GILDE SAMENSTERKER

Cursus Teamdesk

LES 03

Formules

Er is in TEAMDESK een enorme hoeveelheid kant-en-klare formules.
De volledige lijst vind je op https://www.teamdesk.net/help/13.3.aspx
In deze les beperken we ons hoofdzakelijk tot de SOORTEN formules, waar je uit kunt putten.

Voor nu heb ik even geen moeite gedaan om alles vanuit het Engels te vertalen. Dat is ontzettend veel werk en als jullie de stof ook zo kunnen 'verteren', scheelt dat heel veel tijd. Mocht dit experiment mislukken, dan vertaal ik het alsnog….

Inhoud

Bouwlokken in Formules

To construct formulas, you may use 5 different types of building blocks.

Literals and Nullvalues

Column references

Variables

Operators

Functions

Literals

These are simple values, for example, `5`, `-10.5` or “`John Donne`”.

Text literals should be enclosed in single or double quotes. If you need to use quote character inside a text literal, use a combination of backslash and quote character. For example: “`John Donne’s \” For Whom The Bell Tolls\”`”.

For checkboxes `true` indicates checked state and `false` indicates unchecked state.

Numeric literals are used as is.

Date, time and timestamp literals should be enclosed in pound signs (#). Date value should be written in year-month-day format using dashes as separators, for example, `#2009-10-30#`. Time value should be written using 24 hour clock and colon as a separator, for example, `#23:30#` or `#09:15:30#`. Timestamp literals are formatted as combination of date and time portion, the value is provided for GMT timezone. For example, #2010-01-01 08:00#.

Duration literals are numbers followed by a single character representing the unit of measurement: `d` for days, `h` for hours, `m` for minutes or `s` for seconds. For example: 0.25d, 6h, 360m or 21600s represent single value – six hours.

Null

Null is the special value indicating that no value was entered in the input field or no value extracted via lookup, for example, when related record was not found. Null does not compare to anything including null, the result of comparison to null is always false; arithmetical and logical operations on null return null as well as most functions. To check whether the value is null, use `IsNull()` function. `Nz()`, `Min()`, `Max()` and `List()` functions provide special treatment for null arguments.

IsNull()

`IsNull` function, as it follows from the function name, is used to check whether a field contains any data, and returns only true or false values.

Since the `IsNull` function returns true or false results, it can be broadly used to set conditions in the condition functions. For example, the `If(IsNull([Company Name], ToUser("Sam Powter"))` formula, may be used for the `Lead Owner` column to assign all leads where the company name is undefined to a specific user – Sam Powter.

The `IsNull` function can revert its meaning if you use "`not`" before the function name. For example, the `If (not IsNull([Project Manager]), "Assigned")` formula used for the `Project Status` column has the following meaning: if the `Project Manager` field in the project record is not empty, then the project status is changed to "`Assigned`".

Nz()

If you want to perform some operations with the field value, but are unsure whether the field is empty or not, use the `Nz` function: if the verified field is empty, this function returns its second argument. In this way you may get round some difficulties with simple arithmetical operations and others. The function is applicable to various field types.

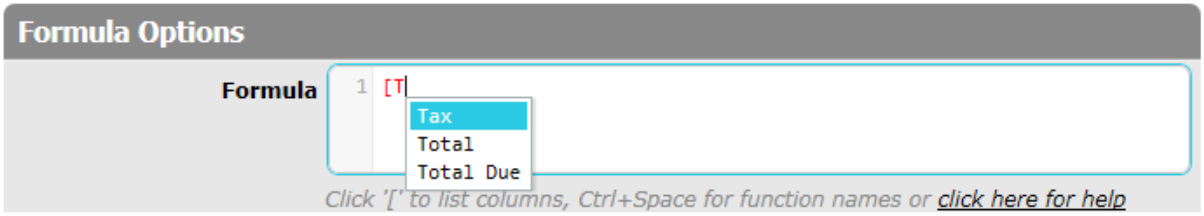
List()

Like `Sum()` function `List()` function ignores null arguments and concatenates remaining values.

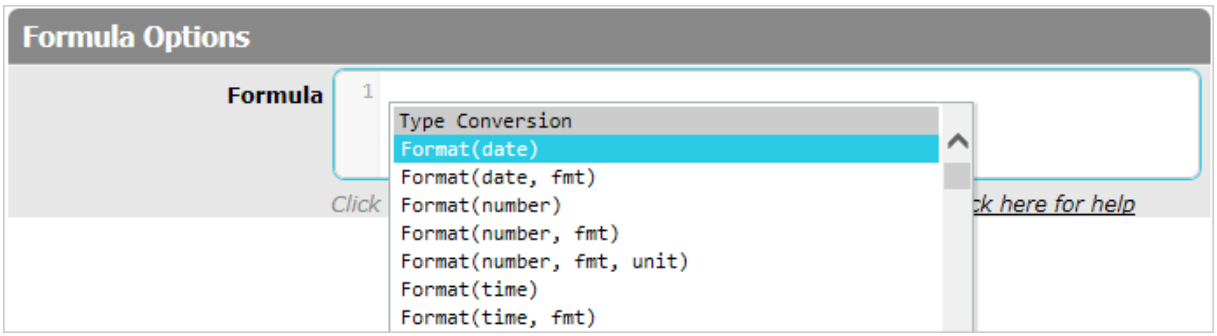
Columns and variables

Column references (as its name states) refer to a certain table column and retrieve column values. When the formula is calculated for a certain table record, the column reference is substituted by a real value taken from the specified column (field) of this record. Column references are enclosed in square brackets and contain the name of the column to which they refer, like `[Client Name]`, `[Budget]`.

