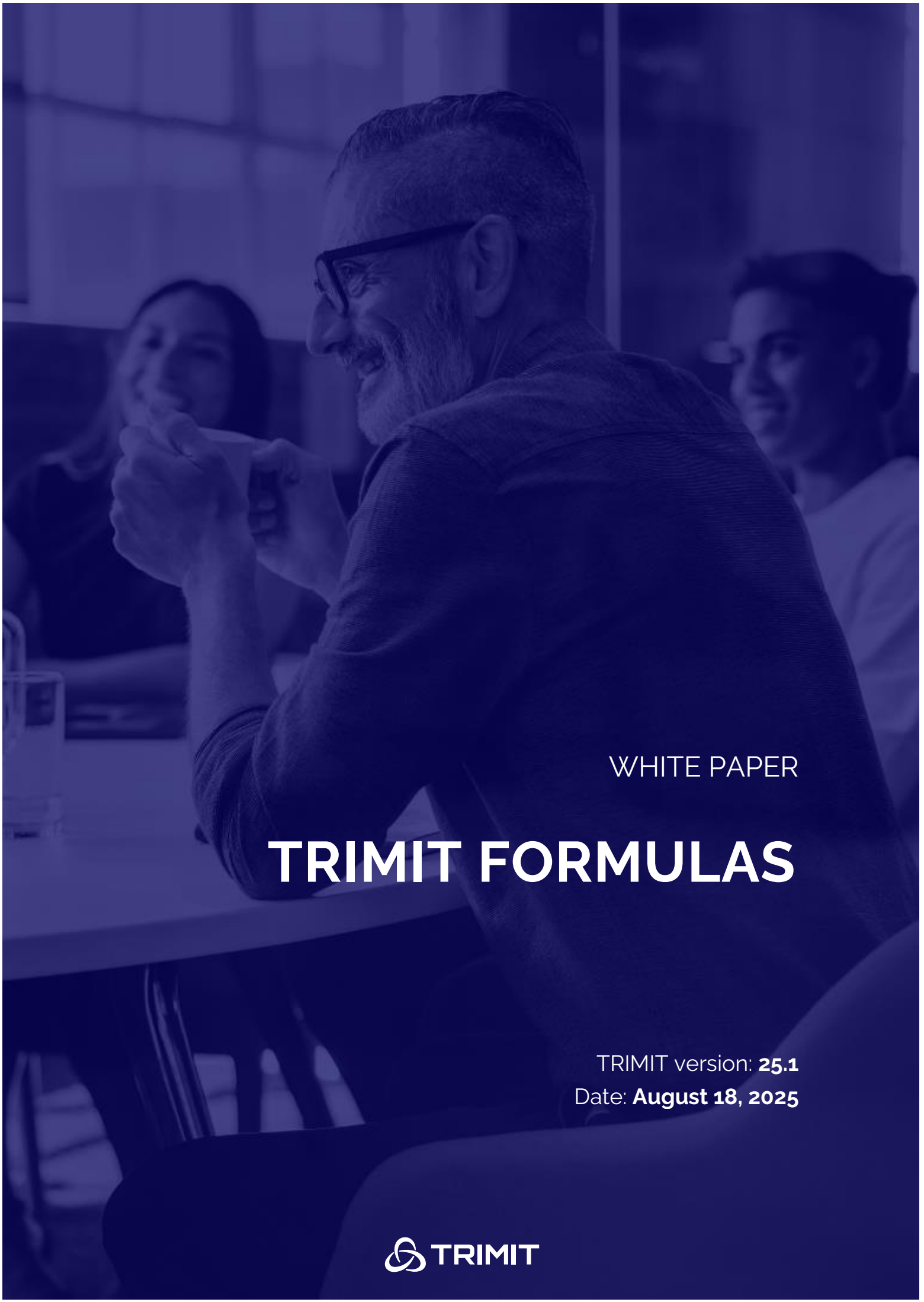


WHITE PAPER

TRIMIT FORMULAS

TRIMIT version: **25.1**

Date: **August 18, 2025**

TRIMIT

Table of Contents

Introduction

This document will describe settings and functionality regarding **Formulas**.
For more elaborate examples and business cases, there is a document called *TRIMIT Formula Cases*. (to be released; under construction)

The audience for this document is the Consultant or very skilled User. It will only describe the TRIMIT Fields/functionality and not the standard Business Central Fields/functionality. Be aware of not changing settings and parameters in a live database without consulting the implementing partner.

The functionality described is the functionality as **TRIMIT 25.1 for Business Central W1 BC26.X**.
The pictures in this White Paper are based on the demo data for **TRIMIT 25.1,** which is based on Microsoft Dynamics 365 Business Central 2025 Release Wave 1, in the demo company **CRONUS TRIMIT W1 Ltd.** (based on CRONUS International Ltd. of Microsoft Dynamics 365 Business Central).

Components

TRIMIT Formulas functionality contains the following objects:

The tables

6036311	**trm Temp Tree View Variable**
6036313	***trm Temp Select Primary Key***
6036316	***trm Temp Form Misc Character***
6036430	***trm Call Formula Processing***
6036502	***trm Caption Class Setup***
6037105	***trm VarDim Order***
6037106	***trm VarDim Item***
6037139	***trm Caption Class Caption***
6037196	***trm API Formula Function***
6037197	***trm API Parameter***
6037198	***trm Formula Function (Legacy)***
6037200	***trm Formula Constant***
6037201	***trm Formula Global Variable***
6037202	***trm Formula VarDim Field Ref***
6037203	***trm Search Table Header***
6037204	***trm Search Table Line***
6037205	***trm Formula Header***
6037206	***trm Formula Line***
6037207	***trm Formula Cross Calc Setup***
6037208	***trm Filter Formula Item***
6037209	***trm Filter Formula Sales***
6037210	***trm Filter Formulas Production***
6037211	***trm Filter Formula Purchase***
6037212	***trm Formula Dialog***
6037214	***trm Formula Log***
6037216	***trm Formula Table Relations***
6037308	***trm VarDim Prod./Purchase Line***
2000000041	***Field***

The pages

6036506	*trm Fields List*
6036513	*trm Caption Class Setup*
6037027	*trm Formula Functions*
6037139	*trm Formula Select Primary Key*
6037140	*trm Form VarDimProd FieldRef*
6037141	*trm Form VarDimPurch FieldRef*
6037142	*trm Caption Class Captions (List)*
6037200	*trm Search Table Header (Setup Card)*
6037201	*trm Search Table Headers (List)*
6037202	*trm Formula VarDimItemFieldRef*
6037203	*trm Formula Header List*
6037204	*trm Formula Exp AssistEdit*
6037205	*trm Constants*
6037206	*trm Formula Global Variables (List)*
6037207	*trm Formula Functions*
6037208	*trm Search Table Lines (List)*
6037209	*trm Form VarDimOrder FieldRef*
6037210	*trm Formula Cross Calculations (Card)*
6037212	*trm Formula Dialog List*
6037217	*trm Formula Table Relations*
6037219	*trm Temp Formula Misc Char (List)*
6037221	*trm Formula Log (List)*
6037222	*trm Formula All Lines (List)*
6037223	*trm Formula Header (Card)*
6037224	*trm Formula Lines Subform*
6037226	*trm Temp Global Variables (List)*
6037800	*trm API Parameters*

The reports

6036805	*trm Test Formula Line*

The codeunits

6036501	*trm Glob param Single Instance*
6037214	*trm Formula Set and Get Parameters*

Introduction

The basis of the TRIMIT solution is the **Configurator**. With this Configurator the system can manage Items that come in many variations from a basic model. In TRIMIT we call this model the **Master**. There could be many sorts of variations. A variation could be a:

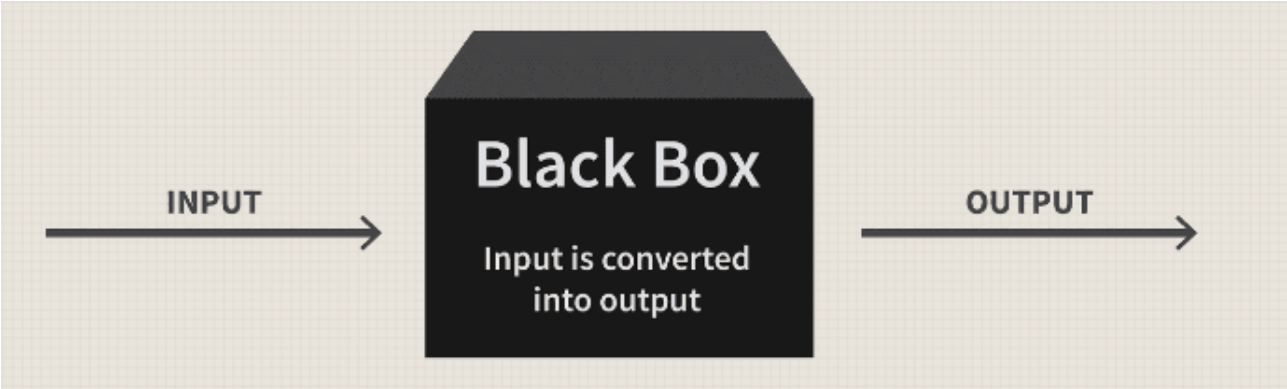
- *Variant* (color, material type, size, shape, treatment, etc.)
- *Measurement* (length, width, height, thickness, depth, etc.)
- *Options* (presence of a particular component, for instance a shelf)
- *Position* (left, right, middle, top, bottom, etc.)
- *Quantity* (number of shelves, drawers, pillows, etc.)

Handling these variations requires **Configuration Rules**[1]. In TRIMIT we have several ways to set up these Configuration Rules. For calculating Unit Price components based on Variants we can use **VarDim Type Adjustments**. In some specific cases in a Bill of Materials (BOM) structure we can use **Consistent Inheritance**[2] in which we inherit the variant of a Product to the variant of its Component (for instance: a red Shirt consists of a red Fabric). If there is a simple relationship between the variant of the main Product and its Price or its Component, we can use **Search Expression** in combination with **Search Tables**[3]. If the Configuration Rules are more complicated TRIMIT has a powerful tool called **TRIMIT Formulas**, which is very flexible and can handle complex calculations.

In this document we focus on **TRIMIT Formulas**, although other Configuration Rules functionality is discussed to some extent.

TRIMIT Formulas is a script language that enables a user, without programming experience, the opportunity to retrieve, change and update Data. Because **TRIMIT Formulas** are written and saved as ordinary Text in a Table, the update of Business Central and TRIMIT can be done without affecting the Configuration Rules that are described by using **TRIMIT Formulas**.

TRIMIT Formulas can be used to describe the Configuration Rules regarding any specific object in BC/TRIMIT. The idea is that Value(s) from Field(s) from a specific Table(s) are put into a Formula (Line). That Input Value is converted into an Output Value that will be placed back into (another) Field of a specific Table. This is schematically shown below.



[1] In other configuration Solutions **Configuration Rules** can be called **Business Rules**.
[2] Although Consistent Inheritance have nothing to do with Formula, it is discussed in this document: chapter Search Table -> paragraph Consistent Inheritance.
[3] In case of Search Expression and Search Tables the system auto-creates a Formula. This is described in the paragraph Search Table Automation.

Especially when used in connection with a BOM Structure, **TRIMIT Formulas** is a powerful tool. With the **TRIMIT Formulas**, variants, positions, quantities, and/or measurements can be calculated of the Component Lines in a BOM. The BOM Structure plays an important role here, so there is a separate chapter about how to analyze this, so that complex BOM Structures can be built with Formulas.

But with **TRIMIT Formulas**, Sales and Purchase Prices can also be calculated. **TRIMIT Formulas** can also be of support during the configuration process. And there are many other applications. In this document we discuss many of these in examples and cases.

TRIMIT Formulas are created and maintained primarily in two tables ***6037205 trm Formula Header*** and ***6037206 trm Formula Line***. Any number of Formula Lines can be attached to one Formula Header. In certain situations, Formula Lines can be created without a relation to a Formula Header (i.e., Formula Lines used in Calculation Templates). In other situations, the Formula Header is created automatically (for instance in connection with using auto generate of Formulas on Item/Master BOM Component Lines). A set of Formula Lines of the same Formula Type and Number combined make up a TRIMIT Formula.

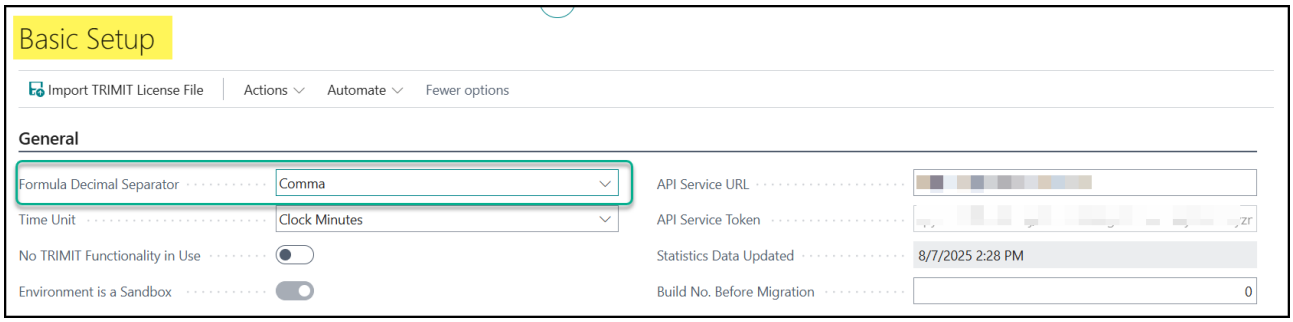
In this document we elaborately discuss this Formula structure with its Fields. Some of these Fields need more clarification and are discussed in their own chapter. This is also the case for some specific Formula features, such as using **Search Tables** and **Cross Calculations**.

The ability to relate Formula Lines to a Formula Header enables the user to reuse a Formula in multiple situations.

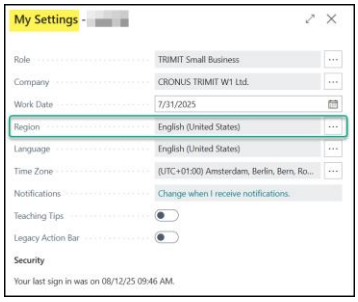
Note
With **TRIMIT Formulas** you can do more than only set Configuration Rules. It can also replace/avoid customization by coding.

Basic Setup

When using Decimal Figures anywhere in the Formulas, be aware that it is with the right Decimal Separator. In the **Basic Setup** you set the **Formula Decimal Separator,** and you can choose between the following two options: *Comma* or *Dot.*



Therefore, make sure this Field is set according to your **Region** setting in the ***My Settings,*** otherwise the system might see a Decimal Separator *Comma* as a Thousand Separator.



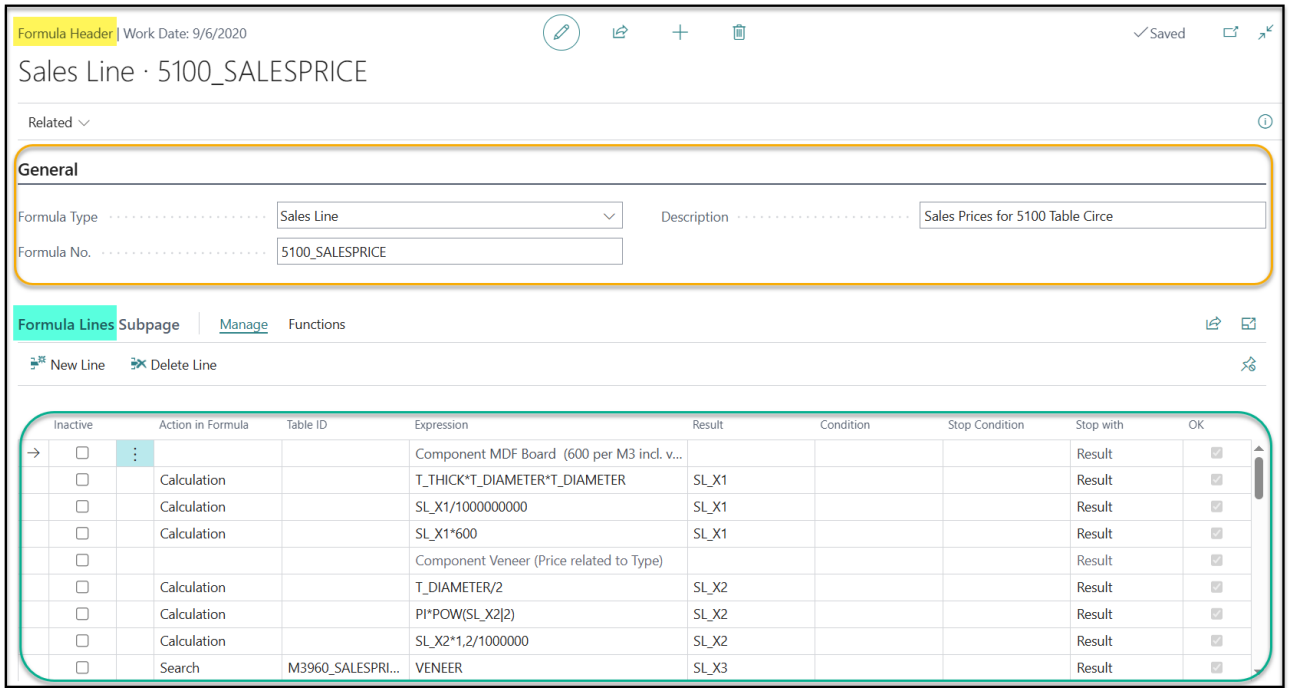
Formula Structure

Introduction

With **TRIMIT Formulas** we can handle extraordinarily complex structures of Configuration Rules. The structure of Formulas consists of three major parts:

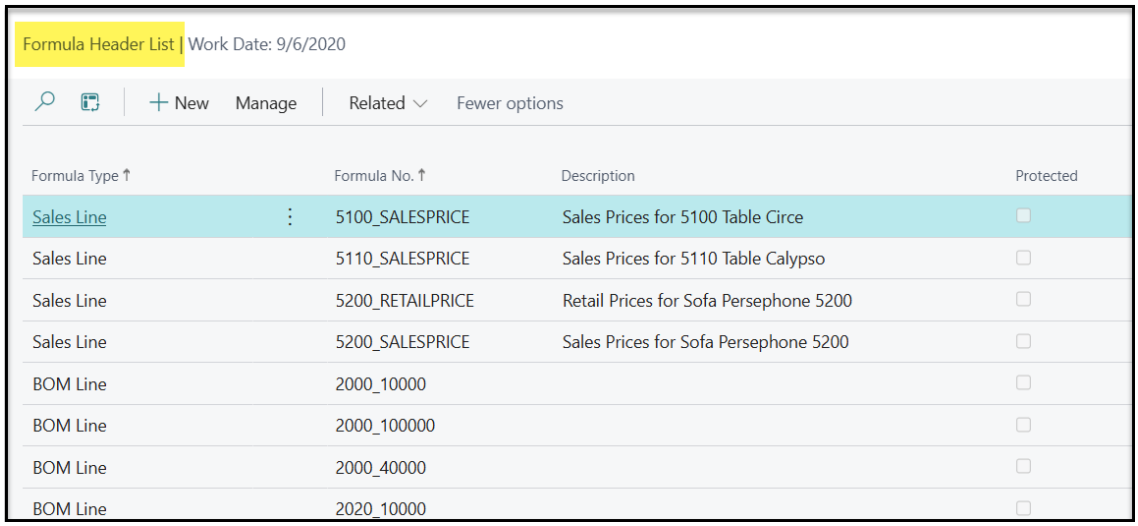
- *Formula Header*
- *Formula Lines*
- *Triggers*

These three parts are discussed in this chapter.

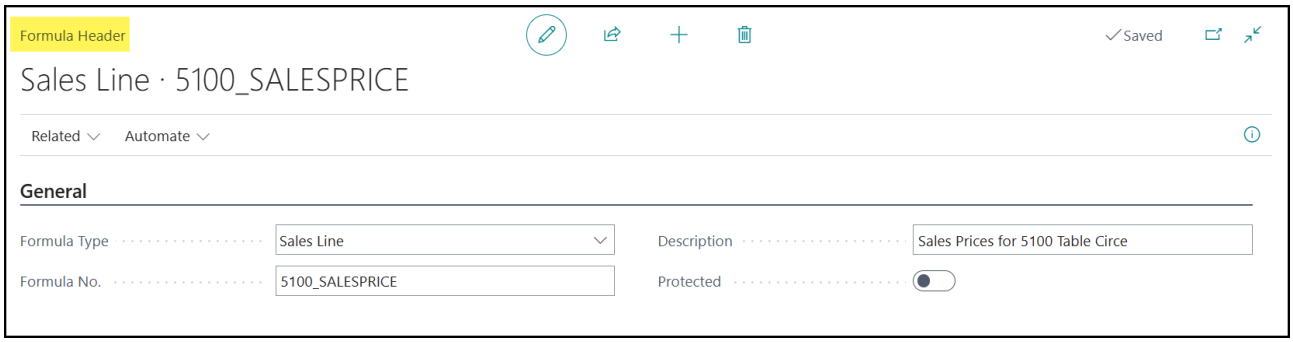


Formula Header

All Formulas contain a Header and Lines (except the Formula Lines used in a Calculation Template). A Formula is an independent entity that can be used in more than one place within the database. These Formulas can be found in the **Formula Header List**.



In this paragraph the Fields and Functions on the Formula Header are discussed.



Formula Type

The **Formula Type** is an indicator on where in the system the Formula is used. The **Formula Type** is normally set automatically in connection with the creation of a Formula via a specific place in the system but can also be set manually in connection with the creation of a Formula Header. The following Formula Types are defined[4]:

Even though the **Formula Type** points at a specific usage, the Formula can in most cases be used **globally**, if execution is possible in the specific situation (all variables are found in the structure where the Formula is called from). In that case the **Formula Type** should be *Global*.
In paragraph *Triggers* the **Formula Types** are described in detail.

Note
When selecting a **Formula** from a List it only shows the Formulas of the appropriate **Formula Type**. You will not see the Formulas of **Formula Type** *Global*. These need to be entered manually by typing. After entering, the system generates a **Confirmation Dialog**. (This can also be done with Formulas of other **Formula Types** but is strongly discouraged.)

[4] Note that in this list you also see the **Formula Type** *Statistics*. This one is obsolete. There are no variables present that point to Statistics Tables. So do not use it!

