name	VALEUR_NOMBRE_REFERENCE
Treatment type	Baselines / References
See Also	BOOLEAN_VALUE, DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	We want to compare the estimate at completion of the project "PERMANENT-A" to the value stored in a baseline. To retrieve the equivalent of the attribute "ESTIMATE_AT_COMPLETION" of the project inside of the baseline called "BASELINE1", we type the following formula:
Class	Project
Formula	`REFERENCE_NUMBER_VALUE("PROJECT","PERMANENT-A","ESTIMATE_AT_COMPLETION","BASELINE1")`
Result	`12`

Note: The use of this formula is recommended on the class project, not on activities.

Not all attributes of projects are stored in baselines. To know which ones are, use the `REFERENCE_EXISTS` function (see REFERENCE_EXISTS on page 71). The attributes saved in baselines are defined by your administrator in Pro Web via **History** > **Reference Templates**.

REFERENCE_STRING_VALUE

Presentation

Description	Returns the value contained in the specified string field stored in a reference or project version, the function returns a list of values if a list of object index is given
Type	STRING
French name	VALEUR_CHAINE_REFERENCE
Treatment type	Baselines / References
See Also	DATE_VALUE, DURATION_VALUE, NUMBER_VALUE, REFERENCE_BOOLEAN_VALUE, REFERENCE_DATE_VALUE, REFERENCE_DURATION_VALUE, REFERENCE_NUMBER_VALUE, REFERENCE_STRING_VALUE, STRING_VALUE, STRING_VALUE_LANGUAGE

Arguments

N°	Type	Description
1	String	Class of the object
2	String	Object ID
3	String	Field of the object whose baseline value will be returned if it exists.
4	String	Name of the baseline

Example

Description	Same than REFERENCE_NUMBER_VALUE but returns a string that is stored in the baseline.
Formula	
Result	

Note: The use of this formula is recommended on the class project, not on activities.

Not all attributes of projects are stored in baselines. To know which ones are, use the `REFERENCE_EXISTS` function (see REFERENCE_EXISTS on page 71). The attributes saved in baselines are defined by your administrator in Pro Web via **History > Reference Templates**.

Formula Evaluation Functions

EVALUATE_DATE

Presentation

Description	Evaluates a given formula and returns its result as a date on the current object.
Type	DATE
French name	EVALUER_DATE
Treatment type	Code and Formulas evaluation
See also	EVALUATE_DATE, EVALUATE_DURATION, EVALUATE_EXPRESSION, EVALUATE_FILTER, EVALUATE_NUMBER, EVALUATE_STRING, EVAL_EXP

Arguments

N°	Type	Description
1	String	Formula to evaluate

Example

Description	Suppose that you have created a string user attribute called MY_FREE_FORMULA_ATTRIBUTE. Then, insert a formula with the statement below. It will evaluate the text in the additional attribute as a formula, calculate it and return the result as a date.
Formula	`EVALUATE_DATE(MY_FREE_FORMULA_ATTRIBUTE)`
Result	`'26/04/02'`

	Name	Planned start	MY_FREE_FORMULA_ATTRIBUTE	Evaluate date
	TESTPROJ	10/07/15		
	T2	10/07/15	PS -'30d'	10/06/15
	T3	10/07/15	PS + '15d'	25/07/15

EVALUATE_DATE example

EVALUATE_DURATION

Presentation

Presentation

Description	Evaluates a given formula and returns its result as a duration on the current object.
Type	DURATION
French name	EVALUER_DUREE
Treatment type	Code and Formulas evaluation
See also	EVALUATE_DATE, EVALUATE_DURATION, EVALUATE_EXPRESSION, EVALUATE_FILTER, EVALUATE_NUMBER, EVALUATE_STRING, EVAL_EXP

Arguments

N°	Type	Description
1	String	Formula to evaluate

Example

Description	Same as EVALUATE_DATE, with a duration output.
Formula	`EVALUATE_DURATION(MY_FREE_FORMULA_ATTRIBUTE)`
Result	`45d`

EVALUATE_FILTER

Presentation

Description	Evaluates a given formula and returns its result as a Boolean current object.
Type	BOOLEAN
French name	EVALUER_FILTRE
Treatment type	Code and Formulas evaluation
See also	EVALUATE_DATE, EVALUATE_DURATION, EVALUATE_EXPRESSION, EVALUATE_FILTER, EVALUATE_NUMBER, EVALUATE_STRING, EVAL_EXP

Arguments

N°	Type	Description
1	String	Formula to evaluate

Example

Description	Same as EVALUATE_DATE, with a Boolean output. Here below is another example.
Formula	`EVALUATE_FILTER("TFT < '10d'")`
Result	`YES`

Note: In Planisware, the word filter is used to describe a formula that returns a Boolean.