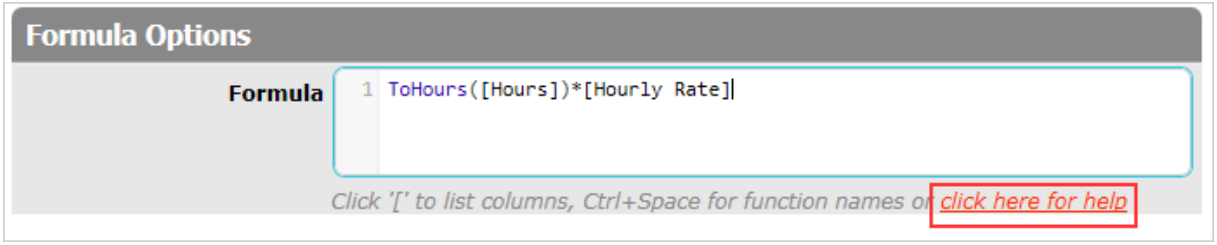
To insert a column reference into your formula, click '`[`' to list all columns of a table and insert a column you need. Also, you can type first letters of the column name after '`[`' sign and the columns starting with these letters will be displayed in the dropdown.



Click `Ctrl+Space` to list function names and select functions you need.



If you need info about the Formula Language Reference, use the `click here for help` link:



To learn more about working with formulas, see Working with Formulas.

Column references are not case sensitive, however, when typing the column name, make sure that it is entered exactly in the way the column is named in the table.

If you use a reference to a certain column and after that change the column name, the name in the formula will be changed automatically. So, there is no need to update all formulas that refer to the modified column.

Variables can also be used for formula creation. In fact, variables were designed to be used in formulas. Variables act much like column references: variable name is enclosed with square brackets (like `[Tax Rate]`). To insert a variable, just click '`[`' to list columns and variables and insert

a variable you need. Also, you can type first letters of the variable name after '[' sign and the variables starting with these letters will be displayed in the dropdown. Als, the variable name can be entered manually and like column references they are case-insensitive.

When working with variables, you should remember that initially all variables are treated as simple text values, so if you want to use them in formulas that process other data types, firstly, you should convert the variable into the necessary data type. Read more on variables in the Variables topic.

Operators and Functions

There are three types of operators that you may use in formulas: arithmetical, comparison and logical. Operators represent a certain simple operation that will be performed over certain values (like in maths, `-` stands for subtraction, `*` for multiplication, and so on.). To view the list of operations, refer to the Formula Language Reference.

Functions, similarly to operators, represent certain predetermined operations that will be carried out when formula result is calculated. Functions consist of a function name and a list of function arguments (separated by commas) in parentheses. Function arguments are values that will be used to calculate the function result; for arguments you may use literals, column references or variables. Functions, like column references, may be entered manually; however, unlike references, they are case sensitive.

For example, the `Sum(number,...)` function is used to sum up several values listed in parentheses. For its two arguments you can use literals, column references or variables. `The Sum([Salary], 300)` function, where `[Salary]` and `300` act as arguments, will add a bonus of 300 to the value retrieved from the `Salary` column.

If needed, functions can be nested within each other.

Conditions

In TeamDesk there are a number of functions used to state conditions. In this topic you may find description of two functions that are used for this purpose more often than others: the `If-` and `Case- functions`.

If-Condition

This type of condition works in the same way as in various programming languages: if the specified condition is true, then one statement is executed, else another statement is executed. The form of the If-function in TeamDesk is the following:

```
If (Boolean condition1, result1, Boolean condition2, result2, ... ,
else-result)
```

The first value in brackets is the condition statement that should answer the `Yes/No` question (that is, the condition can only be true or false). The first and the second function arguments go in pair; these are the condition-result statements, used to define what condition is set and what result should occur if the condition is true. For example, the

`If ([Project Budget] > 1 000 000, "VIP")` formula states that if the project budget exceeds $1 000 000, its type should be changed to `VIP`.

One If-function may contain an unlimited number of the condition-result pairs: all the conditions will be verified one by one until one of them is found true.

The final value in brackets is the else-component. It defines what statement should be executed if none of the specified conditions is true. For example, the `If ([Client's Country] = "USA", 10, 50)` function used in the `[Delivery Cost]` column, states that the delivery cost for domestic (USA) customers is $10, while for all other cases a $50 fee is applied.

The else-statement is optional; if you omit this component, then the else-result will be null. In the `If ([Company Name] = "ABC Inc.", true)` function used for a [Black List] column, a condition may be set to blacklist all leads coming from the ABC Inc. company, while in all other cases lead records are not affected in any way.

When creating condition statements, make sure that the result- and else- statements produce the same type of data, and that the result correlates with the column type.

Case-Condition

This type of function is very similar to the If-function, but is used to verify the condition against multiple cases. Using multiple conditions in the If-function makes you repeat the same value several times, for example, `If( [Client's Country] = "USA", 10, [Client's Country] = "Canada", 20, [Client's Country] = "Ukraine", 30, …)`.

To avoid this repetition, you may use the Case-function, which has the following form:

```
Case (condition expr, value1, result1, value2, result2, ..., else-result)
```

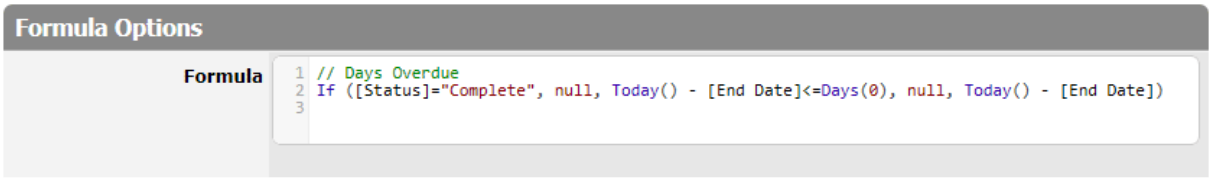
Here you have to state the condition (actually the first part of it) only once, and then enter the condition-verified value and the result. Again, the number of the condition value-result pairs is unlimited. During the process of calculation, the system checks whether the first conditional part equals any of the defined verified values and produces a corresponding result.