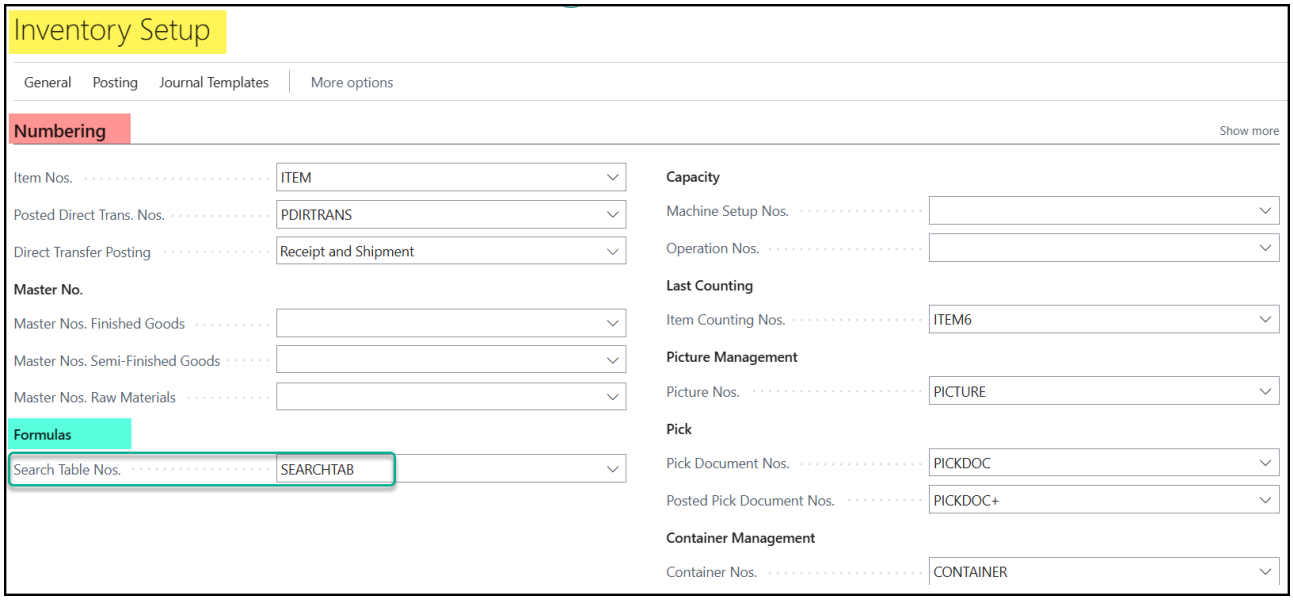
The **Formula Type** is used as a limitation in connection with composition of Formula Lines. It controls which Tables are accessible in connection with lookup on the Field **Table ID**. It also controls which Variables are shown on the page ***Formula Expression AssistEdit*** on lookups for the Fields **Expression**, **Result**, **Condition** and **Stop Condition** on the Formula Lines. This will be elaborated when we discuss the **Formula Lines** in chapter *Formula Lines Structure*.

Formula No.

Because the Formula is an independent entity it has its own unique number: **Formula No.** You are completely free to choose a name for this Formula with the restrictions of a BC Code Field.

In some situations, the number of the Formula is automatically created, when Formula Lines are called. For instance, in connection with a Formula on a **BOM Component Line** for an **Item/Master**. Here the automatically generated Number is a combination of Item/Master No. and the **BOM Component Line No.**[5]. So, Formula *2000_10000* of **Formula Type** *BOM Line,* is a Formula that works on the first BOM Component Line of **Master** *2000*.

Another example is in connection with automatically generated Formulas in connection with **Search Tables** on **Sales-** and **Purchase Prices** where a **No. Series**, set in a Parameter of the **Inventory Setup**, is used.



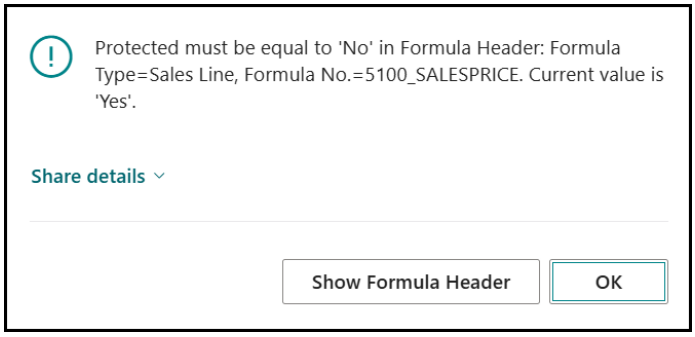
Description

This Field can be used to enter a **Description** of the use of the Formula. This **Text** Field is not mandatory, but it may help clarify the purpose of this Formula and can be used to search for the appropriate Formula in the Formula Header list.

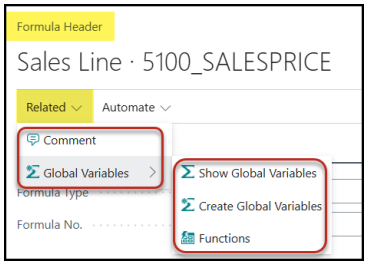
Protected

Field **Protected** controls whether the Formula Lines related to the Formula Header can be changed or not. Editing/creating Formula Lines demands that **Protected** = *FALSE* otherwise you will get an error message:

[5] BOM Component Line numbering is identical to the numbering in standard BC Subpages.



Related to Formula Header



Comment

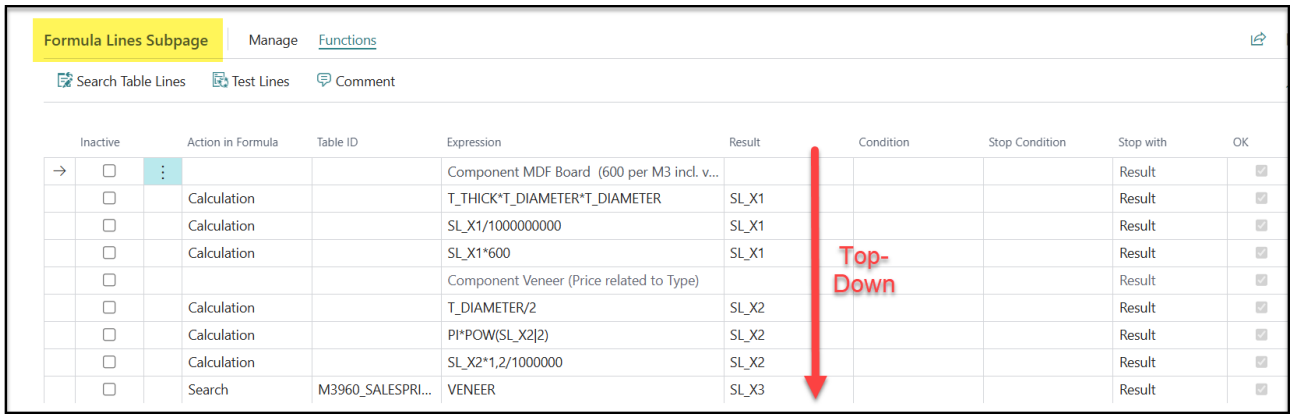
Like in many places in BC/TRIMIT it is possible to add Comments. If a Comment exists, the Boolean **Comment** = *TRUE* (which can be shown in the **Formula Header List** by adding this Field via ***Personalize***). This is set automatically when Comments are entered via ***Related, Comment***.

Global Variables

Via ***Related, Global Variables***, it is possible to see/create the **Global Variables** and **Functions** that can be used in Formulas. More about the Global Variables is described in chapter *Variables.*
For more about Functions: *Appendix B: Functions*.

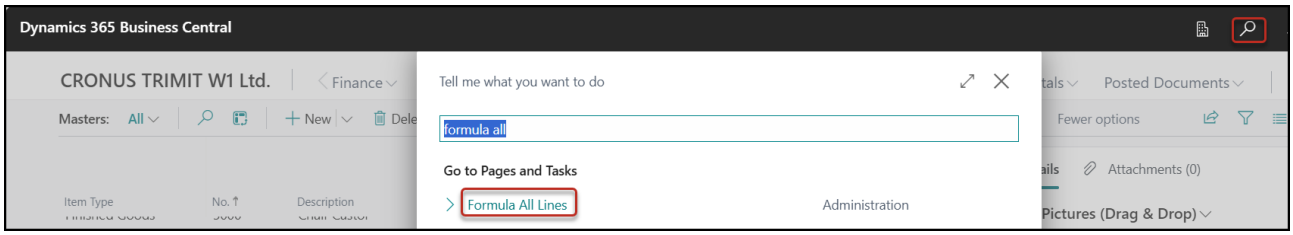
Formula Lines

A **TRIMIT Formula** can consist of any number of **Formula Lines**. Together these describe how Data is extracted from the Database, changed and updated to the Database.
The **Formula Lines** are executed *Top->Down*.

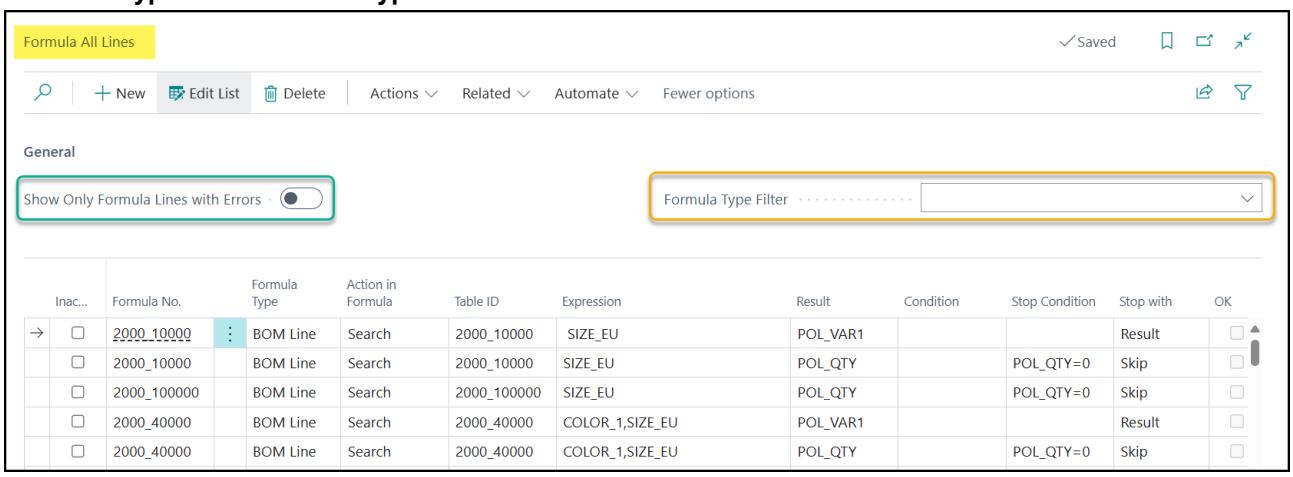


All **Fields** and **Functions** in the ***Formula Lines Subpage*** are described in a separate chapter: *Formula Lines Structure*.

The **Formula Lines** can also be viewed in the ***Formula All Lines*** List (which can be found via **Search**).



Via this List it is possible to filter the Formula Lines that have **Errors** or Formula Lines of a particular **Formula Type**. The **Formula Type** in Formula Lines is inherited from the **Formula Header**.



Inac...	Formula No.	Formula Type	Action in Formula	Table ID	Expression	Result	Condition	Stop Condition	Stop with	OK
→ ☐	2000_10000	BOM Line	Search	2000_10000	SIZE_EU	POL_VAR1			Result	☐
☐	2000_10000	BOM Line	Search	2000_10000	SIZE_EU	POL_QTY		POL_QTY=0	Skip	☐
☐	2000_100000	BOM Line	Search	2000_100000	SIZE_EU	POL_QTY		POL_QTY=0	Skip	☐
☐	2000_40000	BOM Line	Search	2000_40000	COLOR_1,SIZE_EU	POL_VAR1			Result	☐
☐	2000_40000	BOM Line	Search	2000_40000	COLOR_1,SIZE_EU	POL_QTY		POL_QTY=0	Skip	☐

The Formula Lines are connected to a Formula Header via **Formula No.** However, this Field is not by default in the page, but you can add it via ***Personalize.***

All other Fields in the ***Formula All Lines*** List are identical to the ones in the ***Formula Lines Subpage*** and therefore also described in chapter: *Formula Lines Structure*.

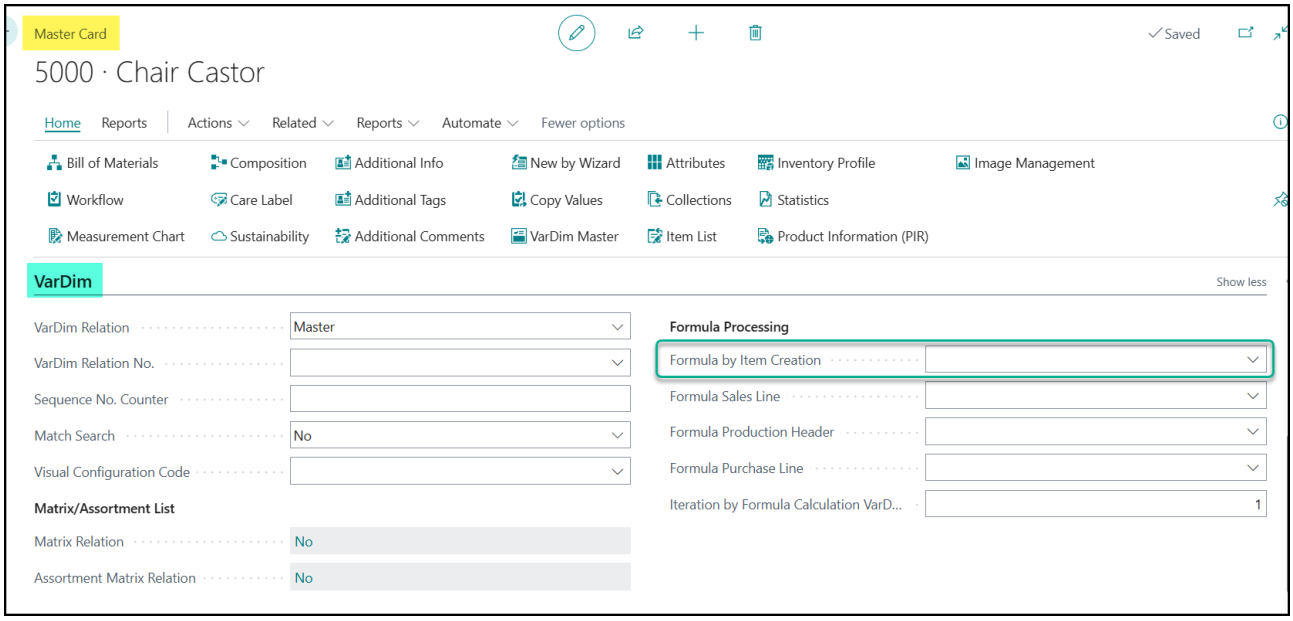
Triggers

Before a **Formula** can be executed, it needs to be triggered. Therefore, TRIMIT has created **Triggers** in the system. Triggers are pre-defined places in the AL Code where Formulas are called.
Described below are the most common places where Formulas can be triggered.

Creating Item from a Master

When an **Item** is configured from a **Master**, a Formula can be executed that populates any of the Fields in the **Item Card** except calculated Fields. Though the Master normally acts as a Template for creating an Item, the Formula overwrites the Values originated from the Master.
This way it is possible to populate Fields on the **Item Card** that deviate from the **Master Card**.

The **Formula by Item Creation** can be set on the ***Master Card*** (FastTab ***VarDim***). The **Formula Type** must be *Item* or *Global,* but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Item*.

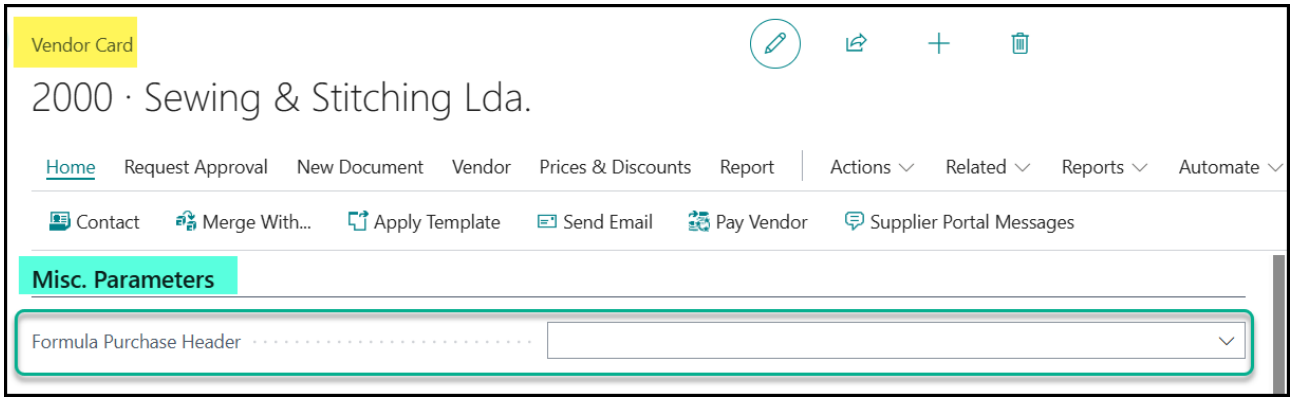


Note
The **Formula by item Creation** is also executed when an **Item** is configured from an **Item Journal** or when it is indirectly created via a **Formula Line** of **Action Type** *Create item*. (The Formula Line Actions are discussed in chapter *Formula Lines Structure*.)

Creating a New Purchase Document

When creating a **Purchase Document (like Quote, Order, Invoice, Credit Memo, Blanket Order, Return Order)**, a Formula can be executed. It is triggered after entering the **Vendor No.** Most Fields on the **Purchase Header** have Values originating from the **Vendor Card**. However, with this Formula it is possible to overwrite these Values or enter Values into Fields that are not inherited from the **Vendor Card**, such as **Order Type**.

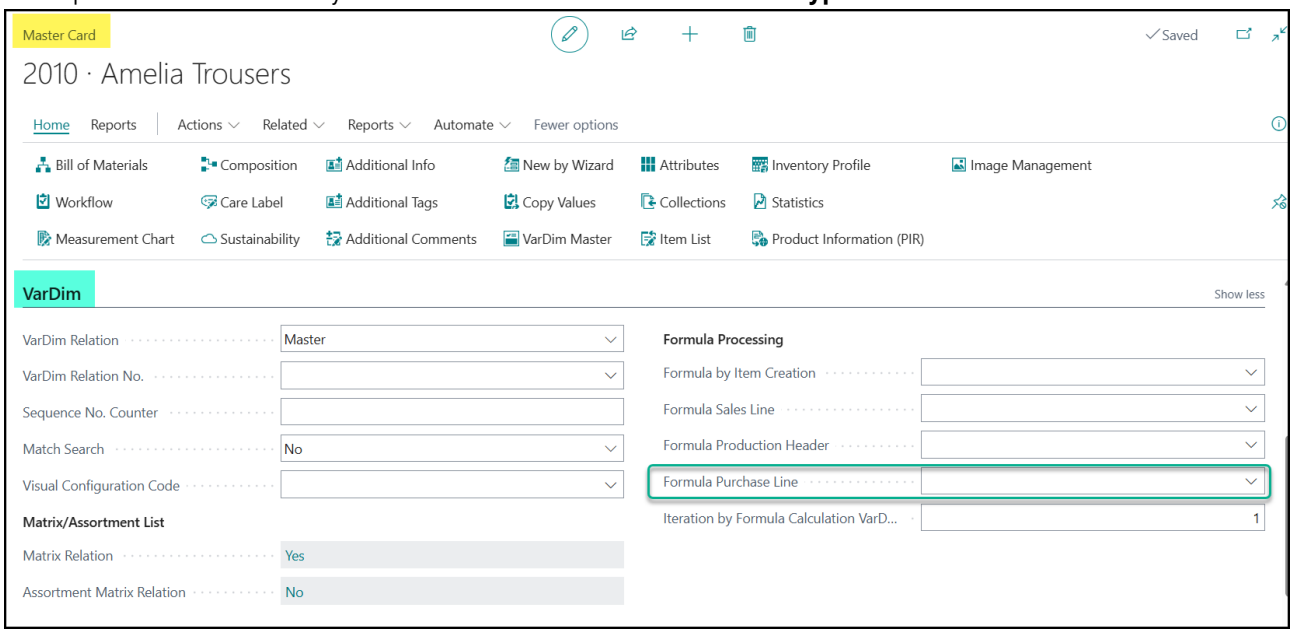
The **Formula Purchase Header** can be set on the ***Vendor Card*** (FastTab ***Misc. Parameters***) and must have **Formula Type** *Purchase Header* or *Global,* but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Purchase Header*.



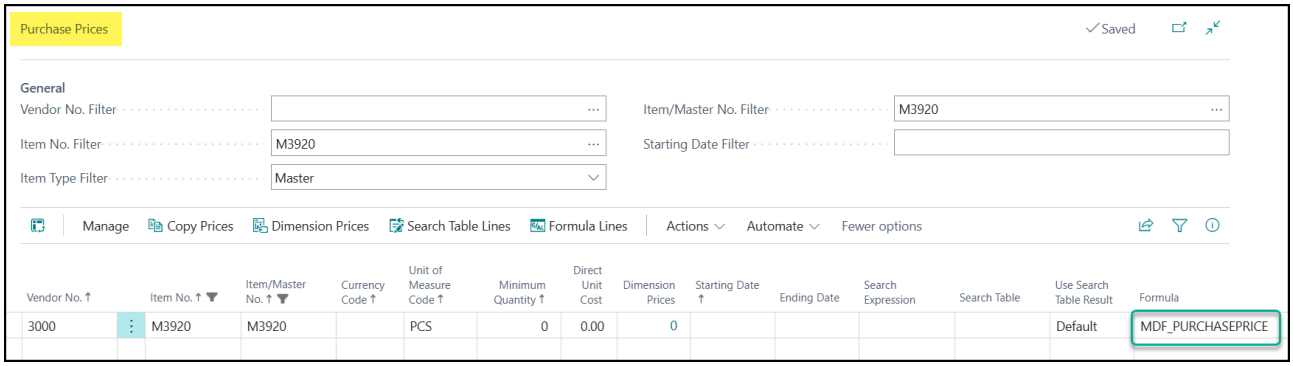
Entering Quantity on Purchase Line

When entering a **Purchase Line**, you enter or configure an **Item** first. On the Purchase Lines, all Data related to the **Item Card** and **Purchase Header** are automatically entered. However, it is possible to manipulate this Data with Formulas. The execution of these Formulas is triggered after entering the **Quantity**.