EVALUATE_NUMBER

Presentation

Description	Evaluates a given formula and returns its result as a number on the current object.
Type	NUMBER
French name	EVALUER_NOMBRE
Treatment type	Code and Formulas evaluation
See also	EVALUATE_DATE, EVALUATE_DURATION, EVALUATE_EXPRESSION, EVALUATE_FILTER, EVALUATE_NUMBER, EVALUATE_STRING, EVAL_EXP

Arguments

N°	Type	Description
1	String	Formula to evaluate

Example

Description	Same as EVALUATE_DATE, with a number output.
Formula	`EVALUATE_NUMBER(MY_FREE_FORMULA_ATTRIBUTE)`
Result	3

EVALUATE_STRING

Presentation

Description	Evaluate a given formula and returns its result as a string on the current object.
Type	STRING
French name	EVALUER_CHAINE
Treatment type	Code and Formulas evaluation
See also	EVALUATE_DATE, EVALUATE_DURATION, EVALUATE_EXPRESSION, EVALUATE_FILTER, EVALUATE_NUMBER, EVALUATE_STRING, EVAL_EXP

Arguments

N°	Type	Description
1	String	Formula to evaluate

Example

Description	Same as EVALUATE_DATE, with a Boolean output. Here below is another example:
Formula	`EVALUATE_STRING("DESC + \" - \" + LOCATION")`
Result	`"Consultant - JAPAN"`

Refer to Syntax Rules: Use of Quotation Marks on page 4 for information on the use of \.

Global Variables

Global variables are variable values that can be added to formulas such as the current user, the system date (`DATE_OF_THE_DAY`), the current date (as set according to which projects are currently open), and so on.

These values are the same whatever the object class for the formula.

To call a global variable, you need to use the $ sign in your formula. For example: `$CURRENT_PAGE_OBJECT_ID`.

Name	Type	Label	Category	Description
BUDGET_TO_DATE	Number	Budget to date	Date and duration	Returns the budget to date of the current object.
CURRENT_DATE	Date	Time now	Date and duration	This date, which depends on which projects are opened, can be editted with the *track > change time now* menu.
CURRENT_PAGE_OBJECT_ID	String	Current page object identifier	Objects & files	Returns the ID of the current object. Eg, on a Gantt view, the current object is the Project.
CURRENT_PAGE_REPORT_ID	String	Current page report identifier	Objects & files	Returns the ID of the report being currently displayed.
CURRENT_USER	String	Current user	Objects & files	Returns the name of the current user.
DATABASE_DESC	String	Database description	Administration	Returns the description of the system database.
DATABASE_NAME	String	Database name	Administration	Returns the name of the system database.
DATE_OF_THE_DAY	Date	Time stamp	Date and duration	This is the system date that cannot be modified by users,
EARNED_VALUE	Number	Earned value	Costs	Returns the earned value of the current object.
ESTIMATE_TO_COMPLETE	Number	Estimate to complete	Costs	Returns the estimate to complete of the current object.
FILE_DESC	String	File description	Objects & files	Returns the description of the file. Eg: project description.
FILENAME	String	File name	Objects & files	Returns the name of the file. Eg: project name.
INTRANET	Boolean	Intranet mode	Administration	Returns TRUE if Planisware is used in intranet mode.

Name	Type	Label	Category	Description
OPX2BATCH_MODE	Boolean	Batch mode	Administration	Returns TRUE if the current Planisware instance is in batch mode.
OPX2PRO_MODE	Boolean	Planisware Pro mode	Administration	Returns TRUE if Planisware is used in Pro mode.
PM_BUDGET_THRESHOLD	Number	Budget threshold	Costs	Returns the value of the budget threshold.
PREDICTED_COST	Number	Estimate at completion	Costs	Returns the estimate at completion of the current object.
PROJECT_FINISH	Date	Project finish	Date and duration	Returns theproject finish date.
PROJECT_START	Date	Project start	Date and duration	Returns theproject start date.
TEMPORARY_FILES_DIRECTORY	String	Temporary files directory	Administration	Returns the path of the temporary files directory.
TIME_WINDOW_END	Date	Time window end	Date and duration	Returns the time window start which is common to all projects.
TIME_WINDOW_START	Date	Time window start	Date and duration	Returns the time window end which is common to all projects.
TOTAL_DURATION	Duration	Duration of opened projects	Date and duration	Returns the total duration of the current project.

Examples using Global Variables

Example with DATE_OF_THE_DAY

Description	To view the delay between the planned finish of an activity and today:
Formula	`DIFF_DATE($DATE_OF_THE_DAY,PF,"")`
Result	`99h09m48s`