The example above may be written in the following form: `Case ( [Client's Country],"USA", 10, "Canada", 20, "Ukraine", 30, 100)`, that is if client's country is USA, the rate is 10, if Canada – 20, if Ukraine – 30, while for all other countries the rate is 100.

Creating Formulas

Formulas can be inserted into the specific formula fields that are found in various forms of the database setup area. When you create formulas for certain columns, the result is calculated for each record in the table.

At the bottom of the formula entry field there are hints how to choose and insert building blocks into the entry field. Click '[' to list columns, click `Ctrl+Space` for function names. If you need info about the Formula Language Reference, use the `click here for help` link.



Formula editor performs syntax highlighting, brace matching. Also it provides suggestions and auto-completion for column and function names.

Basic syntax validation and report malformed pieces of code are highleted in red. Comments are green, known function names are blue, constants are dark red.

Find below a description of aspects that should be taken into account when formulas are created.

Data and Column Types

If the calculated formula result is stored in a table field, make sure that resulted data correlate with the field/ column type: for example, a simple `2*3` formula produces a numeric result `6` which should be stored in a numeric field – you cannot have the numeric result stored in the `Date` or `Timestamp` column (though, in some cases data conversion may help).

When writing formulas, try to use data of the same (or correlated) type. That is, you cannot subtract a numeric value from a date, as that does not make sense, while subtracting project start date from the project end date in the `[Project End Date] - [Project Start Date]` formula will work well.

The same applies to operations that you use: not all of available operations may be applied to all types of data. For example, you will not be able to multiply two text strings or add any value to a valid e-mail address, as this is simply pointless.

Data Type Conversion

Sometimes when creating formulas, you may need to pull together two values that have different data types. For example, to calculate an approximate project budget, you want to multiply the number of planned hours stored in the `[Hours Planned]` field by the hourly rate expressed by the `[Hourly Rate]` variable. Since all variables are treated as text, you cannot write the formula simply as `[Hours Planned] * [Hourly Rate]` – numbers cannot be multiplied by text.

To run this operation successfully, you will need to convert the variable to a numeric value. In TeamDesk this is done with the help of specific `Type Conversion` functions. These functions can be found under the Type Conversion section (or under specific data type sections) of the functions list.

The name of the conversion function usually says to what type the data will be converted; text in brackets shows with what source of data the function works. For example, `ToDate(timestamp)` function is used to convert timestamps to the date format, `ToNumber(text)` – to convert text values or literals to numeric format, and so on.

The example, described above, should be written in the following way to work properly: `[Hours Planned] * ToNumber([Hourly Rate])`.

Operation Sequence

If your formula contains more than one operation, for calculating the result a standard algebraic notation is used. Here you should take into account a standard priority of operations. To calculate the priority, you may use the following table:

Operator	Priority
Parenthesis	Highest priority
NOT logical operator	
Power operator	
Multiplication and division	
Addition, subtraction and concatenation	
Comparison operators	
AND logical operator	
OR logical operator	Lowest priority

The following rules should be taken into consideration as well:

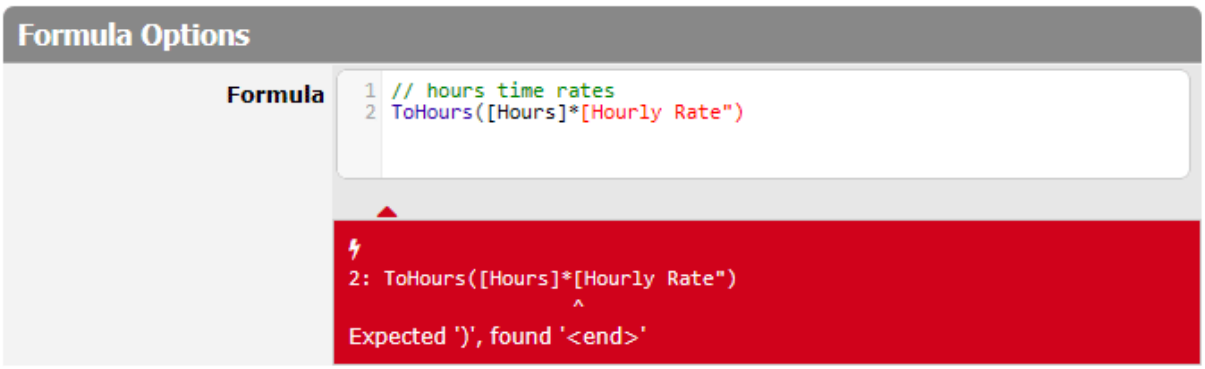
- Unary operators, like `-`, `+` or `not` have the highest priority (for example, in the `10*-5` formula, first 5 will be converted to `-5`, and then the multiplication will be carried out).
- Simple operations of the same priority are performed from left to right. In the `2+3-4` formula, first `3` will be added to `2`, and then `4` will be subtracted from the sum result.
- Operations of different priority will be performed in the order of their priority correspondingly. In the `3*5 – 4/2` formula, first `3` will be multiplied by `5`, then `4` will be divided by `2`, after which two resulting values will be summed.

- If you use parentheses, operations enclosed in parentheses will be performed first. If you use nested parentheses, for example, `10 - (2*(5+2))`, then the operation in the deepest parentheses – `(5+2)` – will be performed first.

Formula Errors

Creating formulas, you will most probably come across various types of errors: like syntax errors when the formula is not arranged correctly or unaccepted characters are used; validation errors, like in cases when formula refers to incorrect fields; or evaluation problems.