There are two places where these Formulas can be set. One of them is on the ***Master Card*** in the Field **Formula Purchase Line** (FastTab ***VarDim***). The **Formula Type** must be *Purchase Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Purchase Line*.



It could be that a Formula to be executed depends on the **Purchase Price Table** Line for determining the Purchase Price (**Direct Unit Cost excl. VAT**) on a Purchase Line. Therefore, it is also possible to set a Formula in the **Purchase Price Table** Lines that must have a **Formula Type** *Purchase Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Purchase Line*.
Formulas in the **Purchase Price Table** are set per Purchase Price Table Line and will only be executed when the associated Purchase Price Table Line is applicable.



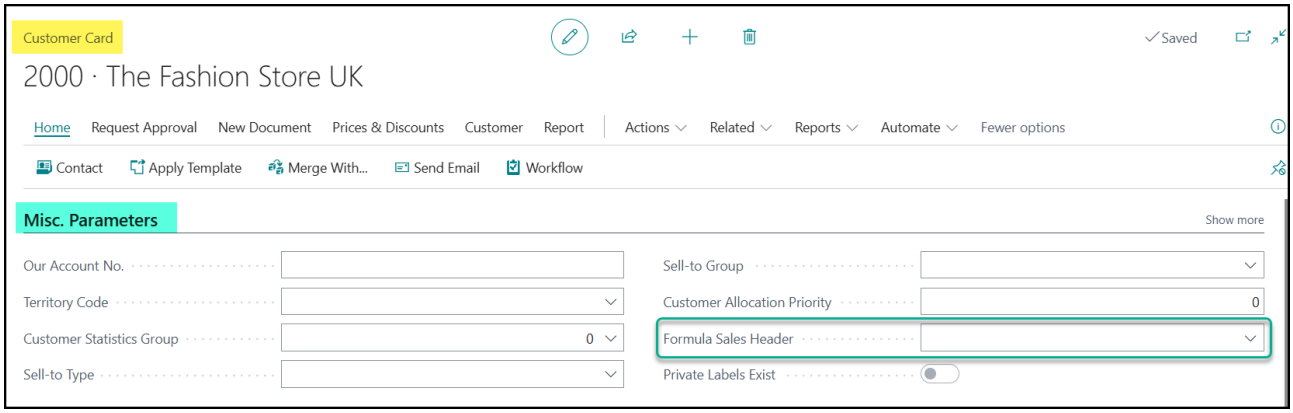
Note

If you have Formulas in both places, the Formula on the **Master/Item** runs first. Then the one on the **Purchase Price Table Line** is executed. So, this may overwrite the Data manipulated by the first Formula.

Creating a New Sales Document

When creating a **Sales Document (like Quote, Order, Invoice, Credit Memo, Blanket Order, Return Order)**, a Formula can be executed. It is triggered after entering **Customer No.** Most Fields in the **Sales Header** have Values originating from the **Customer Card**. However, with this Formula it is possible to overwrite these Values or enter Values into Fields that are not inherited from the **Customer Card**, such as **Order Type**.

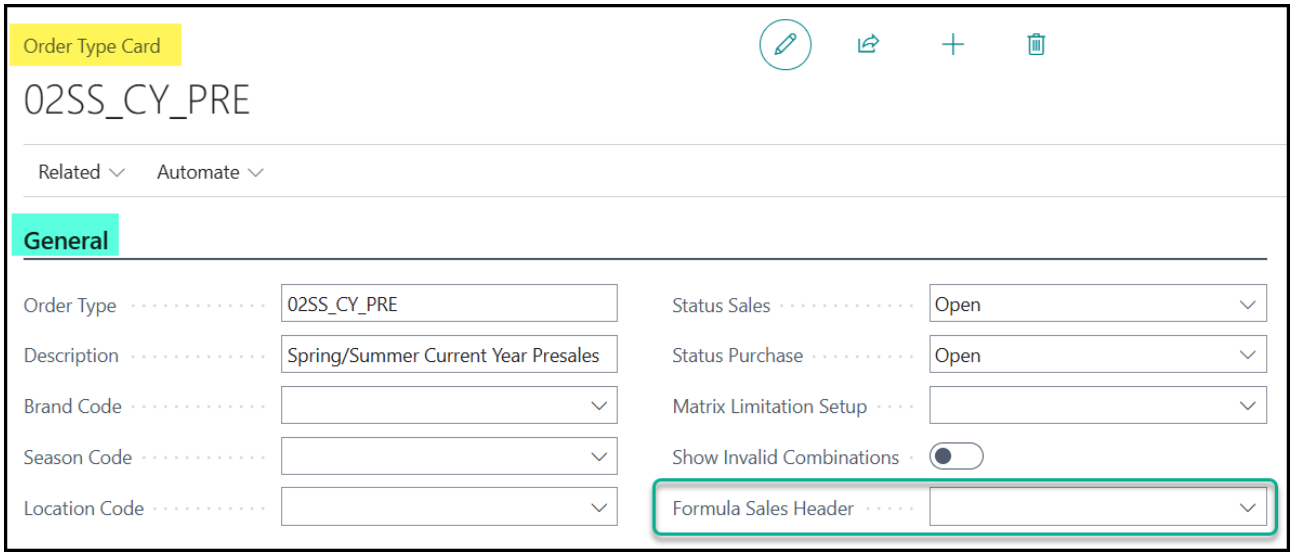
The **Formula Sales Header** can be set on the **Customer Card** (FastTab ***Misc. Parameters***) and must have a **Formula Type** *Sales Header* or *Global,* but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Sales Header*.



Entering Order Type on the Sales Header

There is another moment for which a Formula can be triggered that will overwrite the Values of the Fields on the **Sales Header**. That moment is when the **Order Type** is entered in the Sales Header.

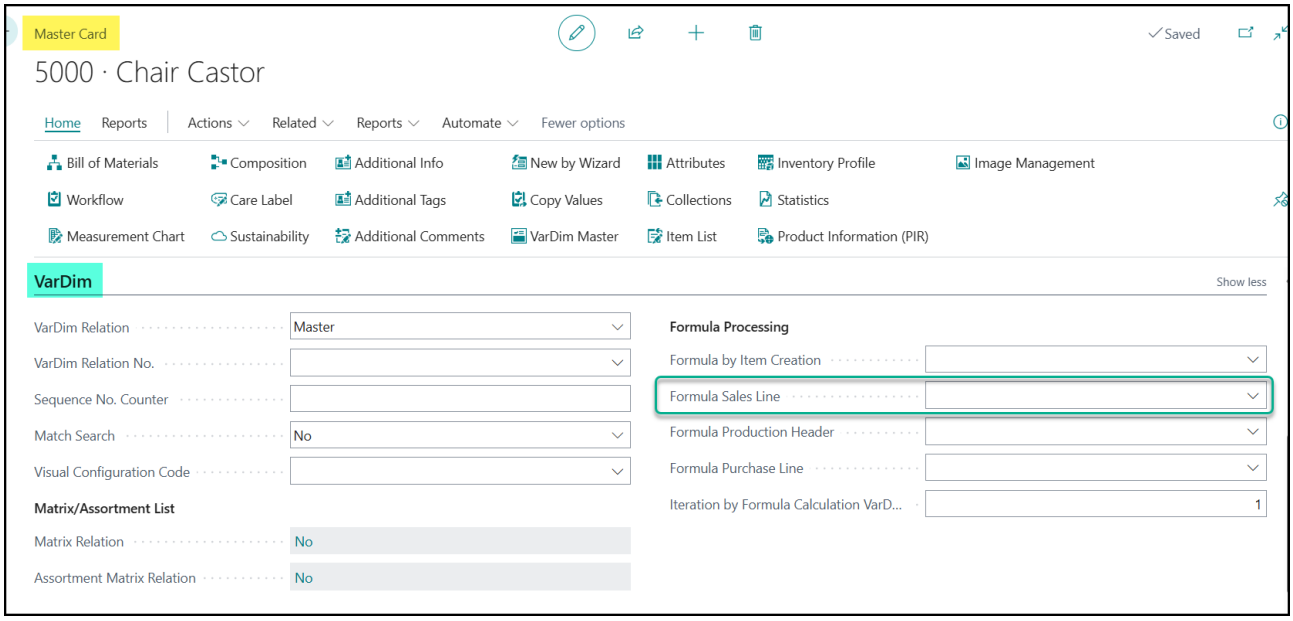
The **Formula Sales Header** can be set on the **Order Type Card**. It is by default not visible; you can add it via ***Personalize***. In the screenshot below, the **Formula Sales Header** is added to the Order Type Card. The **Formula Type** must be *Sales Header* or *Global,* but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Sales Header*.



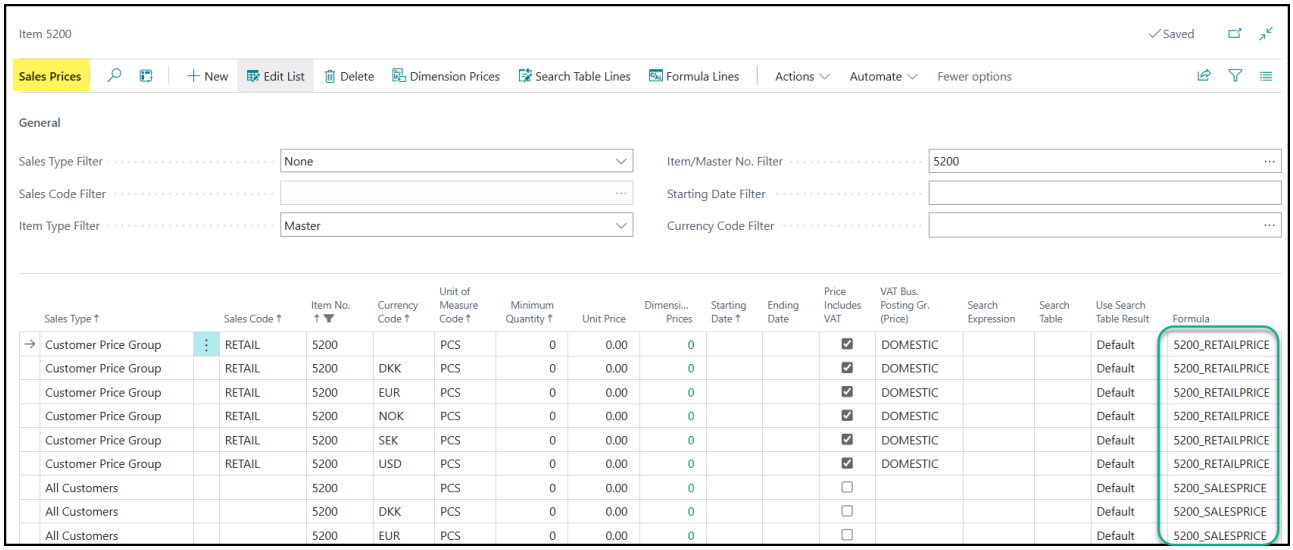
Entering Quantity on Sales Line

When entering a **Sales Line**, you enter or configure an Item first. On the Sales Lines, all Data related to the **Item Card** and **Sales Header** are automatically entered. However, it is possible to manipulate this Data with Formulas. The execution of these Formulas is triggered after entering the **Quantity**.

There are two places where these Formulas can be setup. One of them is on the **Master Card** in the Field **Formula Sales Line** (FastTab ***VarDim***). The **Formula Type** must be *Sales Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Sales Line*.



It could be that a Formula to be executed depends on the **Sales Price Table** Line for determining the Sales Price (**Unit Price excl. VAT**) on the Sales Line). Therefore, it is also possible to set a Formula in the **Sales Price Table** Lines, and it must have **Formula Type** *Sales Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Sales Line*.
Formulas in the **Sales Price Table** are set by Sales Price Table Line and will only be executed when the associated Sales Price Table Line is applicable.



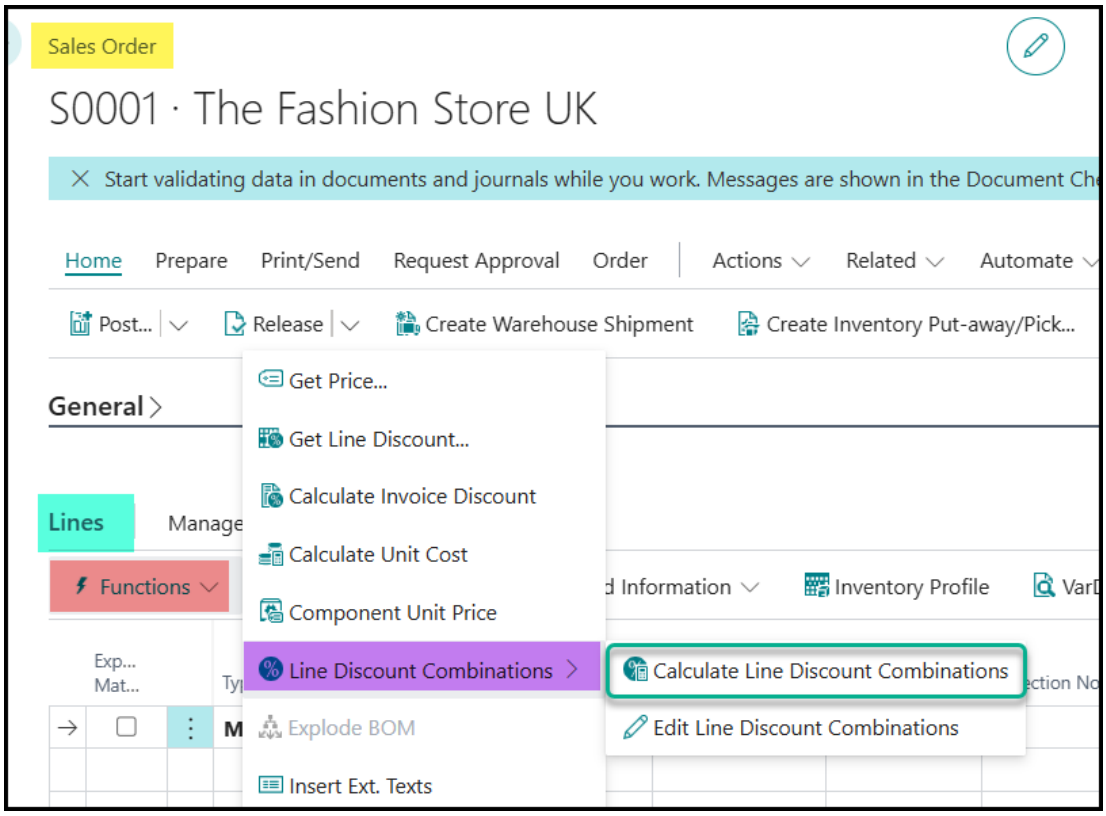
Sales Type	Sales Code	Item No.	Currency Code	Unit of Measure Code	Minimum Quantity	Unit Price	Dimensi... Prices	Starting Date	Ending Date	Price Includes VAT	VAT Bus. Posting Gr. (Price)	Search Expression	Search Table	Use Search Table Result	Formula
Customer Price Group	RETAIL	5200		PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
Customer Price Group	RETAIL	5200	DKK	PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
Customer Price Group	RETAIL	5200	EUR	PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
Customer Price Group	RETAIL	5200	NOK	PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
Customer Price Group	RETAIL	5200	SEK	PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
Customer Price Group	RETAIL	5200	USD	PCS	0	0.00	0			☑	DOMESTIC			Default	5200_RETAILPRICE
All Customers		5200		PCS	0	0.00	0			☐				Default	5200_SALESPRICE
All Customers		5200	DKK	PCS	0	0.00	0			☐				Default	5200_SALESPRICE
All Customers		5200	EUR	PCS	0	0.00	0			☐				Default	5200_SALESPRICE

Note

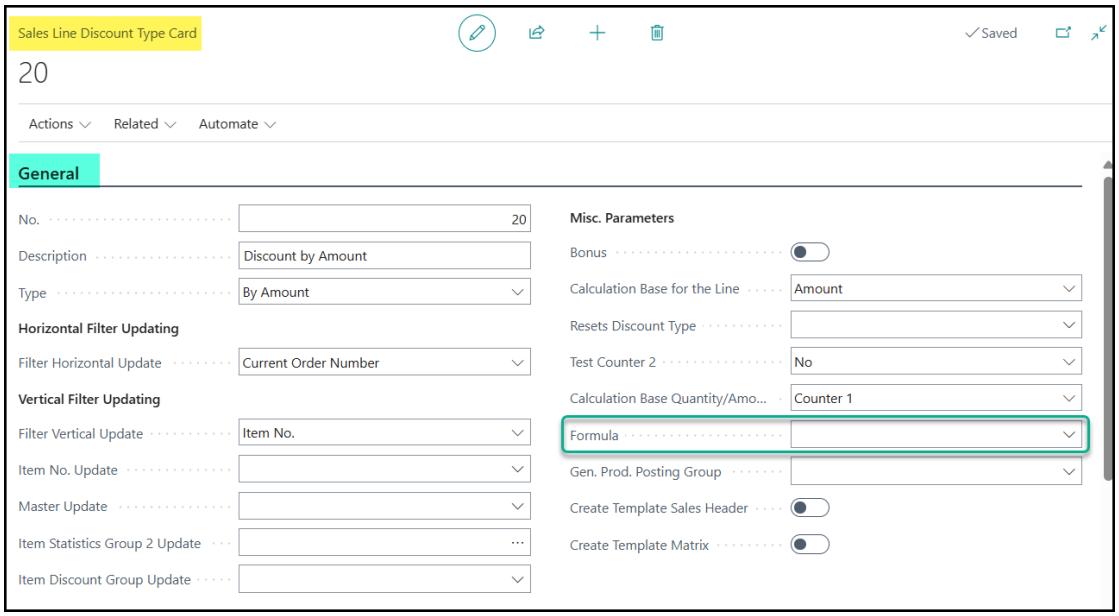
If you have Formulas in both places, the Formula on the **Master/Item** runs first. Then the one on the **Sales Price Table Line** is executed. So, this may overwrite the Data manipulated by the first Formula.

Calculate Line Discount Combinations

In standard BC, a **Line Discount %** is based on the **Quantity** on the **Sales Line**. However, in TRIMIT it is possible to combine the Quantities of multiple Sales Lines to establish whether a **Minimum Quantity** for allocating a **Line Discount %** is reached. Which Sales Line Quantities are totalized depends on the **Discount Combination** settings. The determination of the **Line Discount %** takes place after entering the Quantity on a Sales Line or when calling the function ***Line, Functions, Line Discount Combinations, Calculate Line Discount Combinations*** in case the **Line Discount %** depends on multiple Sales Line Quantities.



This returned **Line Discount %** is not necessarily a fixed Line Discount % for this Minimum Quantity but can be calculated by a **Formula** as well. This Formula can be set on the **Sales Line Discount Type Card** and is triggered when executing the Calculate Line Discount Combination function.
It must have **Formula Type** *Discount* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Discount*.

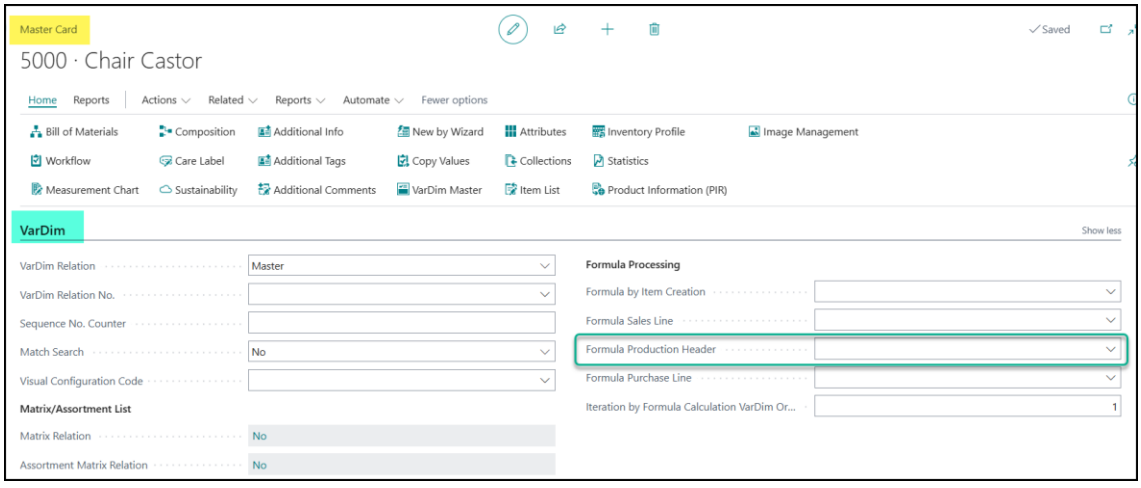


More regarding calculating Sales Line Discount Combinations is discussed in the ***White Paper -TRIMIT Sales Prices and Discounts***.

Creating New Production Order

When creating a new **Production Order**, a Formula can be executed. It is triggered after an **Item No.** is entered on the Production Header. This concerns the **Item to be Produced**. Many Fields in the **Production Order Header** have Values originating from the **Item Card**. However, with this Formula it is possible to overwrite these Values or enter Values into Fields that are not inherited from the Item Card, such as the different **Date Fields** and **Location Code**.

The **Formula Production Header** can be set on the **Master Card** (FastTab ***VarDim***) and has a **Formula Type** *Production Header* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Production Header*.



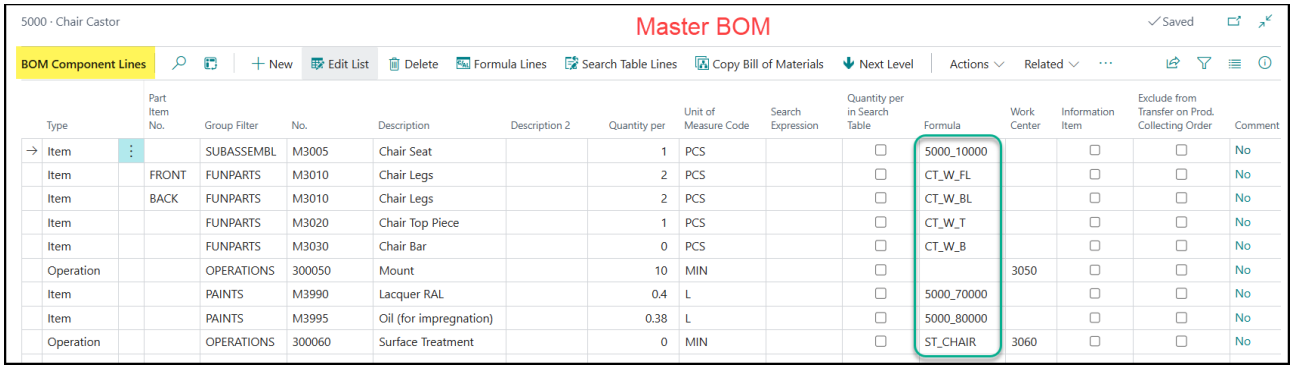
Calculating a Temporary BOM

A **Temporary BOM** can be calculated for an **Item** related to a **Master**. The basis of this Temporary BOM is the **Master BOM**. The difference between a Master and an Item is that the Master has a **VarDim Master** (= set of configuration questions) and the Item has the answers to those configuration questions.[6] To calculate the **BOM Component Lines**, the Master BOM needs to have Configuration Rules based on the configuration answers.