Basic syntax validation and report malformed pieces of code are highlighted in red. Messages about errors that your formula may contain are displayed below the formula entry field: if the system detects a specific incorrect formula part, it marks such a part with the `^` character pointing to the erroneous part (see example below):



Some errors (like division by zero) cannot be tracked at the stage of formula creation. Such errors are revealed when formulas are calculated and should be eliminated as soon as errors are found.

FORMULA-COLUMNS

Together with table columns containing simple values (for example, of the numeric or text type), TeamDesk allows a user to create columns with formula values. The value for such columns is set as a result of configuration or calculation by predefined formulas.

Formula-defined columns are created in the same way as columns of ordinary types (see the Creating new columns section). For columns that contain formulas, you may use the following data types:

Kolomtype	Omschrijving
Formula–Text	This type of columns allows formulas that provide a text result. The formula may contain a text literal, variable or references to columns that store text values. For example, in the “Name” formula-text column you can combine a first name value and a last name value kept in corresponding columns. In this case the List function is helpful: List(“ ”, [First Name], [Last Name]).
Formula-Number	This type allows formulas that provide a numeric result: integer, decimal, positive or negative. The formula may contain actual numbers (literals), references to columns that store numeric values or variables that should be converted first using the ToNumber () function. If you need to calculate the Age using the Birthdate value, insert the following formula into the formula-numeric column: Year(Today()) - Year([Birthdate]) - If(DayOfYear(Today()) < DayOfYear([Birthdate]), 1, 0)
Formula-Date	This type allows formulas that provide a numeric result: integer, decimal, positive or negative. The formula may contain actual numbers (literals), references to columns that store numeric values or variables that should be converted first using the ToNumber () function. If you need to calculate the Age using the Birthdate value, insert the following formula into the formula-numeric column: Year(Today()) - Year([Birthdate]) - If(DayOfYear(Today()) < DayOfYear([Birthdate]), 1, 0)
Formula-Time	This type allows formulas that are used to display time of a day. The result may be created by retrieving time from a Timestamp column or by constructing time with the help of literal values or column references.
Formula-Duration	This type allows formulas that are used to calculate a certain period of time. To calculate a duration formula, you may choose from the list of Duration functions. In addition, a

	duration formula may be composed by subtracting one value from another: • Timestamp from Timestamp • Date from Date • TimeofDay from TimeofDay The calculation result may be positive or negative.
Formula-Checkbox	Checkbox formula always provides the ‘ Yes’ (check box is selected if the expression is true for the record) or ‘No’ response (check box is not selected if the expression is false). For example, when tracking project deadlines, the [Project End Date] < Today() formula in the Deadline Expired column, shows whether the team fails to meet a deadline (Yes) , or there is still time left for development (No) .
Formula-Phonenumber	This type allows formulas that are used to make up a phone number out of the area code, phone number and maybe extension. Use text values for formula arguments: literals, variables or column references. For example, in the List(“-” , [Area Code], [Phone Number], [Ext.]) formula, area code, phone number and extension number values are retrieved from corresponding columns. TeamDesk parses data from Phone and Formula-Phone columns. The system decorates them as Skype links on desktops and tablets and as links to invoke phone dialer on mobile phones. Skype requires phone numbers to contain country code. If the column data is missing country code, TeamDesk infers the code from database Language and Locale settings. When a text is entered and stored in a Phone Number column, it is decorated as a link. Also, there is the tooltip that displays the phone number formatted according to your country rules.

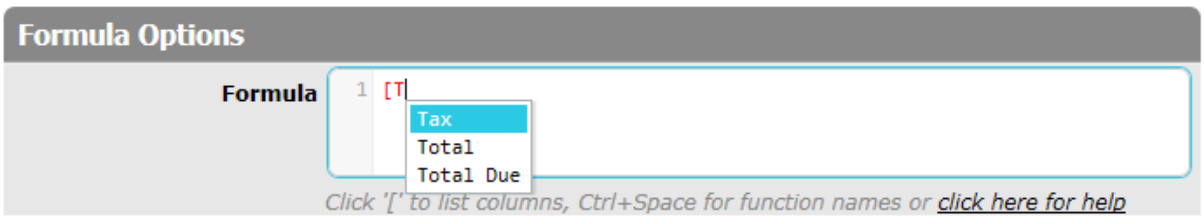
Formula-E-mailaAddress	This type allows formulas used to create an e-mail address either by entering a textual literal in double quotes directly (for example, “ info@mywebsite.com ”) or by drawing up various column values together (for example, List(“@“, [Local E-mail Part], [domain])). Use text values for formula arguments. In the user mode, the calculated formula result will be represented as a mailto hyperlink used to create a new e-mail message that will be sent to the specified recipient.
Formula-URL	This type of formulas, like formulas for the E-mail Address field, can be made up either by entering a textual literal directly (“ www.teamdesk.net ”) or by composing the URL from constituent parts, like variables, literals, column references and other text values. Additionally, the following functions may be used to construct a URL formula: URLRoot () , BackURL () , AppId () , TableId () , RecordId () .
Formula-User	This type of columns allows to provide names of registered database users. The formula may contain column references or text literals converted into the required data format.
Formula-Barcode	Any text value can be displayed as a barcode. The formula-barcode column allows to generate barcodes that can be inserted into the document and scanned.
Formula-Location	This type of column options match the Location column. Also, there are couple of functions you use with locations: • IsNull(location) checks whether location field is blank. • Nz(location, location, …) – to choose first non-blank location from the list • ToLocation(latitude, longitude) – constructs the location from a pair of coordinates • ToLocation(text) – constructs the location from text (comma separated pair of decimal numbers) • Latitude(location) – extracts latitude • Longitude(location) – extracts longitude • Distance(location, location, “unit”) – calculates distance between two locations on elliptic Earth. Result is returned in meters, by default or according to the unit provided: “m” for meters, “km” for kilometers or “mi” for miles. Finally, there is a DeviceLocation() function for use in place of pair DeviceLatutude() and DeviceLongitude() in actions designed for mobile apps.