Note

Not only does the **Product to be produced** have variations and therefore a **VarDim Master**, but its **Components** can have these as well. In that case they also need to be set up as **Masters**. So, the Configuration Rules must take care that the right Variant and Dimension of the Component needs to be determined, possibly besides the **Quantity per**, the *measurements*, the *position* of these Components and even if a specific Component is present in the BOM at all.

In BC/TRIMIT there are several ways to set up Configuration Rules. (These are described in chapter *Introduction*) The most flexible tool that allows a high level of complexity is the TRIMIT Formulas feature. For every BOM Component Line you can have different Configuration Rules. Therefore, a **Formula** can be set up on every Component Line in Master BOM. These Formulas must have **Formula Type** *BOM Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *BOM Line*.



5000 · Chair Castor — Master BOM — Saved

BOM Component Lines | New | Edit List | Delete | Formula Lines | Search Table Lines | Copy Bill of Materials | Next Level | Actions | Related

Type	Part Item No.	Group Filter	No.	Description	Description 2	Quantity per	Unit of Measure Code	Search Expression	Quantity per in Search Table	Formula	Work Center	Information Item	Exclude from Transfer on Prod. Collecting Order	Comment
Item		SUBASSEMBL	M3005	Chair Seat		1	PCS			5000_10000				No
Item	FRONT	FUNPARTS	M3010	Chair Legs		2	PCS			CT_W_FL				No
Item	BACK	FUNPARTS	M3010	Chair Legs		2	PCS			CT_W_BL				No
Item		FUNPARTS	M3020	Chair Top Piece		1	PCS			CT_W_T				No
Item		FUNPARTS	M3030	Chair Bar		0	PCS			CT_W_B				No
Operation		OPERATIONS	300050	Mount		10	MIN				3050			No
Item		PAINTS	M3990	Lacquer RAL		0.4	L			5000_70000				No
Item		PAINTS	M3995	Oil (for impregnation)		0.38	L			5000_80000				No
Operation		OPERATIONS	300060	Surface Treatment		0	MIN			ST_CHAIR	3060			No

Note

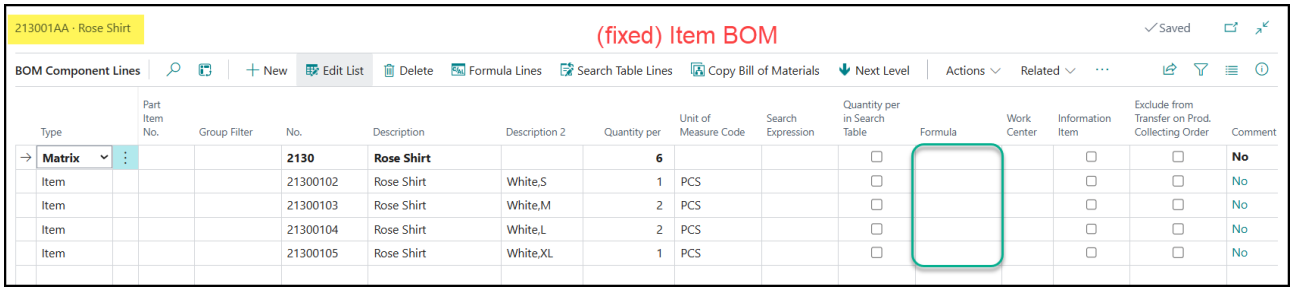
Formulas can be setup for all kind of Component **Types**, so not only for **Items** but for **Operations** and **Setup** as well.

Enter Quantity to Produce on Production Order Header

After entering the **Quantity to Produce** on the **Production Order Header** the system starts populating the **Production Order Lines**. This goes in two phases:

1. The system looks at the **Item No.** to be Produced. If this Item has a **(fixed) Item BOM**, the BOM Component Lines of this Item will be entered on the Production Order Lines taking the **Quantity to Produce** into account. Formulas can manipulate these Production Order Lines. These Formulas are set in the BOM Component Lines of the (fixed) Item BOM.
   The Formulas here must have **Formula Type** *BOM Line* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *BOM Line*.

[6] Note that the configuration answers can be Item specific (**VarDim Item**) or Order specific (**VarDim Order**).



2. If the **Item No.** to be Produced does not have a **(fixed) Item BOM**, the system looks at the **Master BOM** of the related **Master**. The system determines a **Temporary BOM** (see previous paragraph section) and populates the **Production Order Lines** taking the **Quantity to Produce** into account.

Note
Initially the **Production Order Lines** are populated with Data inherited from the **(Master) Item Card**. These can be manipulated by Formulas on the **BOM Component Lines**. However, with these Formulas you can also manipulate the other Production Order Line Fields, for instance the **Quantity per**.

During Configuration

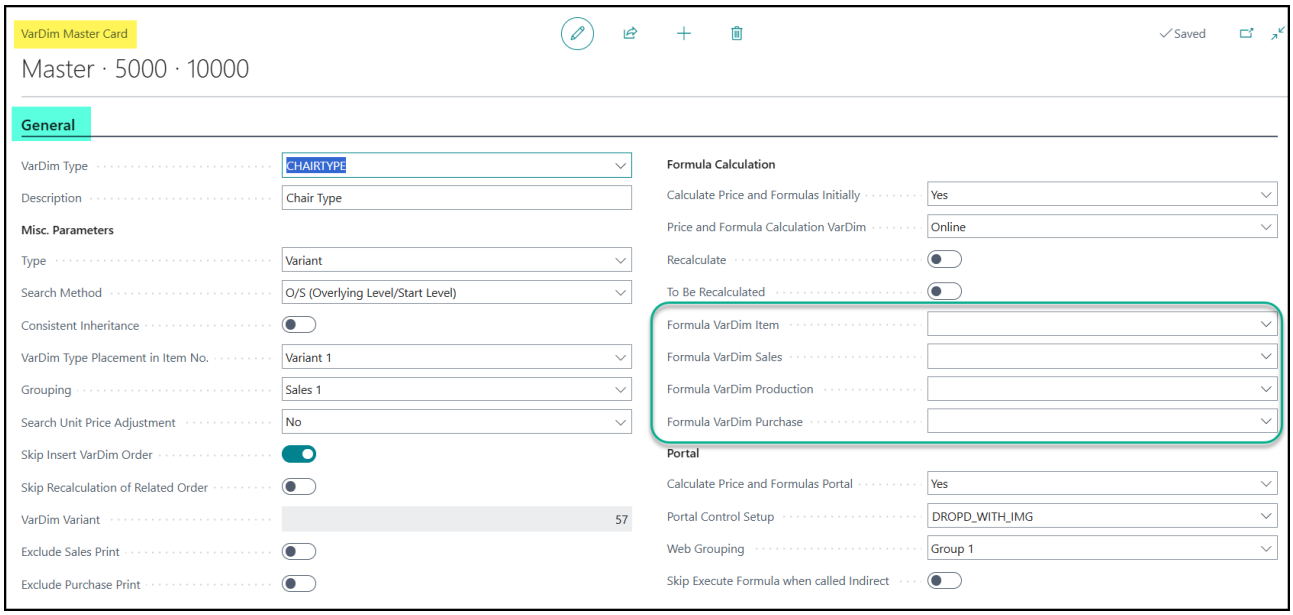
In standard TRIMIT we can configure Items in different places. During this configuration TRIMIT Formulas can be executed depending on where the configuration takes place. These places are:

- Configuring Item directly from **Master**
- Configuring Item on **Item Journal**
- Configuring Item on **Sales Order Line**
- Configuring Item on **Purchase Order Line**
- Configuring Item on **Production Order Header**

Depending on the place where an Item is configured, these Formulas are executed the moment a Value in the **VarDim Item** or **VarDim Order** is validated for a specific **VarDim Type**. In the Table below it is stated from which Tables the Fields can be manipulated by executing these **VarDim Formulas**.

Configuration Place	Formula	Formula Type	Tables
Item directly from Master	VarDim Item	VarDim Item, Global	VarDim item Item (current level)
Item Journal	VarDim Item	VarDim Item, Global	VarDim Item Item (current Level)
Sales Order Line	VarDim Sales	VarDim Sales, Global	VarDim Order Item (current Level) Sales Order Line
Purchase Order Line	VarDim Purchase	VarDim Purchase, Global	VarDim Order Item (current level) Purchase Order Line
Production Order Header	VarDim Production	VarDim Production, Global	VarDim Order Item (to be Produced) Production Order Header
Project Planning Line	VarDim Sales	VarDim Sales Global	VarDim Order Item (current Level) Sales Order Line

The Formulas can be set up on the **VarDim Master Card** (in the Master Card, click ***Home, VarDim Master*** and in the VarDim Master List, click ***Card***). Do not confuse the VarDim Master Card with the **VarDim Type Card**. The VarDim Master Card has some VarDim Type Data but specified for a particular Master. For more about these pages: ***White Paper - TRIMIT Item Structure and Configuration***.



Note
The **VarDim Formulas** can also be made visible on the **VarDim Master List**. This way you do not have to go into the **VarDim Master Card**.

It is important to realize that the VarDim Formulas are only executed when a Value is **validated**. With a VarDim Formula on a particular VarDim Type, it is possible to manipulate the Values in other VarDim Types in the same VarDim Item/Order.
However, these VarDim Types could also have VarDim Formulas. These VarDim Formulas will not be executed because there are no actual manual entries. If you want to cascade down the execution of VarDim Formulas, you need to work with the Booleans on the VarDim Master Card: **Recalculate** and **To Be Recalculated**. More information about this can be found in the ***White Paper - TRIMIT Item Structure and Configuration***.

Formulas triggered in TRIMIT Calculation

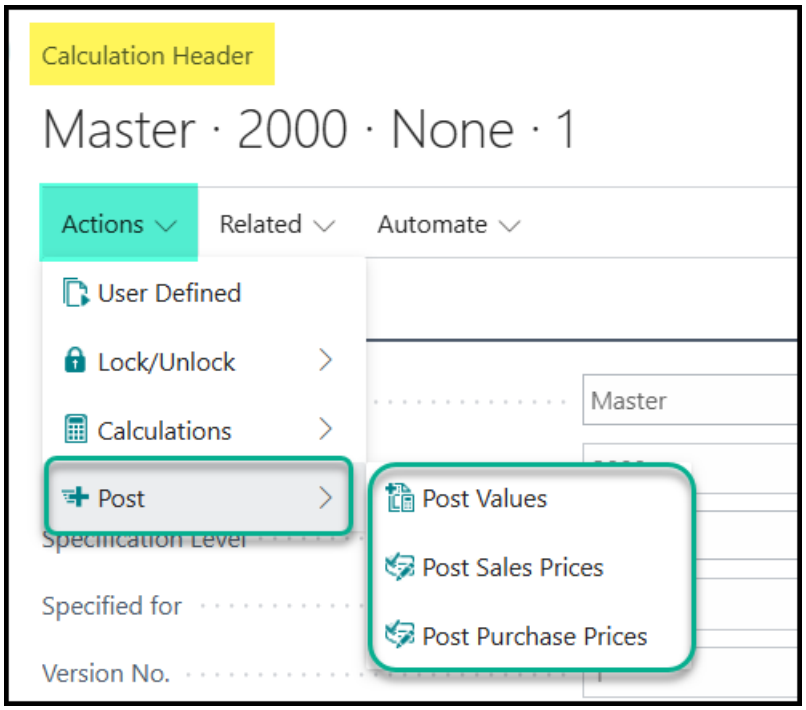
Formulas can be used when calculating Values in a TRIMIT **Calculation** that can be connected to a Master, and can be used to determine Unit Cost, Sales Prices, and/or Purchase Prices. These Formulas will not have a Formula Header but only have Formula Lines set up of **Formula Type** *Calculation* in the **Calculation Template** that is used for calculating the Values in the Calculation. The Calculation and the Calculation Template consist of Columns and Row forming **Calculation Cells** comparable to an Excel Worksheet.

In the **Calculation Template** three types of Formulas can be used:

- **Get Value Formulas**
  These Formulas are used to set the default Values in a new Calculation or to reset Values in an existing Calculation.
  A **Get Value Formula** is triggered when a Calculation is created or recreated. It can determine the default Values in the Calculation Cells.
- **Calculation Formulas**
  These Formulas are used to calculate Values for Calculation Cells based on the Value in the

Calculation Cell to which the Calculation Formula is linked.
A **Calculation Formula** is triggered when the Value in the Calculation Cell is manually entered or changed.

- **Post Formulas**
  These Formulas are used to post (copy) the calculated Values of particular Calculation Cells from the Calculation to specified places in the Database, for instance the Unit Cost of the Item Table or to create specific Sales-/Purchase Prices.
  The Post Formulas are triggered manually via ***Actions, Post*** in the Calculation Header.

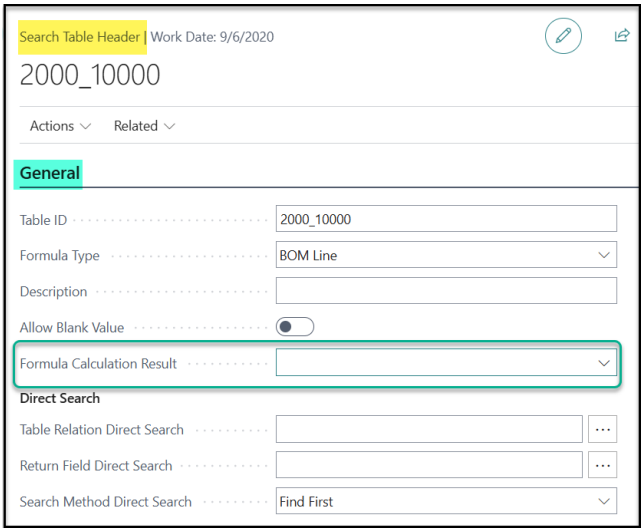


How to set up these Formulas in a Calculation Template is described in ***White Paper - TRIMIT Calculations*** and will not be discussed further in this document.

Formula in Search Table

In **TRIMIT Formulas** it is possible to work with **Search Tables**. Search Tables convert a set of Input Data into different Output possibilities. Normally the Output Data is entered manually in the Search Table. However, in a Search Table also the execution of a **Formula Calculation Result** is possible.
With this Formula you can calculate other Output Data. The Formula is triggered when running the function ***Calculate Search Tables*** in the **Search Table Header**.

The **Formula Calculation Result** can be set up on the **Search Table Header** and has **Formula Type** *Search Table* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Search Table*.



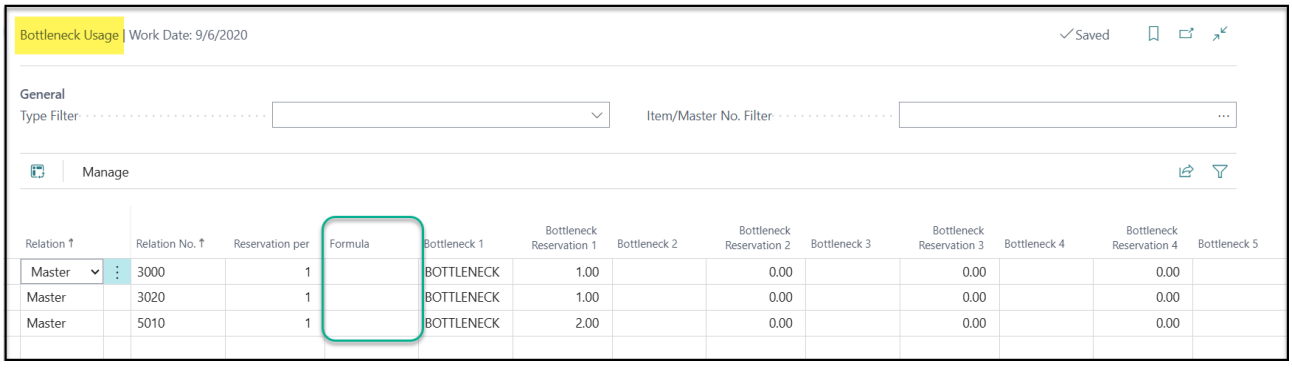
More about Search Tables is discussed in detail in the chapter *Search Tables*.

Formulas for calculating Bottleneck Capacity per Master/Item

During a production process a particular **Work Center** could have critical Capacity. Every one of these Work Centers can be defined as a **Bottleneck**.

If you want to work with a **Fixed Capacity Reservation** per Item, you can set up a **Bottleneck Usage** Table in which you relate a particular **Master** or **Item** to the Capacity reservation for a particular Bottleneck. It is possible to calculate this Capacity reservation with a **Formula**. This Formula is triggered when the system looks for a fixed Capacity reservation.

This Formula can be set up in the **Bottleneck Usage** Table and has **Formula Type** *Bottleneck* or *Global*, but the lookup in this Field will only show the **Formula Nos.** of **Formula Type** *Bottleneck*.



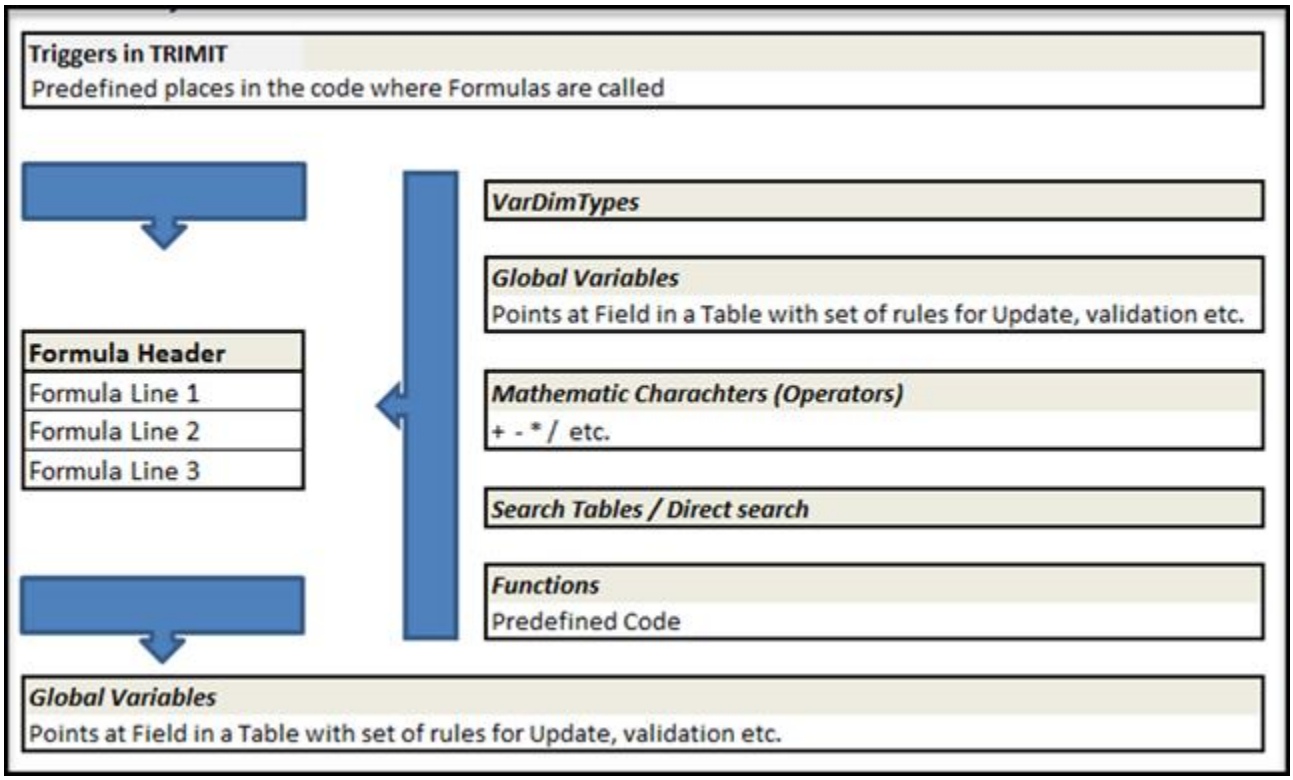
More about Bottlenecks is described in the ***White Paper - TRIMIT Bottleneck***.

Formula Execution Process

Executing a Formula, the BC/TRIMIT system has the following process:

1. The Formula is triggered.
2. It executes Formula Lines *Top -> Down*.
3. For every Formula Line:
    a. See if the condition is met.
    b. Retrieve Input Values (VarDim Values and/or Global Variables) from a Table.
    c. Convert these Values via Calculations, Search Tables, Functions, etc. (Formula Actions)
    d. Put these Return Values as a VarDim Value or Global Variable
4. After executing the whole Formula, put back the final Result Values (Output) back into a Table.

This process is schematically shown in the picture below.

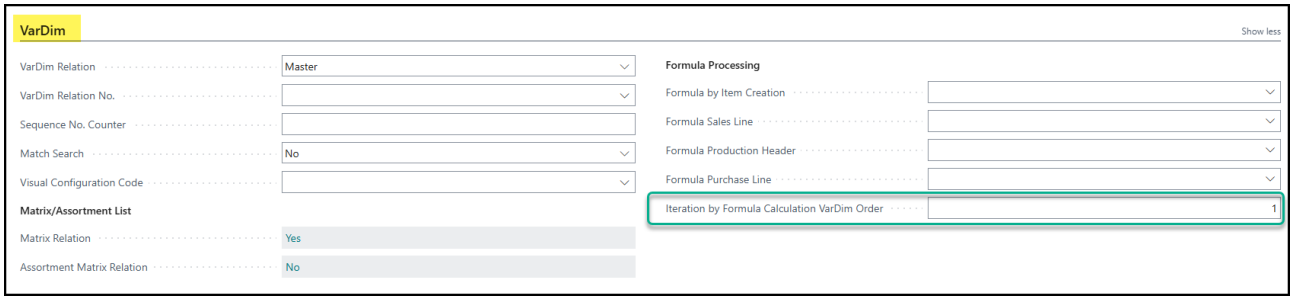


Global Variables, Formula Actions, Search Tables, Function etc. are discussed in the following chapters.

Iterations

When calculating Formulas attached to **VarDim Order** when a **VarDim Order** is left or when the user chooses it, the Formula calculation will be executed until the calculation no longer yields changes. The same applies to the execution of calculations.

If this is not wanted, it is possible to enter how many times the calculation should be executed for the individual Master in the Field **Iteration by Formula Calculation VarDim Order** in table ***6037103 Masters*** (FastTab ***VarDim*** of the **Master Card**). This way a particular calculation can be repeated.



Conclusion

With **TRIMIT Formulas** it is possible to convert Input Values from a specific Table into Output Values of (another) specific Table. The Formulas have their own unique Number and are stored in a specific Table (***6037205 Formula Header***).
Because a Formula is an independent entity, it can be used in more than one place. If possible, it could be useful to make Formulas generic, so that they are easier to maintain.
However, you could also make a Formula specific to the place where it is used.
In general, these are easier to set up, but you will have to create more of them.

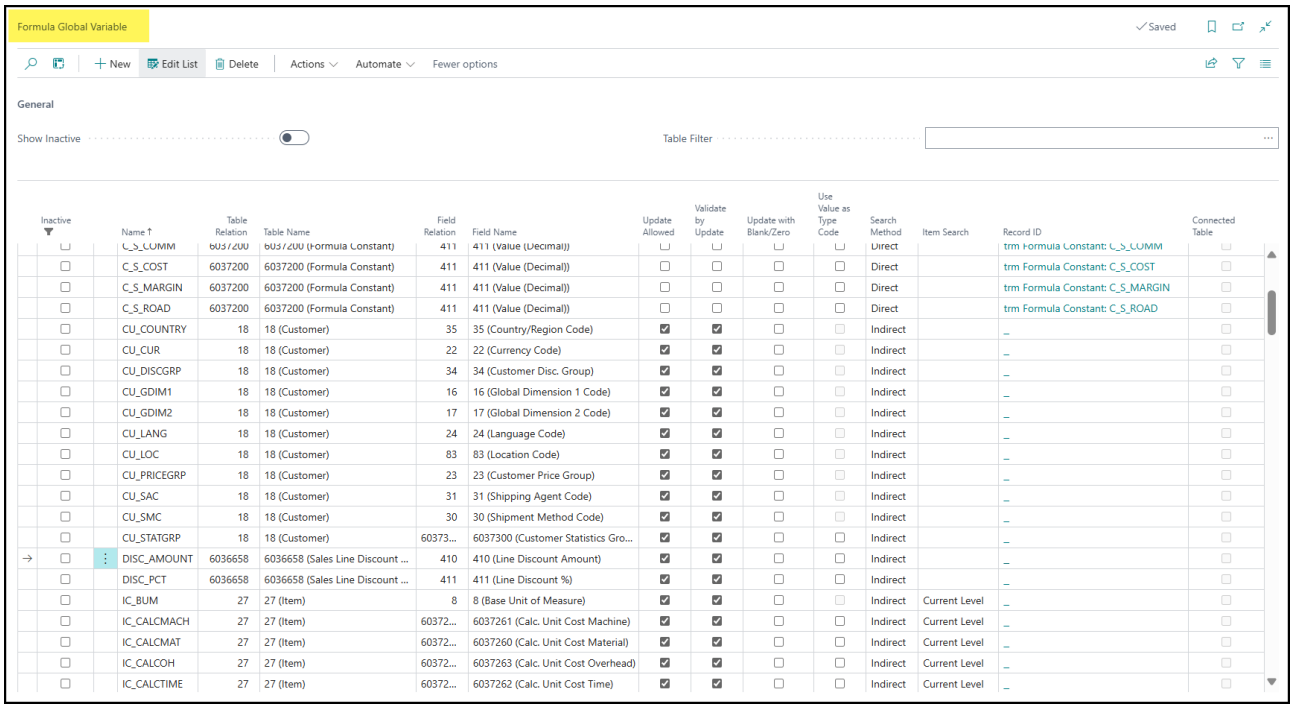
The execution of a Formula takes place according to a specific procedure.
First the Formula needs to be triggered.
Then the Formula Lines are executed Line by Line (*top->down*).
After all the Formula Lines are executed, the Result Values are put back into the appointed Fields.

Variables

Introduction

TRIMIT Formulas act as some kind of mini algorithm. Like any algorithm, variables are needed to calculate with. Since Data need to be extracted from Tables in BC/TRIMT, the Fields in these Tables can be connected to **Global Variables**.

A **Global Variable** points at a specific Field in a specific Table and can be used in the Formula Lines. In connection with the Formula execution, it is possible to read the existing Value of a Field or to modify the Value in the Field depending on if it exists in the structure from where the Formula is executed. The **Global Variables** can be found in the ***Formula Global Variable*** Table (***6037201***).



Inactive	Name	Table Relation	Table Name	Field Relation	Field Name	Update Allowed	Validate by Update	Update with Blank/Zero	Use Value as Type Code	Search Method	Item Search	Record ID	Connected Table
	C_S_COMM	6037200	6037200 (Formula Constant)	411	411 (Value (Decimal))					Direct		trm Formula Constant: C_S_COMM	
	C_S_COST	6037200	6037200 (Formula Constant)	411	411 (Value (Decimal))					Direct		trm Formula Constant: C_S_COST	
	C_S_MARGIN	6037200	6037200 (Formula Constant)	411	411 (Value (Decimal))					Direct		trm Formula Constant: C_S_MARGIN	
	C_S_ROAD	6037200	6037200 (Formula Constant)	411	411 (Value (Decimal))					Direct		trm Formula Constant: C_S_ROAD	
	CU_COUNTRY	18	18 (Customer)	35	35 (Country/Region Code)	✓	✓			Indirect		_	
	CU_CUR	18	18 (Customer)	22	22 (Currency Code)	✓	✓			Indirect		_	
	CU_DISCGRP	18	18 (Customer)	34	34 (Customer Disc. Group)	✓	✓			Indirect		_	
	CU_GDIM1	18	18 (Customer)	16	16 (Global Dimension 1 Code)	✓	✓			Indirect		_	
	CU_GDIM2	18	18 (Customer)	17	17 (Global Dimension 2 Code)	✓	✓			Indirect		_	
	CU_LANG	18	18 (Customer)	24	24 (Language Code)	✓	✓			Indirect		_	
	CU_LOC	18	18 (Customer)	83	83 (Location Code)	✓	✓			Indirect		_	
	CU_PRICEGRP	18	18 (Customer)	23	23 (Customer Price Group)	✓	✓			Indirect		_	
	CU_SAC	18	18 (Customer)	31	31 (Shipping Agent Code)	✓	✓			Indirect		_	
	CU_SMC	18	18 (Customer)	30	30 (Shipment Method Code)	✓	✓			Indirect		_	
	CU_STATGRP	18	18 (Customer)	60373...	6037300 (Customer Statistics Gro...	✓	✓			Indirect		_	
→	DISC_AMOUNT	6036658	6036658 (Sales Line Discount ...	410	410 (Line Discount Amount)	✓	✓			Indirect		_	
	DISC_PCT	6036658	6036658 (Sales Line Discount ...	411	411 (Line Discount %)	✓	✓			Indirect		_	
	IC_BUM	27	27 (Item)	8	8 (Base Unit of Measure)	✓	✓			Indirect	Current Level	_	
	IC_CALCMACH	27	27 (Item)	60372...	6037261 (Calc. Unit Cost Machine)	✓	✓			Indirect	Current Level	_	
	IC_CALCMAT	27	27 (Item)	60372...	6037260 (Calc. Unit Cost Material)	✓	✓			Indirect	Current Level	_	
	IC_CALCOH	27	27 (Item)	60372...	6037263 (Calc. Unit Cost Overhead)	✓	✓			Indirect	Current Level	_	
	IC_CALCTIME	27	27 (Item)	60372...	6037262 (Calc. Unit Cost Time)	✓	✓			Indirect	Current Level	_	

Content of Fields on the Global Variable, controls which Action is allowed in the writing process to the Database. Therefore, a Global Variable is connected to the **TRIMIT Service Parameters,** which controls the writing transactions in the Database, and a **TRIMIT Service Parameter** is automatic created when a Global Variable is created, if it is not existing (see *Appendix D: TRIMIT API Parameters*).
More Global Variables can point at the same Table and Fields.

Note
If Multiple Global Variables exists referring to the same **TRIMIT Service Parameter**, the content of **Update Allowed**, **Validate by Update**, **Update with Blank/Zero** is applied to all Global Variables related to the **TRIMIT Service Parameters**.

The creation of Global Variables is discussed in the paragraph *Creating Global Variables*.

Also, **VarDim Types** can be used as Variables in the Formulas. At the end of this chapter is explained how to work with VarDim Types as Global Variables. For more detailed information about VarDim Types: *Appendix A: VarDim Types* and/or in the ***White Paper - TRIMIT Item Structure and Configuration***.

Fields in Formula Global Variable Table

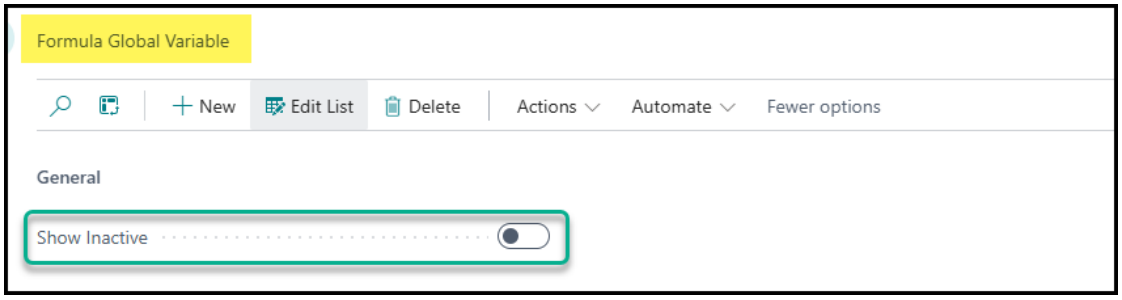
Inactive

It is possible to activate or deactivate a **Global Variable**. If you place a checkmark in the Boolean **Inactive**, the Global Variable becomes de-activated. This means that the system does not recognize a link between the Global Variable and the Field it relates to.
Values of that Field can neither be extracted from the Database nor be placed in.

If the Value of this Field is *TRUE*, the Global Variable will not be visible in the List of Active Global Variables. The default Value of this Boolean is *FALSE*.

You could decide to deactivate all Global Variables that are not in use in Formulas. This limits the List of active Global Variables and on the List page where a Global Variable is selected from the **AssistEdit Formula Expression**. (The latter will be described in the chapter *Formula Expression AssistEdit (Alt+Arrow Down)*.)

In the **Formula Global Variable** Header there is a setting that shows you the Global Variables that are *Inactive*. To make an already created Global Variable active again, switch **Show Inactive** on and remove the checkmark in Field **Inactive** on the appropriate Global Variable.



Name

The **Name** of the Global Variable is a Field of **Type** *Code*. You are completely free to choose any given Name of a Global Variable. However, it is recommended that a certain system in numbering is applied. This will be elaborated in the paragraph *Creating Global Variables*.

Table Relation + Table Name

The **Table Relation** reflects the **Table No.** that the Global Variable should refer to. Based on this **Table No.**, the **Table Name** will be entered automatically.

Although you can have Global Variables for every Field in every Table, it only makes sense to have them for Tables which records can be used in the Formulas. Those tables are:

Table No.	Table Name		Table No.	Table Name
6036672	Bottleneck Setup		6037301	Production Header
6037200	Formula Constant		6037302	Production Line
18	Customer		36	Sales Header
6036658	Sales Line Discount Combination		37	Sales Line
27	Item		6037204	Search Table Line
6037103	Master		6037195	Temp Formula Result
38	Purchase Header		23	Vendor
39	Purchase Line			

Field Relation + Field Name

The **Field Relation** reflects the **Field No.** of the **Table Relation** that the Global Variable should refer to. Based on the chosen **Field No.** the **Field Name** will be entered automatically.

Also, Fields added to this Table from other apps or from customization can be selected.

Update Allowed

If the Boolean **Update Allowed** is set to *TRUE*, the Field is allowed to be updated when executing a Formula, otherwise the update of the Field is skipped.
This event takes place after executing a Formula Line.

The default Value is *FALSE*.

Validate by Update

If the Boolean **Validate by Update** is set to *TRUE,* the **OnValidate** AL Code on the Field is executed when placing the result after executing a Formula and running all the Formula Lines.

The default Value is *FALSE*.

Update with Blank/Zero

If the Boolean **Update with Blank/Zero** is set to *TRUE,* a Blank Value for a Code Field or Zero Value for a Numeric Field is allowed as a result of the Formula Line execution and thus overwriting the current Value. If it is set to *FALSE*, the current Value remains.

The default Value is *FALSE*.

Use Value as Type Code

By definition, **Decimal** Fields contain decimal numbers. **Code** Fields, however, are alphanumeric, which means that they can also be an integer. The contents of these Fields can behave differently.
For instance, a **Code** Field could have Value *05*. A **Numeric** Field will always see this as *5*.
In the Formula Global Variables Table it is possible to define for a Numeric Field that the Values can be treated as a Code Field. If that is the case the Boolean **Use Vale as Type Code** should be set to *TRUE*.

The default Value is *FALSE*.

Note:
Use Value as Type Code Field can only be set *TRUE* for those Global Variables that are connected to decimal Fields and for which the Boolean has not been greyed out.

Search Method + Item Search

In the Field **Search Method,** it can be stated how the Value in the Field is found when the Global Variable is used in a Formula.

The default Value of the Field **Search Method** is *Indirect*.

In a **BOM Structure** there are more Levels of Items. To point to the correct Item Level, it is necessary to have a Global Variable pointed to the appropriate record. Therefore, more Global Variables are defined for **Masters** and **Items** based on the appropriate Item Level: *Current Level*, *Overlying Level* and *Start Level*. The Item Search defines this.

If no level is selected in the Field **Item Search**, the Level is considered as *Current Level.*

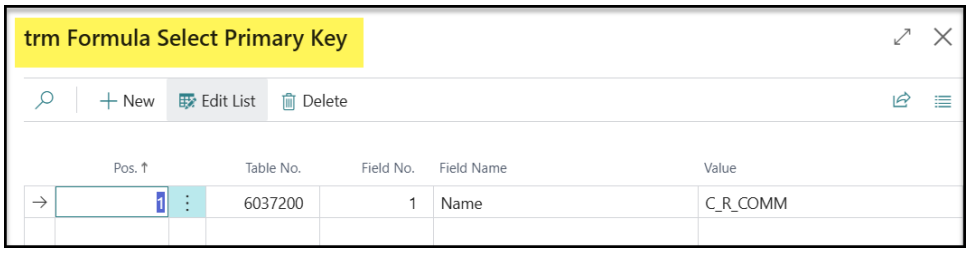
Indirect	That the **Search Method** is *Indirect* indicates that at any given time it is the Value of the current record that is used. Which record is the current record in the specific situation, is decided based on different principles. If the Formula is executed in connection with a Field in a Table and the Formula execution uses Global Variables that points at Fields in this Table, the current record is the record from which the Formula is executed. If the Formula is executed in connection with the creation of a BOM and the Formula execution makes use of Global Variables which points to Fields in the following Tables, the current record will be the records which are present in the BOM Structure: • Table ***27 Item*** • Table ***36 Sales Header*** • Table ***37 Sales Line*** • Table ***38 Purchase Header*** • Table ***39 Purchase Line*** • Table ***6037103 Master*** • Table ***6037105 VarDim Order*** • Table ***6037301 Production Header*** • Table ***6037302 Production Lines*** Regarding a Global Variable that points to a Field on an Item or a Master, the Field **Item Search** gives the possibility to select which Item or Master to use when the Formula is executed in a BOM Structure. The options are *Start Level, Overlying Level* and *Current Level.* If the current record is not found by the methods described above, it is tested if the record from which the Formula is executed has a direct relation to the Table the Global Variable points. This method can only be used regarding tables where the primary key consists of only one Field, for instance: Table ***18 Customer.*** If the record from which the Formula is executed has Fields with direct relation to the Table the Global Variable points too, it will automatically be the first Field that will be used, unless it is stated in table ***6037216 Formula Table Relations,*** which Field should be used. An example of this is the relationship between Table ***37 Sales Line*** and Table ***13 Salesperson/Purchaser***. In this relationship, Fields **6039226**, **6036927** and **6036929** points to Table 13.
Direct	If the **Search Method** is *Direct,* a specific record must be stated in the Field **Record ID**. Lookup on the Field (Alt+Arrow Down) shows the Fields in the primary key that need to be filled. **Direct Search** is commonly used in combination with TRIMIT **Formula Constants**. (Formula Constants is discussed in *Appendix C: Constants*).
Temporary	That the **Search Method** is *Temporary* means that the record only exists in connection with the Formula execution. If a specific record is entered in the Field **Record ID,** the record will be initialized with the current Values of this record. Otherwise, the record will be uninitialized.

	Regarding the Formula execution, the record can be updated and subsequently be read. This enables a Temporary record to be used as a container for Temporary results. Formula execution regarding the creation of a BOM, the concluding Formula execution of a **VarDim Order** and the Formula execution regarding a Calculation, the Temporary records will keep the current Value during the entire Formula execution. Up to 10 Temporary Tables can be used in a Formula.

Record ID

The Field **Record ID** is only used in connection with **Search Method** *Direct.*

With this Field you can set the specific record in the Table set in the Field **Table Relation**. You cannot write directly in the Field but must use lookup (Alt+Arrow Down) in the Field, which is called the Page ***Formula Select Primary Key*** that is automatically filled with the Fields in the primary key of the Table set in the Field **Table Relation.** Enter the **Value** Field for each Key position to identify the record you want to point at in the Direct Search.

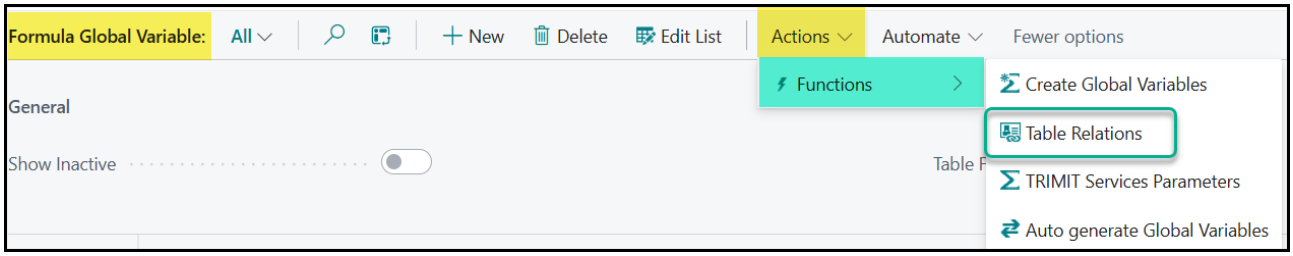


Note:
You can only do **Direct Search** in Tables that have only one **Primary Key**.

Connected Table

The Field **Connected Table** can only be used in connection with **Search Method** *Indirect* and *Temporary*.

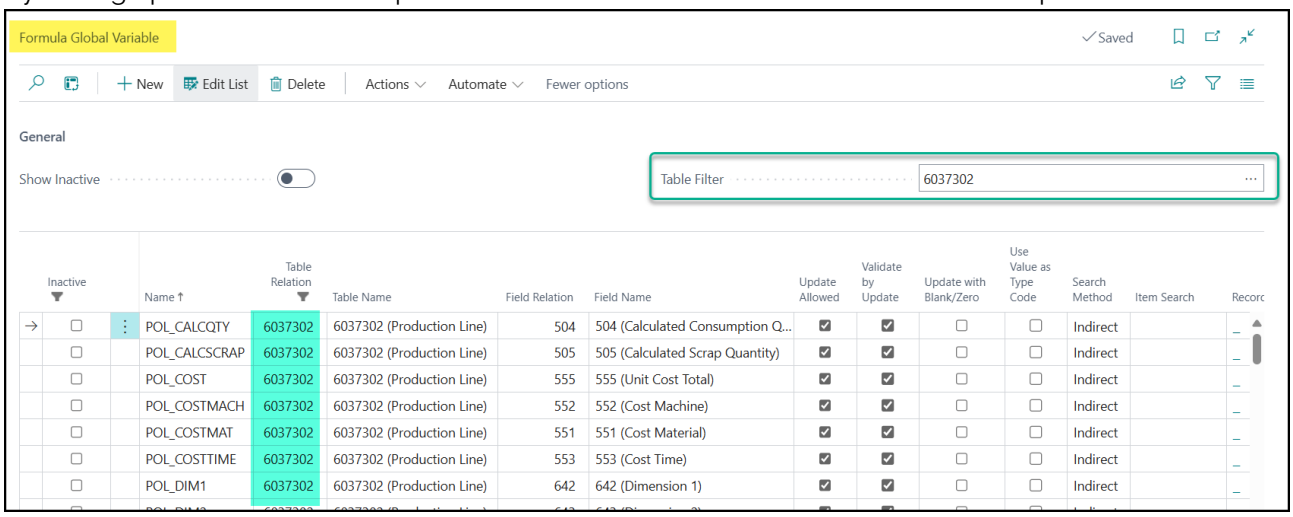
The Field itself is calculated and only shows if a connected Table is present if a checkmark is set in this Boolean. Which Table is connected can be found by using the function ***Table Relations*** via ***Actions, Functions***



This functionality can be used to find related information in other Tables if a Table has multiple possibilities to be attached to another Table. This will be clarified with an example in the next paragraph section *Global Variables with Table Connections*.

Table Filter

By setting up a **Table Filter** it is possible to see all the Global Variables related to a specific Table.