After a column of a required type has been created, it is necessary to specify the formula itself in the properties of a created column.

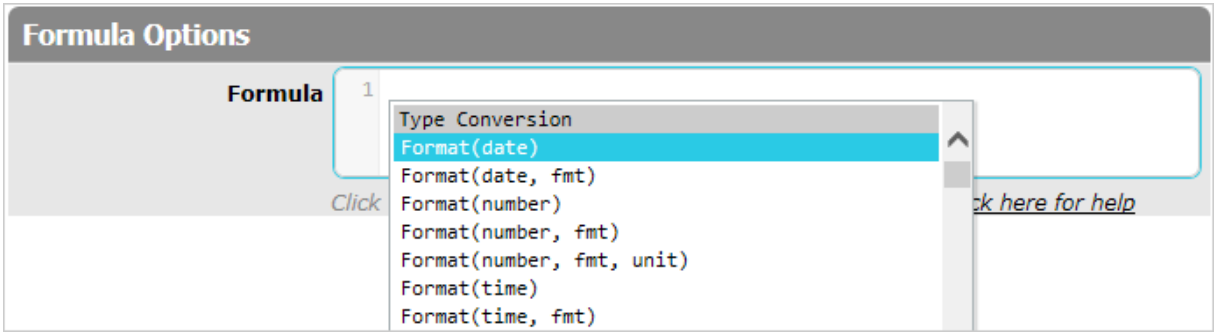
To set a new formula:
Click the Setup link at the top right corner of the window.
Select a table you need.
From the setup menu select Columns > Customize existing columns . The system will display the Columns form;

Click the Edit button next to a newly created formula defined column;
In the Formula field set a new formula according to which the value of a column will be set.

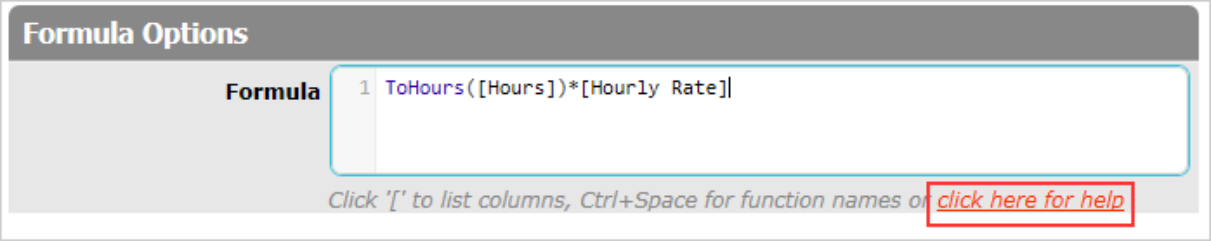
Click '[' to list all columns of a table and insert column variables you need. Also you can type first letters of the column name after '[' sign and the columns starting with these letters will be displayed in the dropdown.



Click Ctrl+Space to list function names and select functions you need.



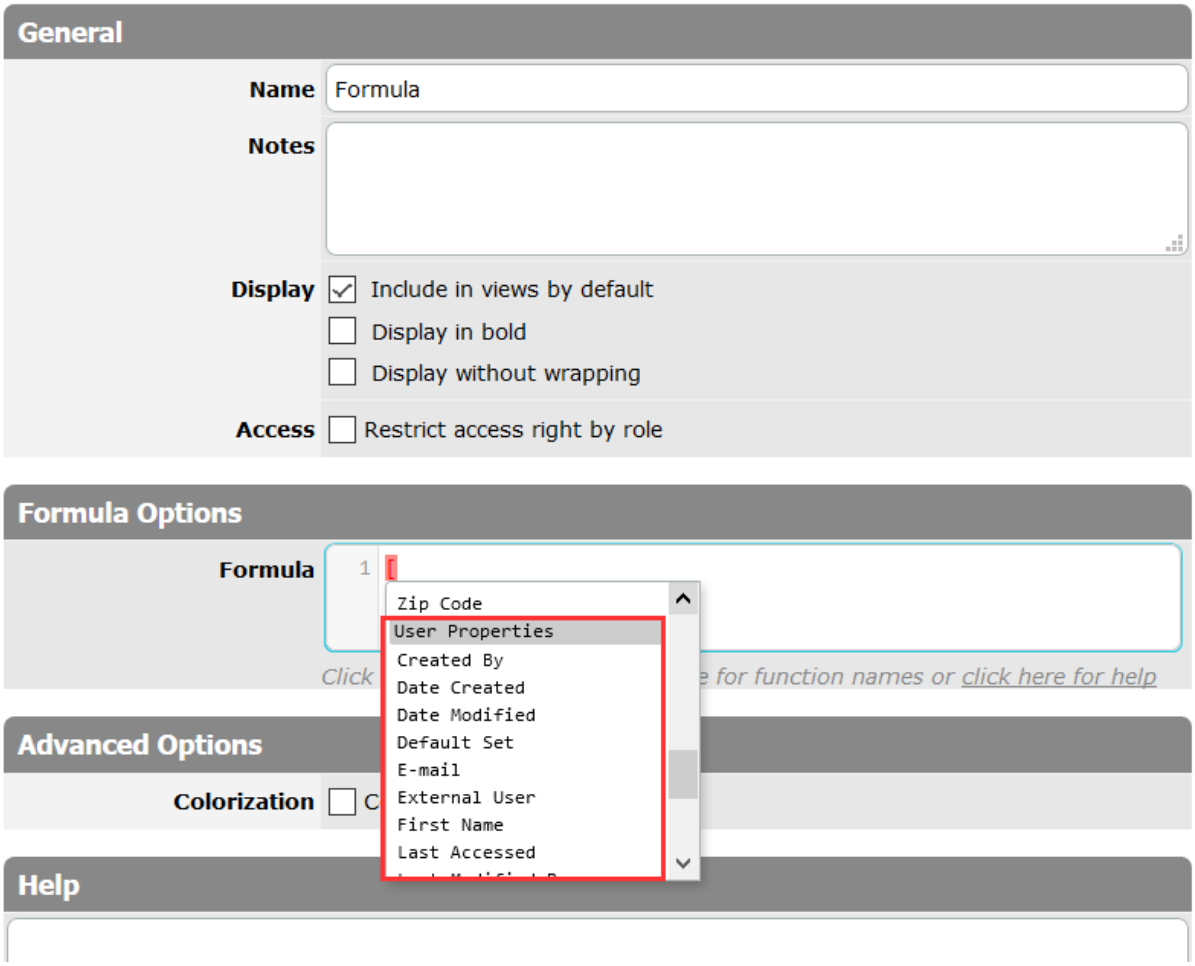
If you need info about the Formula Language Reference, use the click here for help link:



To learn more about working with formulas, see Working with Formulas.

To view a full list of functions used for formula generating, see Formula Language Reference.

Wherever you need information from User Properties related to a current user, you can simply refer to a column of the User Property table by its name. Just click '[' to list all columns of a table and choose a column from the User Properties section. Also, you can type first letters of the column name after '[' sign and the columns starting with these letters will be displayed in the dropdown.



You can colorize values stored in the Formula-Text columns. Just use the Colorization checkbox displayed in the Advanced Options. Colorization bases on whether the value contains certain text. More details are described in the Column Value Colorization section.

The last option is the Help field.

When this Help text property is filled in, the question mark is displayed near the field name on the View/Edit form. If you point to this question mark, the pop-up with a tip or a help text message will be shown to a user.

VOORBEELDEN

Even als voorbeeld een paar van de honderden formules in detail:

Operator: &
Je kunt met & teksten aan elkaar plakken:

[Voornaam] & “ “ &
If(IsNull([Voorvoegsel]), “”, [Voorvoegsel] & “ ”) &
[Achternaam]

[Achternaam] & ", " & [Voornaam] & " " & [Voorvoegsel] ➔ **Graaf, Jan van der**
Maar…. Als het voorvoegsel LEEG is, zoals bij **Jense, Ton** gaat het mis: Je ziet dan geen resultaat. Je moet dan de functie Nz() gebruiken om het correct te laten werken
[Achternaam] & ", " & [Voornaam] & " " & **Nz(**[Voorvoegsel]**)**

Er vanuit gaande dat Achternaam en Voornaam ALTIJD een waarde hebben, want anders moet je ook daar de NZ-functie gebruiken.

Stel dat bij Ton Jense de voornaam niet gevuld is, dan zou

[Achternaam] & ", " & [Voornaam] & " " & **Nz(**[Voorvoegsel]**)** ➔ **(NIETS)**

[Achternaam] & ", " & **Nz(**[Voornaam]**)** & " " & **Nz(**[Voorvoegsel]**)** ➔ **Jense,**

Functie: Contains()
Parameters: (Text u, Text v)
Resultaat: Boolean
WAAR als v gevonden wordt in u, anders ONWAAR

Contains("test", "es") ➔ WAAR

Contains([Merk], "Ford") ➔ WAAR als de tekstkolom Merk de waarde “Ford” bevat

Contains("test", "et") ➔ ONWAAR

Functie: Left()
Parameters: (Text t, Number n)
Resultaat: Text
De eerste n tekens in de string t.

Left([Postcode],4) ➔ 2717 (bijvoorbeeld)

Functie: Lower()
Parameters: (Text t)
Resultaat: Text
Converteert tekst t naar klein kapitaal

Lower("TonJense@GildeZoetermeer.nl") ➔ tonjense@gildezoetermeer.nl

Functie: If()
Parameters: (Conditie, Resultaat bij WAAR, Resultaat bij ONWAAR)
NESTEN: If(conditie1, WAAR, conditie2, WAAR, Conditie3, WAAR, ONWAAR)

Bijvoorbeeld: If([Leeftijd] > 65, “Senior”, [Leeftijd]<=14], “Junior”, “Lid”)
Of: If([Geslacht] = “M”, “de heer”, “mevrouw”)

Functie: Today()
Parameters: Geen
Resultaat: 26-06-2020

Functie: YearsBetween()
Parameters: (Einddatum, Begindatum)
YearsBetween(Today(),[Geboortedatum])
In mijn geval wordt dat dan: **68**

Operator: &
Je kunt met & teksten aan elkaar plakken:

[Voornaam] & “ “ &
If(IsNull([Voorvoegsel]), “”, [Voorvoegsel] & “ ”) &
[Achternaam]

Functie: Nz
Negeert lege (NULL) waarden. Anders krijg je
OF: [Achternaam] & ", " & [Voornaam] & " " & **Nz(**[Voorvoegsel]**)**

Formula-XHTML

Formula – XHTML is een heel andere divisie….. Het is een eigen taal om zaken in de opmaak voor elkaar te krijgen. Het is best lastig om te leren en ook wel beperkt, maar soms heb je het nodig. Weet dat het bestaat en ga het pas gebruiken als je wat verder bent met Teamdesk. Maar het hoort wel bij “formules”, vandaar dat je het nu al van me krijgt….