Inactive	Name	Table Relation	Table Name	Field Relation	Field Name	Update Allowed	Validate by Update	Update with Blank/Zero	Use Value as Type Code	Search Method	Item Search
	POL_CALCQTY	6037302	6037302 (Production Line)	504	504 (Calculated Consumption Q...	☑	☑	☐	☐	Indirect	
	POL_CALCSCRAP	6037302	6037302 (Production Line)	505	505 (Calculated Scrap Quantity)	☑	☑	☐	☐	Indirect	
	POL_COST	6037302	6037302 (Production Line)	555	555 (Unit Cost Total)	☑	☑	☐	☐	Indirect	
	POL_COSTMACH	6037302	6037302 (Production Line)	552	552 (Cost Machine)	☑	☑	☐	☐	Indirect	
	POL_COSTMAT	6037302	6037302 (Production Line)	551	551 (Cost Material)	☑	☑	☐	☐	Indirect	
	POL_COSTTIME	6037302	6037302 (Production Line)	553	553 (Cost Time)	☑	☑	☐	☐	Indirect	
	POL_DIM1	6037302	6037302 (Production Line)	642	642 (Dimension 1)	☑	☑	☐	☐	Indirect	

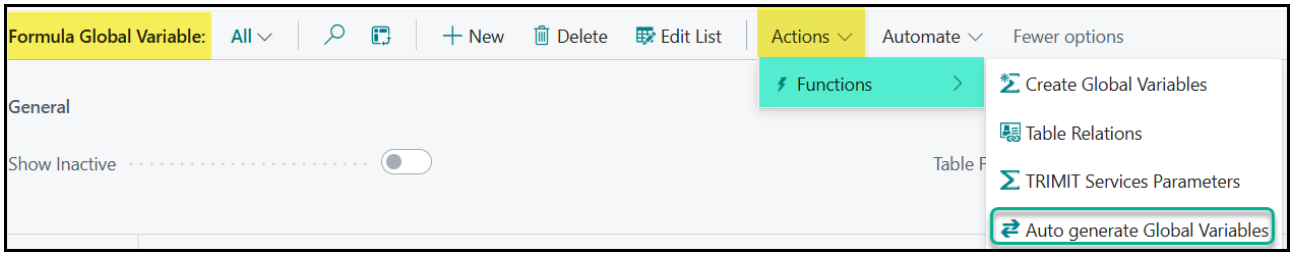
Creating Global Variables

Introduction

The most frequently used Global Variables are auto created. However, it is still possible to add Global Variables of Fields from a particular Table that are not in the Global Variable Table yet.
This could also be the case for Fields added to a Table by customization or via other apps.
In this paragraph we also discuss the creation of a Global Variable with a Connected Table.

Auto-created Global Variables

When installing the **TRIMIT App** or when a new Company is created, the most frequently used Global Variables are auto created. This is included in Codeunit ***6036550 trm General***.
However, the auto-creation of the most frequently used Global Variables can also be started manually via ***Actions, Functions, Auto generate Global Variables***.



Creating Global Variables Manually

Global Variables can also be created manually. You can do this for standard BC/TRIMIT Fields for which no Global Variables are auto-created, but also for Fields integrated in the Tables via other apps or even for customized Fields. Just use the ***+New*** to enter a new Global Variable in the Line.

Although the **Name** of a Global Variable is free to choose when you create a new Global Variable, it is recommended to use the same naming system as the auto-created Global Variables.
If you take a good look at these auto-created Global Variables, you can see a certain structure in the names. They begin with a reference to the Table the Global Variable is related to, then an underscore, followed by a reference to the Field of this Table. The **Reference Code** for the Tables are shown in the Table below:

Table No.	Table Name	Item Search	Reference Code Table
6036672	Bottleneck Setup		BN
6037200	Formula Constant		C
18	Customer		CU
6036658	Sales Line Discount Combination		DISC
27	Item	Current Level	IC
27	Item	Overlying Level	IO
27	Item	Start Level	IS
6037103	Master	Current Level	MC
6037103	Master	Overlying Level	MO
6037103	Master	Start Level	MS
38	Purchase Header		PH
39	Purchase Line		PL
6037301	Production Header		POH
6037302	Production Line		POL
36	Sales Header		SH
37	Sales Line		SL
6037204	Search Table Line		STL
6037195	Temp Formula Result		TR
23	Vendor		VE

Although you can create Global Variables for any Field in the Database, TRIMIT Formulas can only handle Global Variables from Tables mentioned in the Table above. In standard TRIMIT, all Formula execution triggers use one of these **Table Nos.** Without any customization it only makes sense to create Global Variables of these Database Tables.

Global Variables with Table Connections

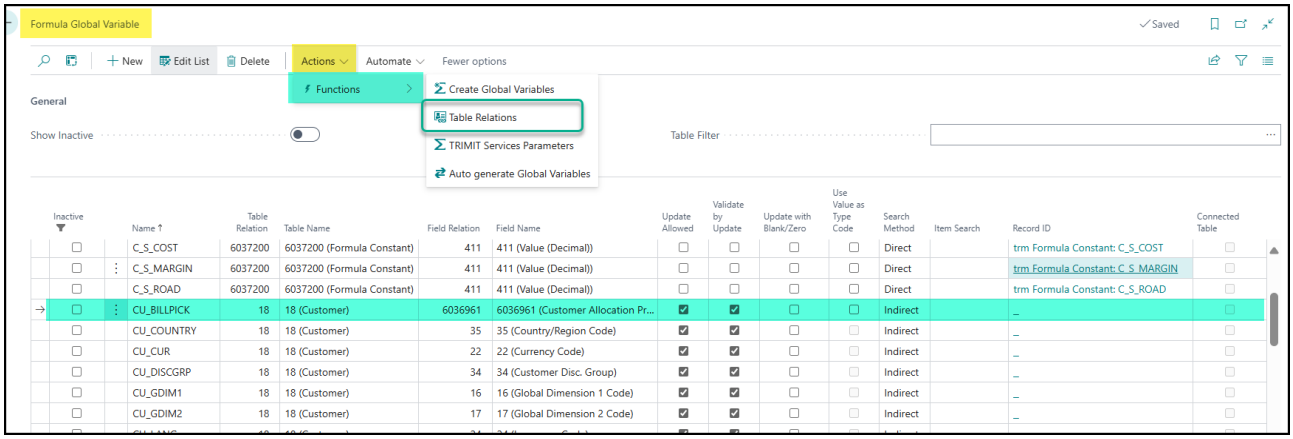
When discussing the Field **Connected Table** in the **Formula Global Variables** Table it is mentioned that a particular Table has multiple links to the same Table. That means that in some cases we must tell which record to take. This is elaborated in the example below:

A **Sales Header** (table 36) has multiple links to the **Customer** (table 18). We have the **Sell-to Customer No** and **Bill-to Customer No.** By default, a **Global Variable** in the structure that connects the Sales Header to the Customer is linked to the Field **Sell-to-Customer**. This means that all Data from this Customer record is available for the TRIMIT Formulas. If we need a Field from this record required for a TRIMIT Formula that manipulates the Sales Header, we can use the straightforward setting of Global Variables.

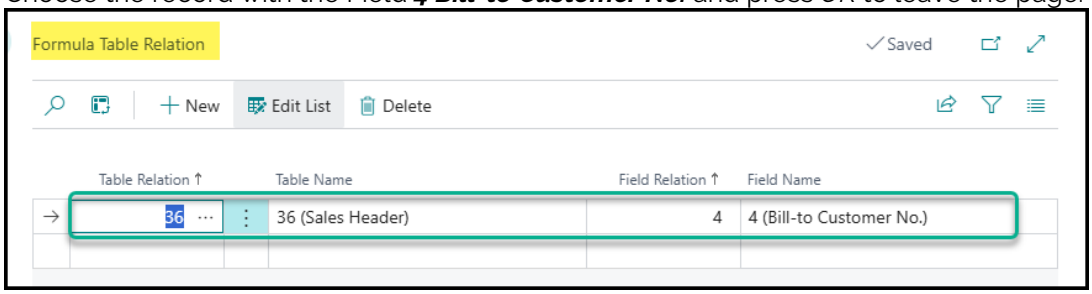
However, let us assume that on a Sales Header, we do not want for instance the **Customer Allocation Priority** of the **Sell-to Customer** but instead of the Bill-to Customer.
That means that we need to have a Global Variable pointing at another record on the same Customer Table. Therefore, we need to create a Global Variable not pointing at the **Sell-to-Customer**, but at the **Bill-to Customer**.

For this case we must set up a Table relation changing the relation from the default **Sell-to Customer No.** to the **Bill-to Customer** by taking the following steps:

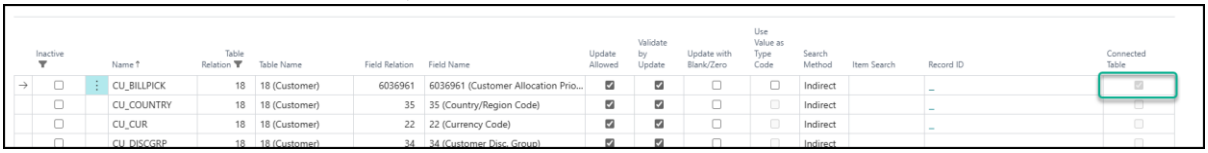
1. Create a new Global Variable (for instance *CU_BILLPICK*) with relation to Table ***18 Customer*** and Field ***6036961 Customer Allocation Priority***.
2. Put **Search Method** on *Indirect*.
3. Click ***Actions, Functions,*** **Table Relations**



4. In the page ***Formula Table Relation*** enter the **Table Relation** with table ***36 Sales Header***
5. Go to the Field **Field Relation.** A lookup in the Field (**Alt+Arrow Down**) shows the *4* options, on the page ***Fields,*** for a relation between the two Tables.
   Choose the record with the Field ***4 Bill-to Customer No.*** and press *OK* to leave the page.

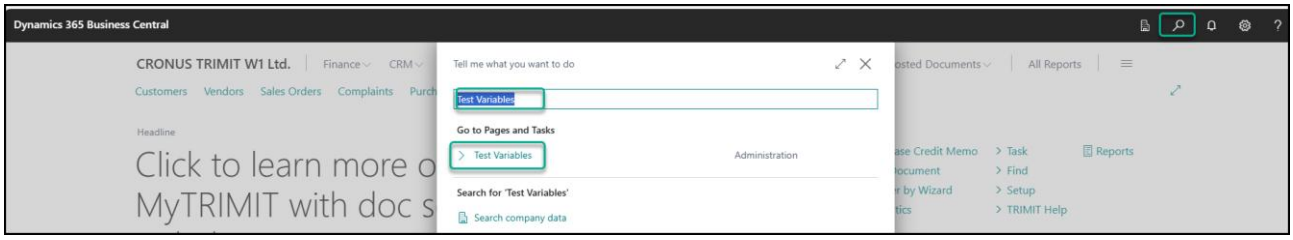


6. Leave the ***Formula Table Relation*** Page and note that in the Formula Global Variable Table for the Global Variable *CU_BILLPICK*, there is a checkmark in the Field **Connected Table**.



Test of Variables

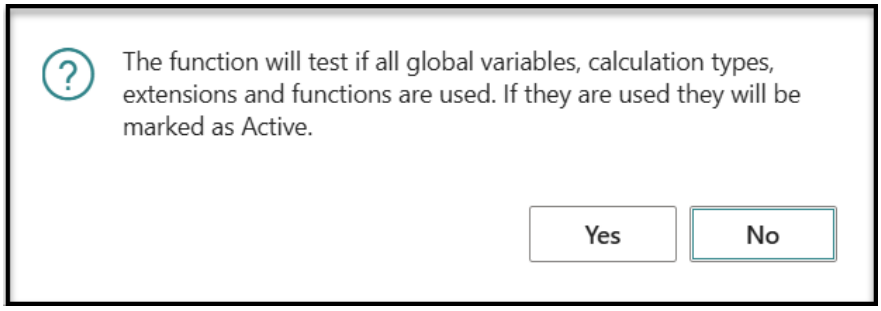
In certain cases, in an already live Database, it can be a good idea to get an overview of which Variables and Functions etc. is used in connection with the Formula execution.
TRIMIT is delivered with a Function **Test Variables** that tests all *Global Variables, VarDim Extensions, Calculation Types, Calculation VarDim Extensions* and *Functions* if they are actively used in the existing Formulas.
You can find it via **Search** *Test Variables*.



If the Variables are not used in existing Formulas, they are marked with *TRUE* in the Field **Inactive** in the corresponding Tables. The Variables that are *Inactive* will not be shown in the Variable Overview when editing the Formulas via page ***6037204 Formula Expression AssistEdit*** and will be removed from the Active Overviews in the corresponding pages where they are created. You also have the possibility to manually remove the *Inactive* Variables permanently by deleting them.

Note
An *Inactive* Variable can be made *Active* by simply changing the Field **Inactive** from *TRUE* to *FALSE*

You will get a message before the Test really starts, which you must confirm by clicking **Yes.**



Customization Fields

It is possible to use customization Fields in the Formulas. Of course, for these Fields no Global Variables exist. So, they must always be added to the **Formula Global Variable** Table. This is enough for most customization Fields.

However, for some customization Fields in TRIMIT it is recommendable to create equivalents/ counterparts in the **Service Tables** if you want to use them in TRIMIT Formulas. For instance, if your Customer needs a Field on the *Sales Line* the Service Table ***API Sales Line*** needs to be created as well.

This is the case for customized Fields in the following Tables:

- *Sales Header*
- *Sales Line*
- *Purchase Header*
- *Purchase Line*
- *VarDim Order*
- *VarDim Item*
- *Production Header*
- *Production Line*

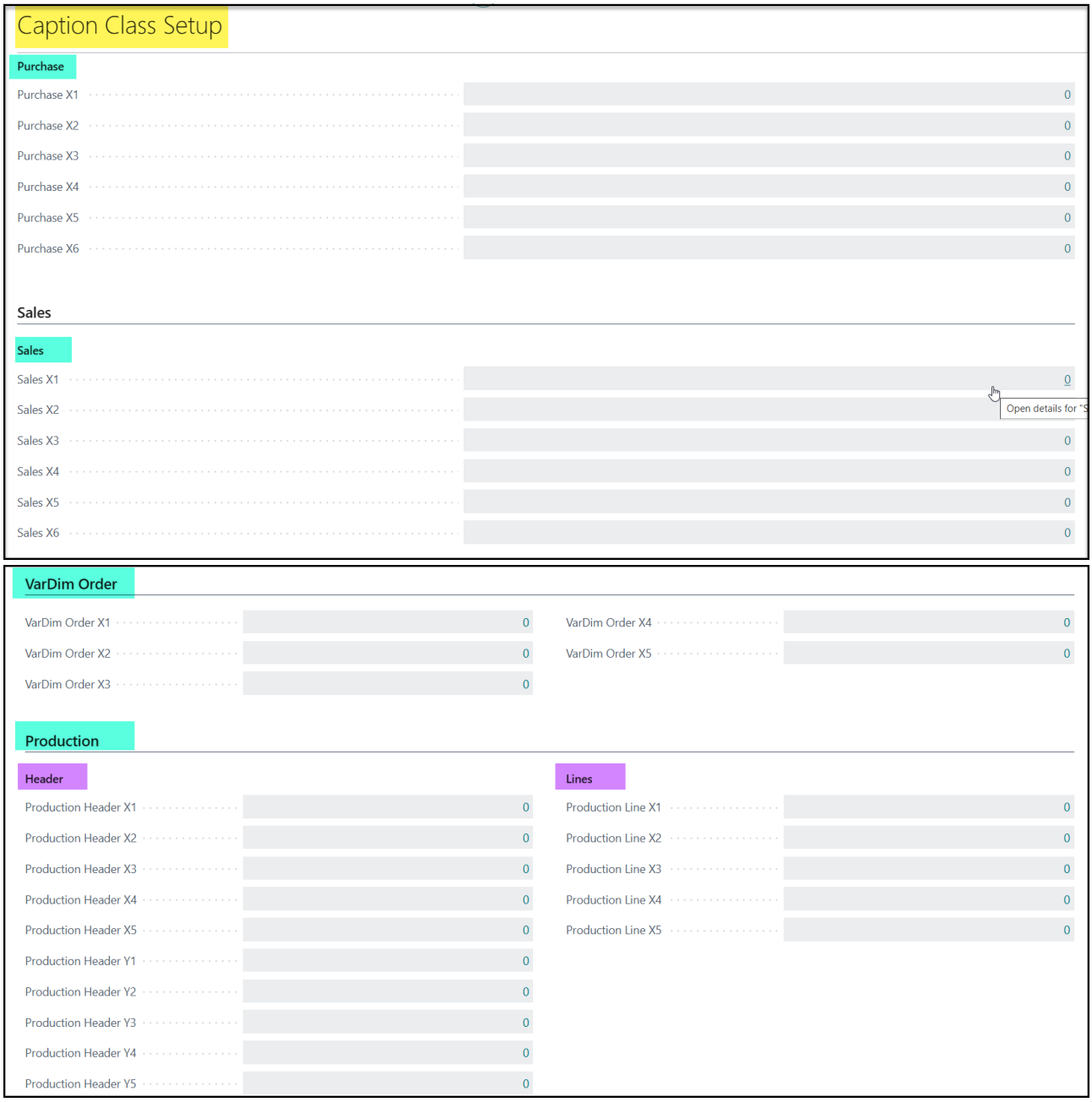
This issue is discussed more elaborately in *Appendix D: API Parameters*.

Global Variables Used for Extensive Calculations.

A Formula can consist of many Formula Lines in which complex calculations can be carried out to calculate Output Value. It is therefore very convenient that many Tables on which the Formulas can be active have extra Fields that can be used in the calculations for storing interim solutions. The Tables are:

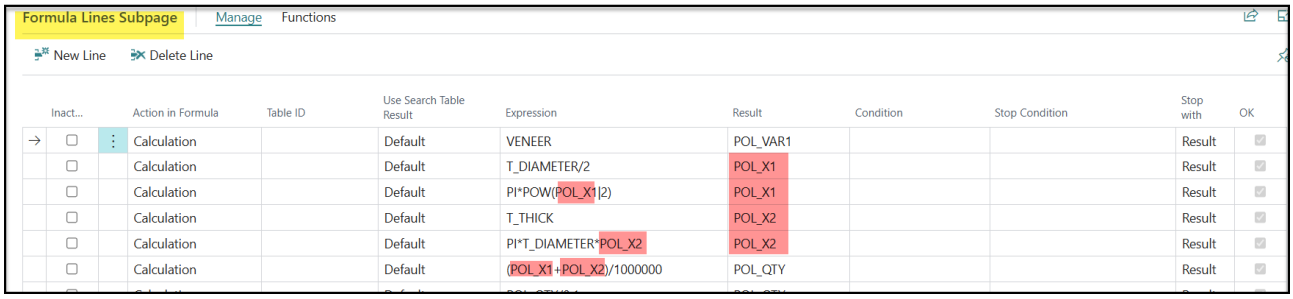
- Purchase Line
- Sales Line
- VarDim Order
- Production Header
- Production Lines

These Fields are called **X1-n** for *Decimal* Fields and **Y1-n** for *Code* Fields.
Like the **Item Statistic Groups 1-5** on the Master, these Fields can have their own Caption to be setup in the **Caption Class Setup**. This Caption can be multi-lingual. Below is a screenshot of the Caption Class Setup Fields that can be used in TRIMIT Formulas.



By clicking on the number in the Field, you can open the **Caption Class Setup** Table for entering the Caption per Language. After closing the number in the Field shows how many Captions are set for this Field.

These Fields have also **Global Variable** appointed so that they can be used in Formula Calculations. You could recognize these Global Variables via the format ***Reference Code Table_X1-n/Y1-n***. For instance:
POL_X1 for Field **X1** on the **Production Line**, or *SL_X2* for Field **X2** on the **Sales Line**.



These Global Variables can be used in all Fields **Expression**, **Result**, **Condition** and **Stop Condition**, also as a part of a calculation in the Expression Field.

The Use of VarDim Types as Variables

Introduction VarDim Types on Formula Lines

All **VarDim Types** can be used automatically as a Variable in a TRIMIT Formula. This is also the case for newly created VarDim Types. The only exception is the VarDim Type of **Type** *Headline*.

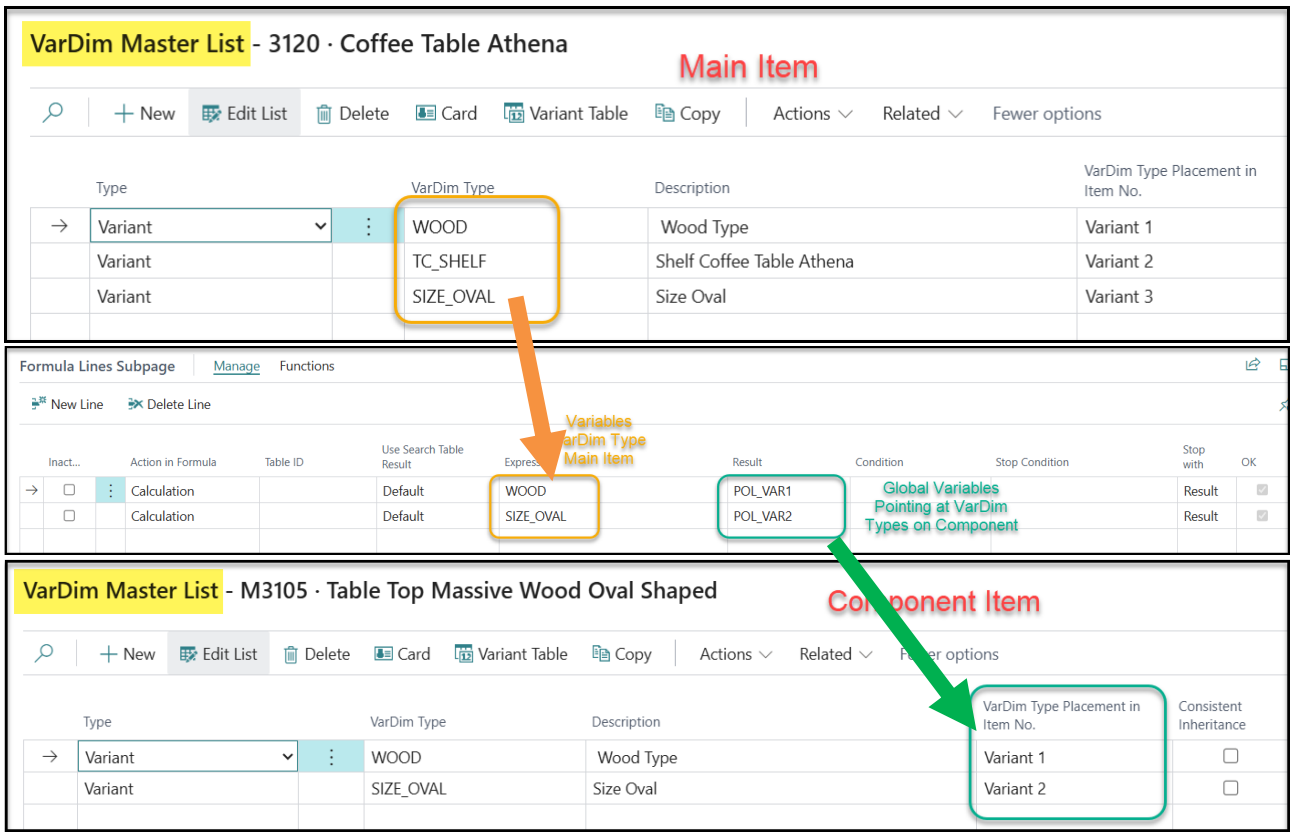
If a Formula Line with a VarDim Type is executed, the **Value** of the VarDim Type in the current **VarDim Order** or **VarDim Item** is taken. However, it is also possible to get other information from these Tables. In that case you must work with **VarDim Extensions**. This is explained in chapter *Configuration Formulas*.

VarDim Types on Formula Lines in BOM

With TRIMIT Formulas on BOM Component Lines, you often want to calculate the VarDim Type Values of Items in the underlying Production Order Lines. In those cases, it is not enough to have a VarDim Type in the Field **Result** on the Formula Line, because that **VarDim Type** refers to the **VarDim Order** of the current Item and not on the underlying Item. For this reason, you need to use the **Global Variables** *POL_VAR1-9* and/or *POL_DIM1-3*, where these Global Variables point at the VarDim Type Values of the Component (Item in the BOM). Note that these Global Variables point at the VarDim Types in the **VarDim Master** of the Component related to the Field **VarDim Type Placement in Item No.** This is shown on the Table below.

Global Variable pointing to VarDim Types on Production Line	VarDim Type Placement in Item No. in VarDim Master Component.
POL_VAR1	Variant 1
POL_VAR1	Variant 2
POL_VAR3	Variant 3
POL_VAR4	Variant 4
POL_VAR5	Variant 5
POL_VAR6	Variant 6
POL_VAR7	Variant 7
POL_VAR8	Variant 8
POL_VAR9	Variant 9
POL_DIM1	Dimension 1
POL_DIM2	Dimension 2
POL_DIM3	Dimension 3

The simplest example is to inherit the Value of a particular VarDim Type of a ***Main Item*** to the Value of the VarDim Type with identical **Options** on the ***Component Item***.



The setting in the example above means: the *Table Top* inherits the *Wood Type* and the *Table Top Size Oval* from the configuration Values of the *Coffee Table Athena.*

Of course, this is a simple inheritance calculation, but the **Result** can also be for a *Text Replacement*, *Search Table*, *Cross Calculation*, complex calculations or from multiple Formula Lines with all sorts of **Actions**. During the calculation, the Values of the Variables may change. The Output will be placed on the Production Line when all Formula Lines are executed. Only the last calculated Value will be placed back as Output.

Note:
To calculate **Variants** and/or **Dimensions** of the Components on the **Production Order Lines**, these Items must have an *intelligent* number based on *Master No. + VarDim Values* according to the setting in the **VarDim Master** Field **VarDim Type Placement in Item No.** This means that you cannot work with (partial) generic numbers for Raw Materials and Sub-assemblies.

When the **Variants** and/or **Dimensions** of a Component are calculated with Formulas, you must make sure that all **VarDim Type Values** of the Component are addressed properly. If one of them stays empty, the system cannot select/ create the appropriate Item on the **Production Order Line** and generates an error.

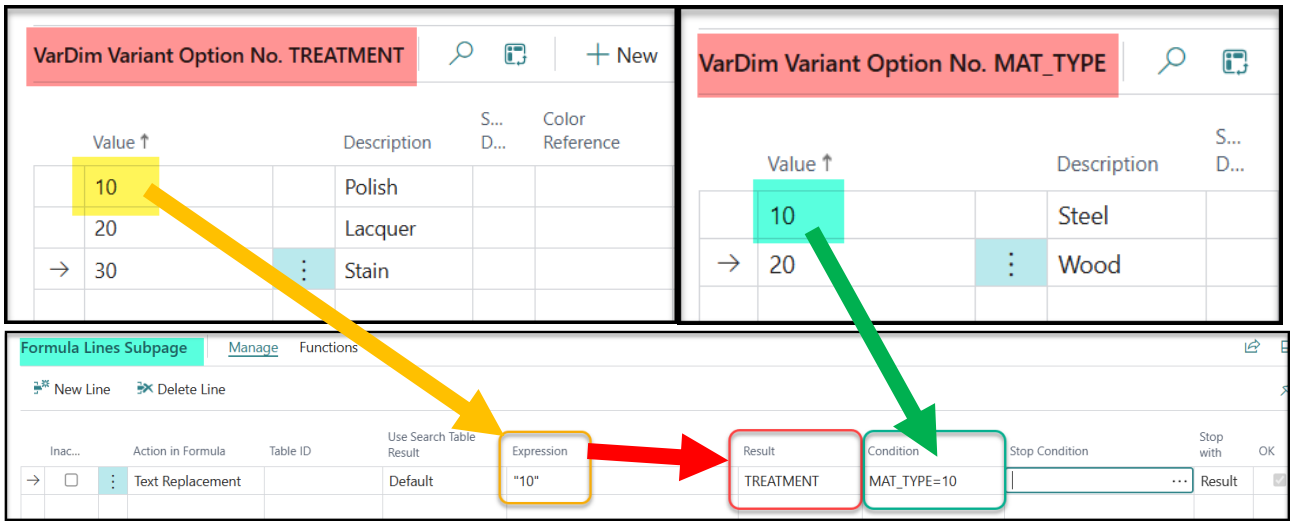
Tip:
Adress the **Output** Variables one-by-one. So, make sure that *POL_VAR1* is calculated first, then *POL_VAR2*, etc.! Do the same for the *POL_DIM1-3* until you have all VarDim Types of the Component Item determined. By the way: there is no technical reason why you must do it in this order. If there is a good reason to deviate from it, that is OK as long as you make sure that all VarDim Types of the Component are addressed. However, doing the things in a logical order makes the Formula easier to read and understand.

VarDim Types used as Variables in Configuration Formulas

When you are configuring an Item, it is possible to have a Formula executed, when you are entering a **Value** in one of the **VarDim Types** in the **VarDim Order**. This way you can manipulate one or more of the other answers.

An example:
Let us assume that we have a Chair of which the Legs can be made of Steel or Wood. Depending on the choice you make there could be a choice of Surface Treatment. Steel Legs need to be polished, while Wood has more possibilities such as Staining or Lacquering. So, if you choose Steel, then the answer in the VarDim Order for the VarDim Type that represents the Surface Treatment, should always be polished. It does not make sense to select from an Option list, which consists of one Value. The system can do that for you with a Configuration Formula.

After entering a **Value** in a particular **VarDim Type** in the **VarDim Order**, a Formula can force an answer into another VarDim Type based on a Condition that is also depending on a Value in the first answer or other earlier entered VarDim Types. In contradiction with the previous paragraph, in this case we manipulate the VarDim Types on the same level. Therefore, the VarDim Types can be used as an output Variable.



The Formula above says: if the **Material Type** is *Steel,* then **Surface Treatment** becomes *Polish*.

However, when configuration Formulas are executed, they may need to have Values from (other) VarDim Types. If those VarDim Types are not configured yet, you may get an Error message: *"VarDim Type XXXXX is in VarDim Item with assignment to Item YYYYYY Assignment2 None set to "W (Optional with Default)"[7] but could not be found in VarDim Order".*

This error message typically pops up when opening the configurator. Because when that happens there are no Vales yet entered in the VarDim Order. This message can be avoided by fulfilling one of the setup criteria below:

- The **VarDim Type** has *W (Optional with Default)* in the Field **Permanent or Optional** in the VarDim Item and the Field **Value** is populated.
  The Field must be found on the Item with **Item Type** *Master Item* if the **Item** has Variants and/or Dimensions as part of the **Item No.,** and on the actual **Item Type** of the Item used on the **Sales Line** for an Item without Variants and Dimensions as part of this Item No.

[7] Depending on content.

- The **VarDim Type** has option *No* in the Field **Calculate Price and Formulas Initially** in the **VarDim Item** (which is inherited from the VarDim Master).
  The Field must be found on the Item with an **Item Type** *Master Item* if the **Item** has Variants and/or Dimensions as part of the **Item No.**, and on the actual **Item Type** of the Item used on the **Sales Line** for an Item without Variants and Dimensions as part of the **Item No.**
- The **VarDim Type** itself has *TRUE* in the Field **Allow Blank Variant/Code** if **Type** is *Variant* or *Code or* **Allow Blank Dimension/Decimal** if **Type** is *Dimension* or *Decimal Figure.*

These Fields are more elaborately discussed in *Appendix A: VarDim Types*.

For more information about the setup of ***Configuration Formulas*** I refer to the chapter *Configuration Formulas*.

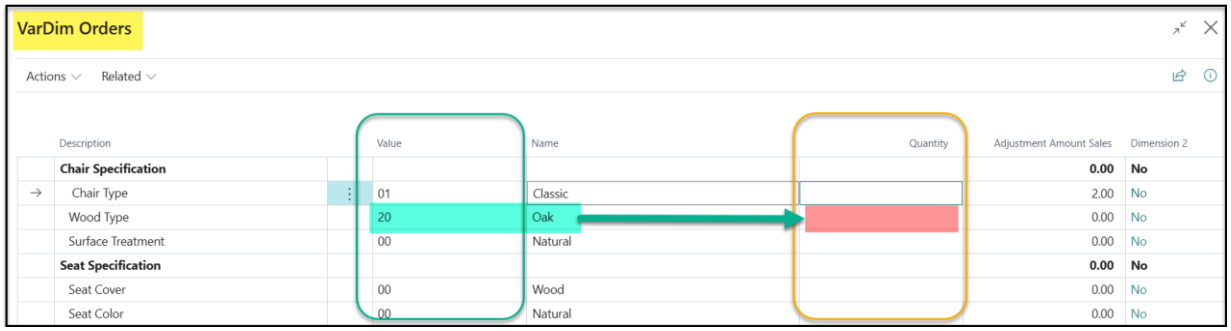
VarDim Extensions

VarDim Extensions can be used when you want to look for the content of a Field, which is not the **Value** of the VarDim Type itself, but of a Field in the record related to that VarDim Type in the **VarDim Item/Order**.

For instance: if we want to get the Value of the ***Wood Type*** in a Formula, we use the VarDim Type *WOOD* to return the Value *20*. However, if we do not want the **Value**, but the Field **Quantity** to return, then we need to use a **VarDim Extension**. The Extension will be placed between brackets behind the VarDim Type in the Formula. The format will be:

VarDim Type[Extension].

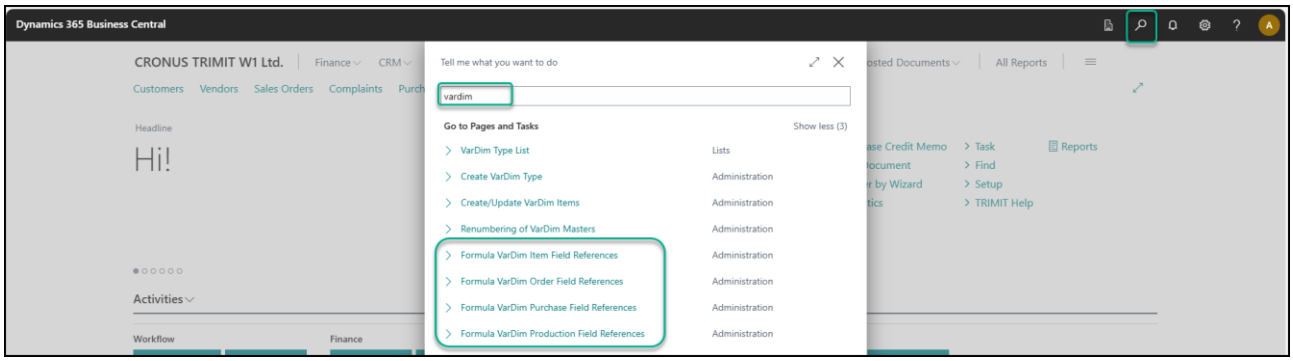
In the example below it would be *WOOD[VDO_QTY]*.

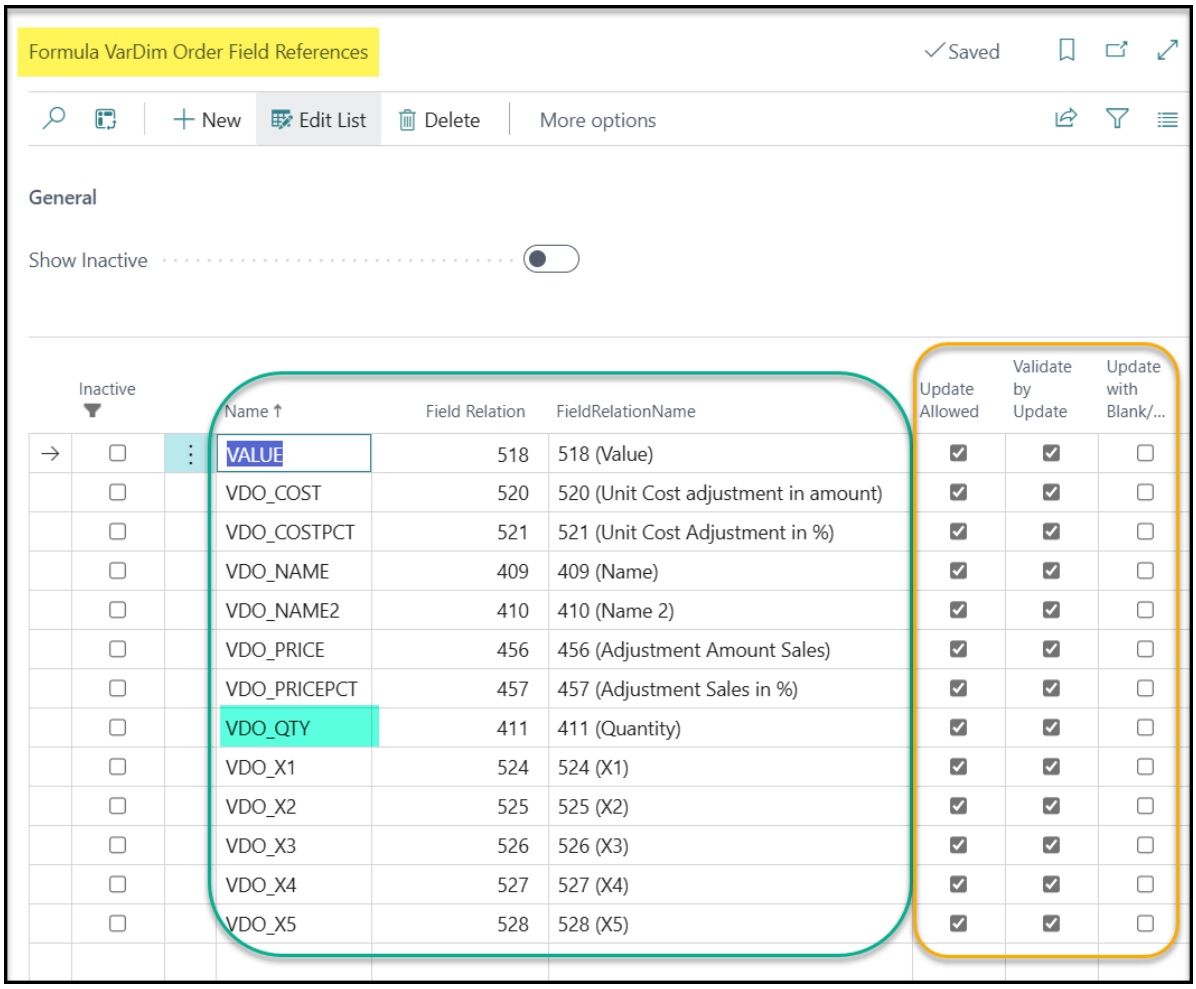


The VarDim Type Extensions can be used on different Table records:

Table	Extension Code	Reference Table Page	VarDim Table
Item (Current Level)	[VDIC_...]	Formula VarDim Item Field References	VarDim Item
Item (Overlying Level)	[VDIO_...]	Formula VarDim Item Field References	VarDim Item
Item (Start Level)	[VDIS_...]	Formula VarDim Item Field References	VarDim Item
Sales Order Line	[VDO_...]	Formula VarDim Order Field References	VarDim Order
Purchase Order Line	[VDPP_...]	Formula VarDim Purchase Field References	VarDim Prod/Purchase Line
Production Line (Current Level)	[VDPC_...]	Formula VarDim Production Field References	VarDim Prod/Purchase Line
Production Lines (Overlying Level)	[VDPO_...]	Formula VarDim Production Field References	VarDim Prod/Purchase Line

The pointing of the Extensions Variables works the same way as Global Variables. In the **VarDim Reference** Tables we create an Extension **Name** that we relate to a particular Field of the Tables as in the column **VarDim Table** in the Table above. In the screenshot below, you see the Extensions created for the Fields in the VarDim Order.
You can find the VarDim Field References via **Search**:





Inactive	Name	Field Relation	FieldRelationName	Update Allowed	Validate by Update	Update with Blank/...
☐	VALUE	518	518 (Value)	☑	☑	☐
☐	VDO_COST	520	520 (Unit Cost adjustment in amount)	☑	☑	☐
☐	VDO_COSTPCT	521	521 (Unit Cost Adjustment in %)	☑	☑	☐
☐	VDO_NAME	409	409 (Name)	☑	☑	☐
☐	VDO_NAME2	410	410 (Name 2)	☑	☑	☐
☐	VDO_PRICE	456	456 (Adjustment Amount Sales)	☑	☑	☐
☐	VDO_PRICEPCT	457	457 (Adjustment Sales in %)	☑	☑	☐
☐	VDO_QTY	411	411 (Quantity)	☑	☑	☐
☐	VDO_X1	524	524 (X1)	☑	☑	☐
☐	VDO_X2	525	525 (X2)	☑	☑	☐
☐	VDO_X3	526	526 (X3)	☑	☑	☐
☐	VDO_X4	527	527 (X4)	☑	☑	☐
☐	VDO_X5	528	528 (X5)	☑	☑	☐

BOM Level VarDim Item / Production

The Field **BOM Level VarDim Item** can only be used in combination with **Formula VarDim Item Field References** and **Formula VarDim Production Field References**. The purpose is to control on which BOM Level the record should be found.

We have the following options:

Start Level	This option is only used in connection with **VarDim Item.** **Start Level** is the Initiating Level. If you call a Formula from a BOM Component Line in a Related Order structure Start Level is the Sales Line.
Overlying Level	Example: If you call the Formula from a BOM Component Line the **Overlying Level** is the Production Header.
Current Level	The Level of the Position where you call the Formula

If the Field is 'blank' then *Current Level* is the default Value.

In the **VarDim Field Reference Table** you can also set a checkmark, if the Value can be Updated, Validated or when the update is possible with a Blank or Zero Value. These Fields are stored in Table ***6037197 TRIMIT Service Parameters***.

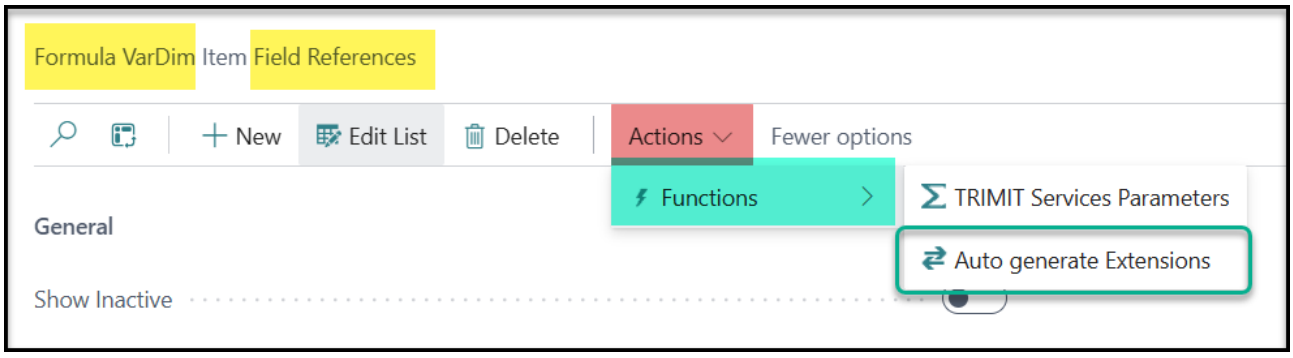
Extensions with a Direct Field Reference

In *Appendix A* it is described that VarDim Types can also be of **Type** *Code*, which allows you to have a specific cross-section of an existing Table in BC/TRIMIT. If the **Table Relation** is filled with Table *27* (*Item*), it is possible to point directly to a Field in the Item Table and get the Value returned in a **Global Variable** in a **Formula** using **Field No.** from the Item Table as an Extension to the VarDim Type.

Let us discuss an example here to show how this looks like:
Let us assume that we have a **VarDim Type** *ITEM1* of **Type** *Code related to* ***Table*** *27 (Item).*
The ***Item Card*** has a Field **Measurement 6** with **Field No**. *6037362.*
The **Expression** in the Formula will be: *ITEM1[6037362],* which you can put in the Global Variable in **Result**.

Auto Generate Extensions

Click ***Actions***, ***Functions, Auto generate Extensions*** in the **Formula VarDim . Field Reference** Card*.* This way you can automatically generate all the standard Extensions for ***VarDim Order***, ***VarDim Item***, ***VarDim Purchase*** and ***VarDim Production*** that are commonly used within the Formulas. The Action can be found on all ***VarDim Extension*** pages. You will get a message, which must be confirmed by clicking **OK**.



Note:
In connection with a new installation, Codeunit ***6036550 trm General*** will execute the auto generation of the commonly used Extensions automatically.