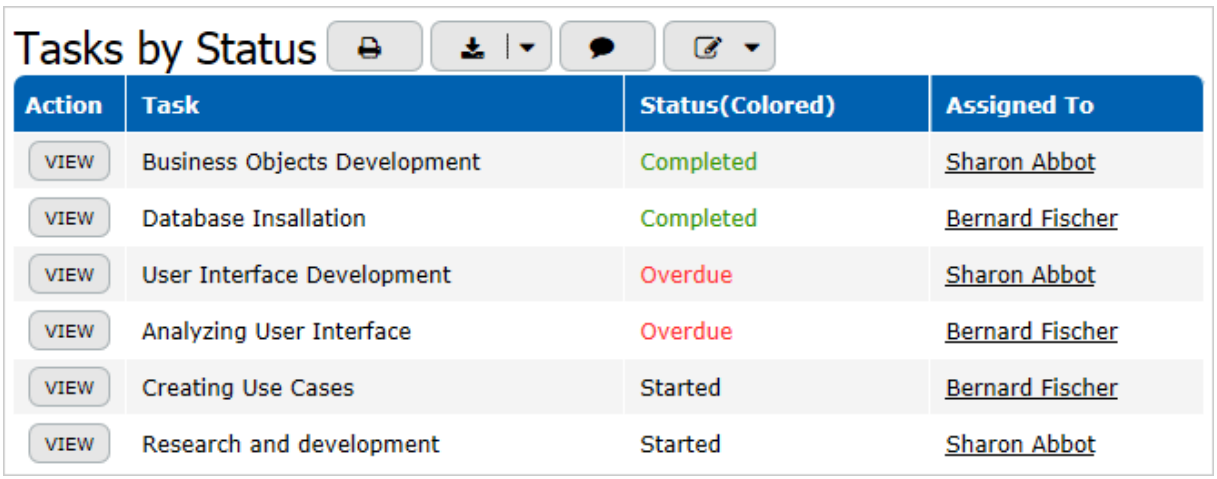
Formula-XHTML provides the way to decorate column values with tags and styles. It is a mixture of limited (X)HTML markup and TeamDesk formula language.

Using Formula – XHTML you can write HTML markup as is, decorating TeamDesk formula pieces (code points) with <%…%> marks; those of you who dealt with ASP should be familiar with the syntax. Each formula piece is evaluated; the result of evaluation is properly encoded and inserted in the markup.

Code points can appear inside attribute values and/or in tag contents, no dynamic tag generation is allowed. We designed it this way to ensure the formula result is a well formed HTML and won't affect the rest of the page's content.

Here are typical usage examples for the formula:

Example 1
Color the status value “green” if it is “Completed”, “red” – “Overdue”, “black” - if any other statuses are selected.

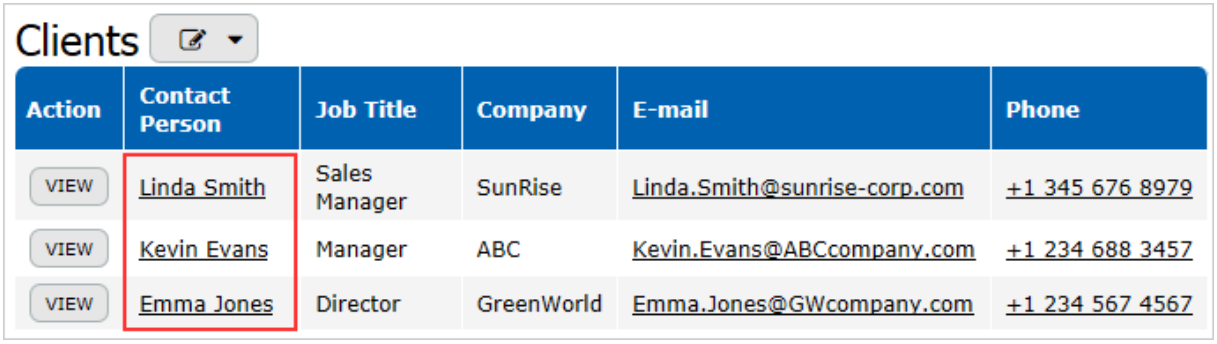


Action	Task	Status(Colored)	Assigned To
VIEW	Business Objects Development	Completed	Sharon Abbot
VIEW	Database Insallation	Completed	Bernard Fischer
VIEW	User Interface Development	Overdue	Sharon Abbot
VIEW	Analyzing User Interface	Overdue	Bernard Fischer
VIEW	Creating Use Cases	Started	Bernard Fischer
VIEW	Research and development	Started	Sharon Abbot

You can add a new Formula – XHTML and name it “Status(Colored)”. This column can be added to a Table View instead of the existing “Status” text column that can be used only for record editing. When the Formula – XHTML column is created, you can enter the following formula there:

```
<font
color="<%Case([Status],"Completed","#339900","Overdue","#ff3333","#000000")%>"
><%=[Status]%></font>
```

Example 2
Decorate contacts name as an e-mail link:



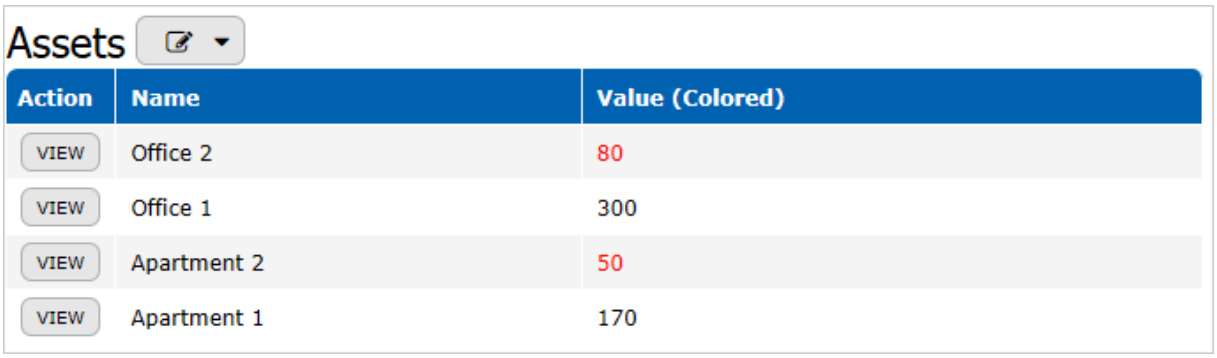
Clients

Action	Contact Person	Job Title	Company	E-mail	Phone
VIEW	Linda Smith	Sales Manager	SunRise	Linda.Smith@sunrise-corp.com	+1 345 676 8979
VIEW	Kevin Evans	Manager	ABC	Kevin.Evans@ABCcompany.com	+1 234 688 3457
VIEW	Emma Jones	Director	GreenWorld	Emma.Jones@GWcompany.com	+1 234 567 4567

Create the “Contact Person” Formula-XHTML column, where the existing “Name” column is associated with a corresponded E-mail listed in the E-mail column.
Enter the following formula:
<a href="<% Nz("mailto:" & [E-mail]) %>"><% [Name] %>

Behind the scenes Formula – XHTML creates a text formula: a concatenation of markup code and formula results enclosed in HTML encoding function, such as the code below produced from the example:
"" & Encode([Name]) & ""

Example 3
Color the value red if it is less than 100:



Assets

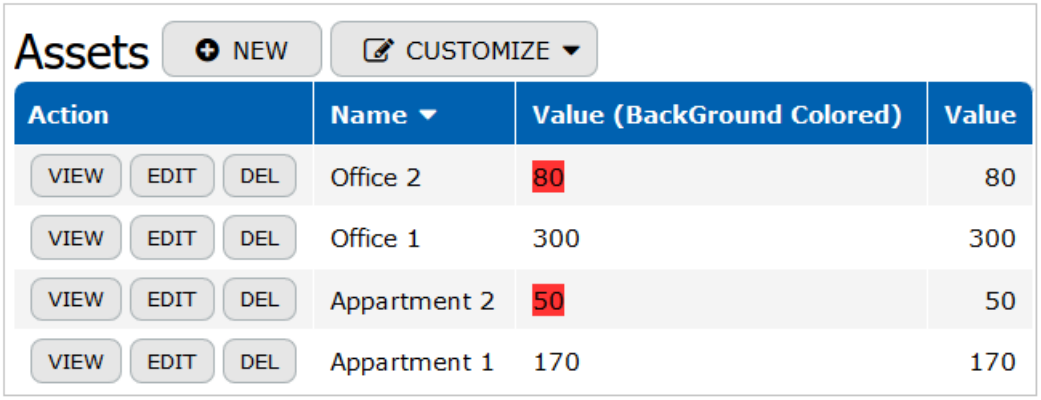
Action	Name	Value (Colored)
VIEW	Office 2	80
VIEW	Office 1	300
VIEW	Apartment 2	50
VIEW	Apartment 1	170

Create the “Value (Colored)” Formula-XHTML column, where the Values should be colored.

Enter the following formula:

<span style="<% If([Value] < 100, "color:red", "") %>">
<%= [Value] %>

Example 4
Color the value background in red, if it is less than 100:



Create the “Value (BackGround Colored)” Formula-XHTML column, where the Values should be colored.

Enter the following formula:

```
<span style="background-color:<%If([Value]<100,"#ff3333","#FFFFFF")%>">
<%= [Value] %>
</span>
```

Be aware of the way concatenation handles NULL values – if one of the parts is NULL whole result is also NULL – first example produce no markup at all when the value of [Name] is blank. While it is useful in some scenarios, enclosing each and every field you want to display in a sort of NULL checks is a cumbersome task.

To handle this case we created a shorthand for code points containing a sole reference to a column, a <%= [Column] %>. It creates Nz() wrapper so that NULL values produce empty strings that can be safely concatenated; numbers, dates, times and timestamps are converted to text via locale-aware Format() function, other types converted via ToText(). In second example, <%= [Value] %> produces Encode(Nz(Format([Value]))).

The usage of HTML and Formula – XHTML is somewhat limited due to technical reasons.

1. Markup will be displayed as text in:
    a. Dropdown lists, as dropdowns do not support markup;
    b. Calendar views and Lookup columns with "Display As Link" options checked, as very limited set of tags can be displayed inside of link;
    c. Documents as the Word file format have no connection to HTML;
2. Some tags are disabled to avoid interference with the rest of the page:
   <html>, <head>, <script>, <noscript>, <title>, <meta>, <link>, <base>, <style>, <body>, <form>, <input>, <isindex>, <textarea>, <select>, <optgroup>, <option>, <fieldset>, <legend>.
   <button> tag is allowed only when its type attribute is set to "button".
3. All allowed tags require closing tag, even if it is not required by HTML. For example, in HTML you can write:
   First paragraph
   Second paragraph

   In formula code you should closing tag before opening new tag:
   First paragraph
   Second paragraph
4. Tags that cannot have inner content, such as <img>,
, <hr> and <param> should be self-closed, e.g. written as
 or

Huiswerk

Voeg de volgende formulevelden toe in de tabel Donateurs:

- Naam (Achternaam, spatie, voornaam, spatie, voorvoegsel)
  Je kunt hier de functie LIST gebruiken
- Adres (straat, huisnummer, komma, postcode, plaats)
  Doe dit met LIST of plak alles met & achter elkaar
- Postcodegebied (Eerste vier cijfers van de postcode)
- Leeftijd (gebruik de geboortedatum en de huidige datum)
- Wijk (spiek maar even in de tabel LEDEN)