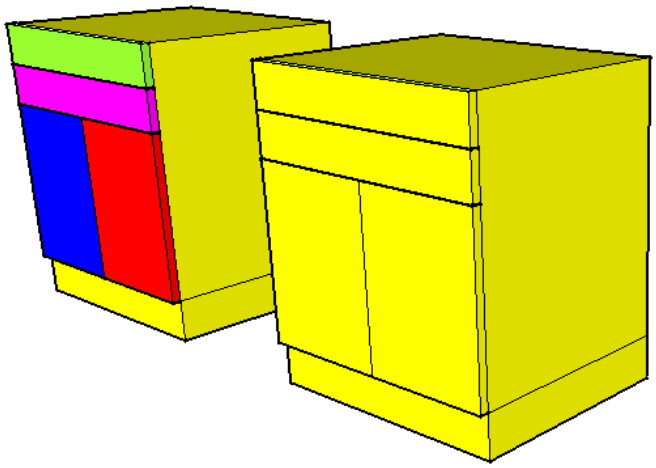
VarDim Positions

Both **VarDim Item** and **VarDim Order** are three-dimensional. To every **VarDim Type** on the first Level there can be a **2. Dimension** attached with a number of Positions on the second Level, and to every Position on the second Level there can be a **3. Dimension** attached with a number of Positions on the third Level.

The Names of the **2. Dimension** and **3. Dimension** are written manually at the creation of the Dimension. The **Value** that can be chosen for **VarDim Type** of **Type** *Variant* is directly related to the Values that can be chosen on the **VarDim Type** these Dimensions are connected to (Taken the Positive/Negative Lists and Overall/Subordinate List in consideration).

How the **Positions** in **2nd Dimension** and **3rd Dimension** are set up is described in more detail in the ***White Paper - TRIMIT Item Structure and Configuration***.

Case:
A Cabinet consists of two Drawers and two Doors. It is normally sold in one chosen Color, but the Customer can individually choose different Colors for the Drawer section and the Door section. However, it should also be possible to choose a Color for every individual Drawer and/or Door.



To control this, we need **2. Dimension** for the sections and **3. Dimension** for the individual Components.

- In the second Dimension of **VarDim Type** *CAB_COLOR* (= Cabinet Color), we have two positions: *DRAWERS* and *DOORS*.
- In the 3rd dimension of *DRAWERS*, we define two positions: *DRAWER1* and *DRAWER2*.
- In the 3rd dimension of *DOORS*, we define two positions: *LEFT* and *RIGHT*.

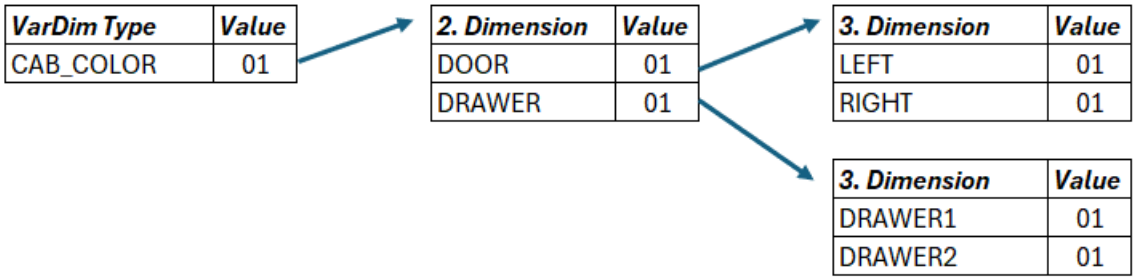
VarDim Type	*Value*
CAB_COLOR	

2. Dimension	*Value*
DOOR	
DRAWER	

3. Dimension	*Value*
LEFT	
RIGHT	

3. Dimension	*Value*
DRAWER1	
DRAWER2	

By entering the **Value** on the **VarDim Type,** the Value from the **VarDim Type** can be set automatically on the **2. Dimension** and the **3. Dimension**.



VarDim Type	Value
CAB_COLOR	01

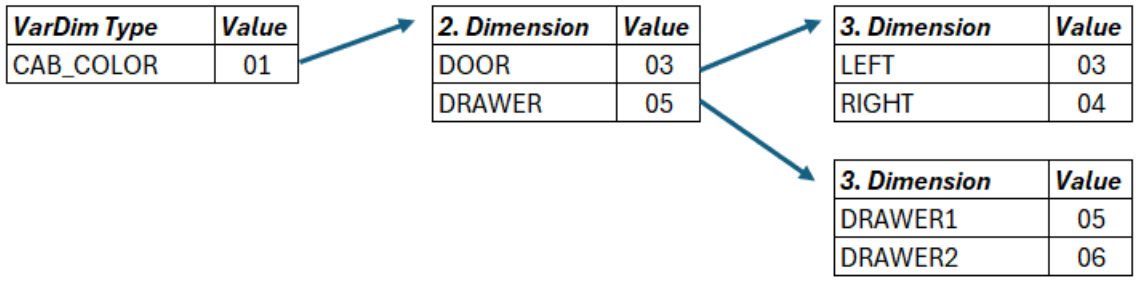
2. Dimension	Value
DOOR	01
DRAWER	01

3. Dimension	Value
LEFT	01
RIGHT	01

3. Dimension	Value
DRAWER1	01
DRAWER2	01

Note:
This only works this way, if the Value in the Field **Permanent or Optional** is *W (Optional with Default)*. In that case the Value in the VarDim Type will be inherited to the Positions but can be overwritten.

If the Customer has individual Color demands the **2.** and **3. Dimension** can be called and the Values changed in connection with the Order entry.



VarDim Type	Value
CAB_COLOR	01

2. Dimension	Value
DOOR	03
DRAWER	05

3. Dimension	Value
LEFT	03
RIGHT	04

3. Dimension	Value
DRAWER1	05
DRAWER2	06

If we put the VarDim Type on a Formula Line, it points to the *CAB_COLOR*. However, if we want to retrieve or manipulate the Values in the 2nd and/or 3rd Dimension, we need to have these positions in the Formula according to the following format:

VarDim Type[POSITION (2nd)] or
VarDim Type[POSITION (2nd), POSITION (3rd)]

Examples of the case above:

Initial Level: *CAB_COLOR*
2. Dimension: *CAB_COLOR[DOOR]*
CAB_COLOR[DRAWER]
3. Dimension *CAB_COLOR[DOOR,LEFT]*
CAB_COLOR[DOOR,RIGHT]
CAB_COLOR[DRAWER,DRAWER1]
CAB_COLOR[DRAWER,DRAWER2]

This Position format can be extended, if you also want to include Field **Quantity**, which is also a Field in the VarDim Order Dimension Table. In that case the format would be:

VarDim Type[POSITION (2nd),Quantity] or
VarDim Type[POSITION (2nd), POSITION (3rd), Quantity]

Examples of the case above (with the same initial Level):

2. Dimension:	*CAB_COLOR[DOOR,Quantity]*
	CAB_COLOR[DRAWER,Quantity]
3. Dimension	*CAB_COLOR[DOOR,LEFT,Quantity]*
	CAB_COLOR[DOOR,RIGHT,Quantity]
	CAB_COLOR[DRAWER,DRAWER1,Quantity]
	CAB_COLOR[DRAWER,DRAWER2,Quantity]

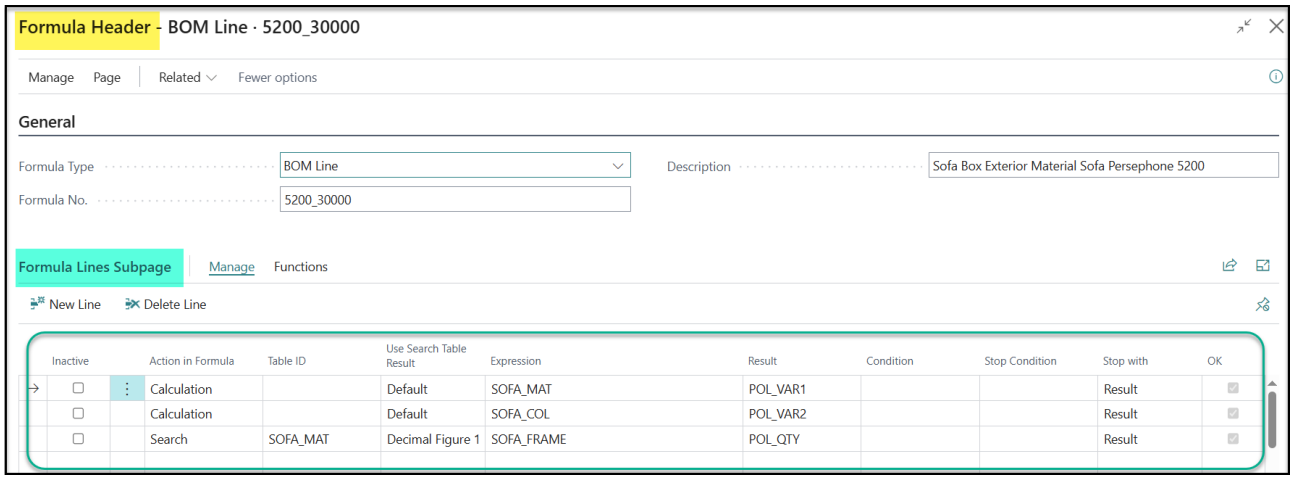
In connection with the Formulas that are used at the creation of the BOMs (Production Order Lines) it is possible to use a question mark ("?") as Position pointer. This is applicable for Fields **Part Item No, Position 1, Position 2,** and **Position 3** in the BOM Component Lines and therefore also in the Production Order Lines. At the Formula execution, the question marks will be replaced by the Values in the BOM Fields **Part Item No., Position 1, 2** and **3**, where after the properties will be inherited from these Positions in **VarDim Item** and **VarDim Order**. This creates the opportunity to use the same Master BOM at the Underlying Level.

Formula Lines Structure

Introduction

In chapter *Formula Structure* it is described that a Formula exists of three parts:
Formula Header, Formula Lines and Triggers. In this chapter Formula **Lines** are discussed.

Every Formula Line acts as a piece of code. There is an Input required, which is converted to a Result.
After all Formula Lines have been executed the **Results** are put back as Output into the Database.
However, to avoid actual coding the Formula Lines consist of columns in which we put Data.
The system knows what to do with that Data depending on the function of the column.
We could say that creating Formulas is like coding without code!



This has particular advantages:

1. A skilled user can create logic in Formulas without having development skills of BC/TRIMIT.
2. Because the logic is written as Data, there are no upgrade issues, because upgrading only affects actual code.
3. Creating business (or configuration) logic can be very flexible and supports a high level of complexity.
4. Formulas may replace customizations in specific circumstances.

In the paragraphs below the columns in Formula Lines are discussed.

Line No.

The integer for **Line No.** is the unique identifier of a **Formula Line**. It is auto generated and comparable to the Line Nos. in all kinds of Subpages which you can find in Orders. **Line No.** can be used in combination with Field **Go To**. This is described in the paragraph *Go To*.

By default, **Line No.** is not visible. It can be made visible on screen via ***Personalize***.

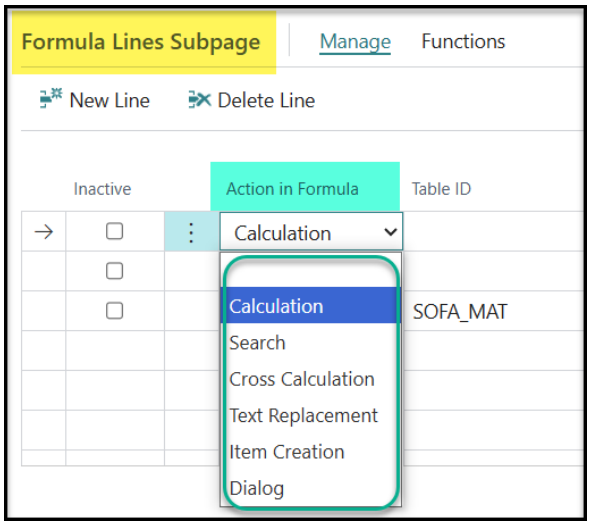
Inactive

The Boolean **Inactive** can be used to deactivate a Formula Line. If **Inactive** is *TRUE*, the Formula Line is excluded in the Formula evaluation. The default value of this Field is *FALSE*.

This Field can be used when a Formula needs to be tested, and some Lines should be excluded from the test. This is especially useful to investigate possible flaws in the Formula, if the Output is not what is expected.

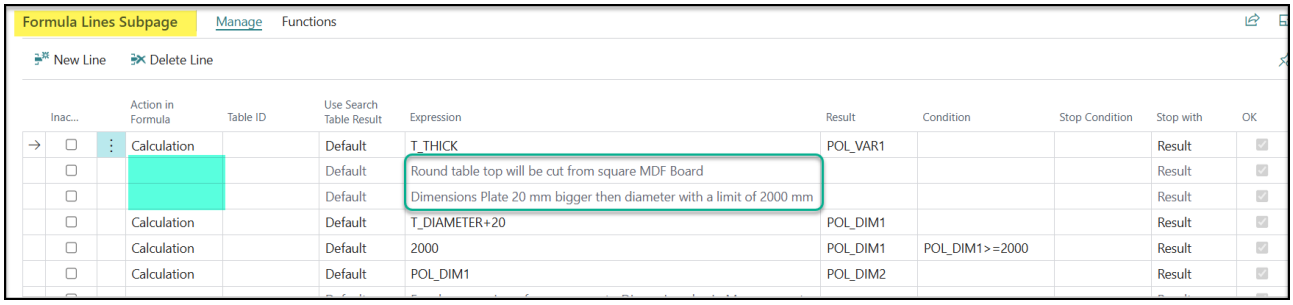
Action in Formula

The **Action in Formula** is an Option Field and describes how Data is found and converted to Output in different situations.
Which actions that should be carried out depend on what you want to do.
You can choose between the following seven options:



Blank

If **Action in Formula** is "*Blank*", the **Formula Line** is skipped when executing the Formula.
A Formula Line with **Action in Formula** *Blank* is normally used to insert short explanations for the Formula of the following Line/Lines. These explanations are entered into the Field **Expression**.



The **Action in Formula** "*Blank*" can also be used to deactivate a Formula Line. However, we recommend using the Boolean **Inactive** on the Formula Line instead, as this keeps all information on the Formula Line untouched. For more information about Expression go to paragraph *Expression*.

Calculation

If **Action in Formula** is *Calculation*, this Field works together with the Field **Expression**.
A Formula Line with the **Action in Formula** *Calculation* can be used for:

- Extracting a Value from a Table in the Database. This could be a **VarDim Value** or a **Global Variable**.
- Making a calculation with VarDim Values or Global Variables, which are extracted or calculated in the Formula Lines above the current one.
- Entering a **fixed decimal figure**, which cannot be retrieved from the Database.
- Executing a pre-defined **Function**[8].

[8] For more information about Functions: see *Appendix B: Functions*.

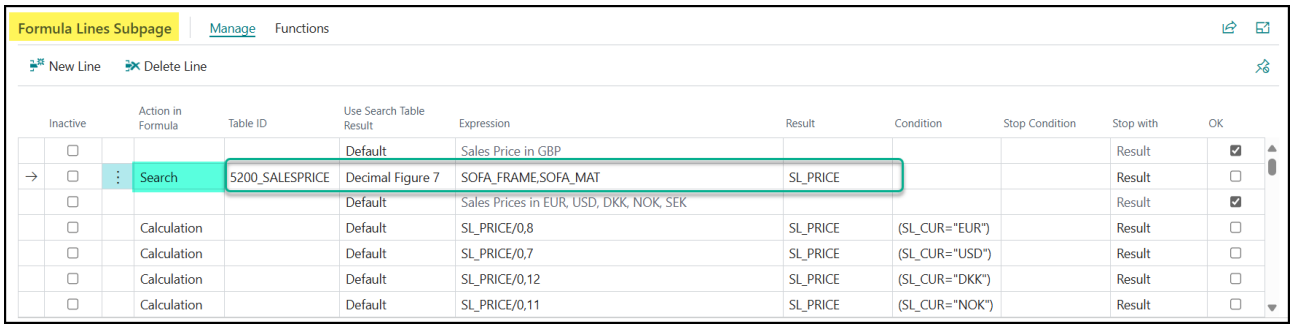
More information about the Field **Expression** and **Result** can be found in the paragraph *Expression* and *Result*.

Search

If **Action in Formula** is *Search*, the Formula Line works with a **Search Table**, you need the following Fields in order to work with the **Search Table** properly:

- **Table ID** — The Table that is used for the search.
- **Expression** — To define (the combination of) the Input Variable(s)
- **Result** — The Variable to put in the Result of the search.
- **Use Search Table Result** — To define the proper Output column from the Search Table

Note
Even when the contents of the Field **Expression** are used as Input for a **Search Table**, it still may contain mathematical expressions or fixed Values identical to **Action in Formula** *Calculation*.

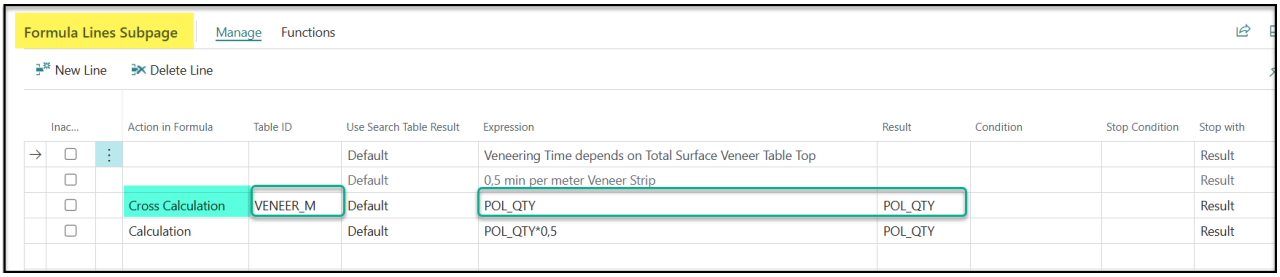


How Search Tables work and how they are set up is discussed in chapter *Search Tables*.
See also the paragraphs for *Table ID*, *Expression*, *Result* and *Use Search Table Result*.

Cross Calculation

The **Action in Formula** *Cross Calculation* can only be used in connection with Formulas on Master/Item BOM's. A Formula on a BOM Component Line is always pointing at this BOM Component line.
With a *Cross Calculation* it is possible to work with Variables from other BOM Component Lines.

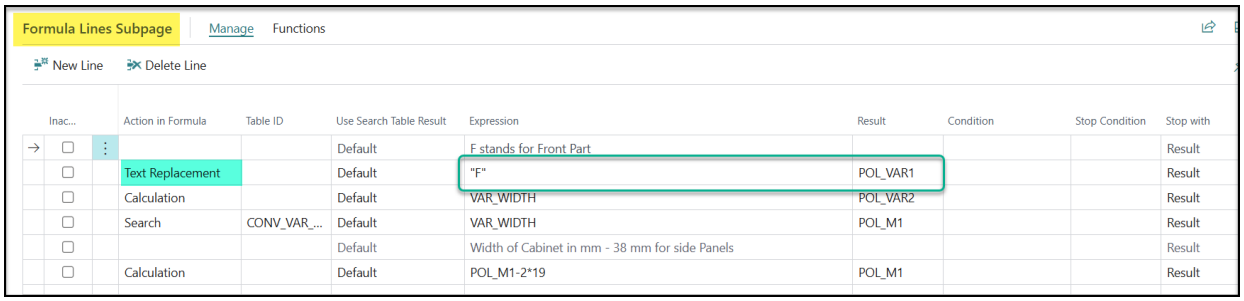
The *Cross Calculation* works together with the Field **Table ID** in which we can select the appropriate **Cross Calculation**. The Input for this *Cross Calculation* comes from the Field **Expression**. The Output will be placed in the Field **Result**.



You can find more information in chapter *Cross Calculation*.
See also the paragraphs for *Table ID*, *Expression* and *Result*.

Text Replacement

One of the possibilities of **Action in Formula** *Calculation* is that you can give a fixed Value to a Global Variable pointed at a Decimal Field. However, what happens if that **Global Variable** points at a *Code* or *Text* Field? In that case you must work with **Action in Formula** *Text Replacement*. The Text string that you want to put in the Global Variable in Field **Result**, should be put into the Field **Expression**. Make sure that the Text string is between "" (double quotes).



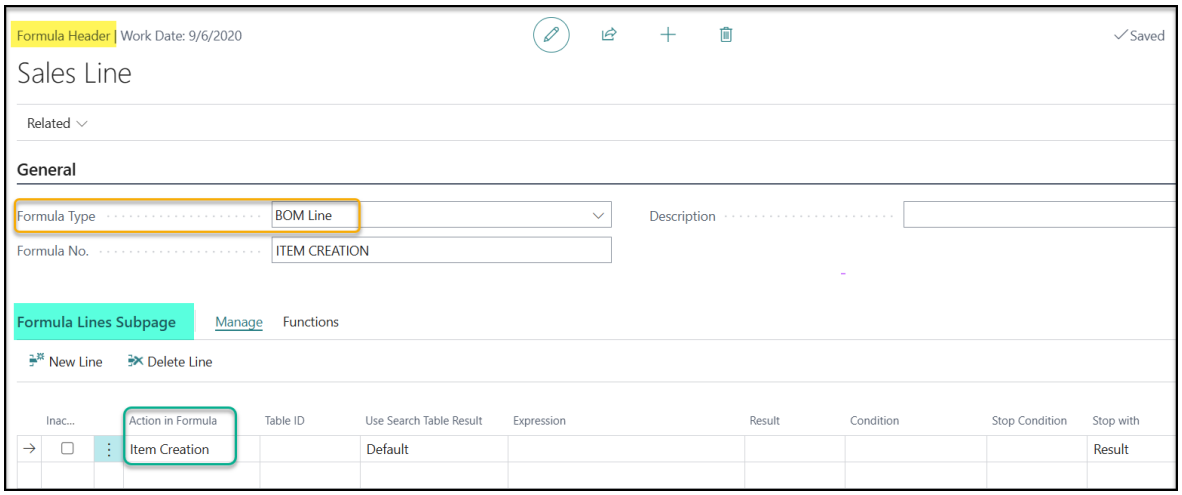
Note
A **VarDim Type** Value is always seen as Code, so in case of a manual input of a **VarDim Variant Option** Value or a Decimal Value in a **VarDim Type** in the Field **Result** on a Formula Line you always must use *Text Replacement.*

For more information about the Fields **Expression** and **Result**: paragraphs *Expression* and *Result*.

Item Creation

If an Item on the **BOM Component Line** is a *Master Item*, a Formula execution should result in creation of a specific **Item Number**. When the Formula provides the relevant properties, TRIMIT will automatically generate the specific **Item Number** in connection with the Formula execution, and it will commit it in the Database after the Formula has been executed.

However, in some scenarios the new Item needs to be committed to the Database earlier.
For instance: when in later stages the execution of the Formula is used to enter Field Values on the newly created Item or the related **VarDim Item**. In those cases, we can force the creation of the new Item before the Formula execution has ended by using **Action in Formula** *Item Creation*.



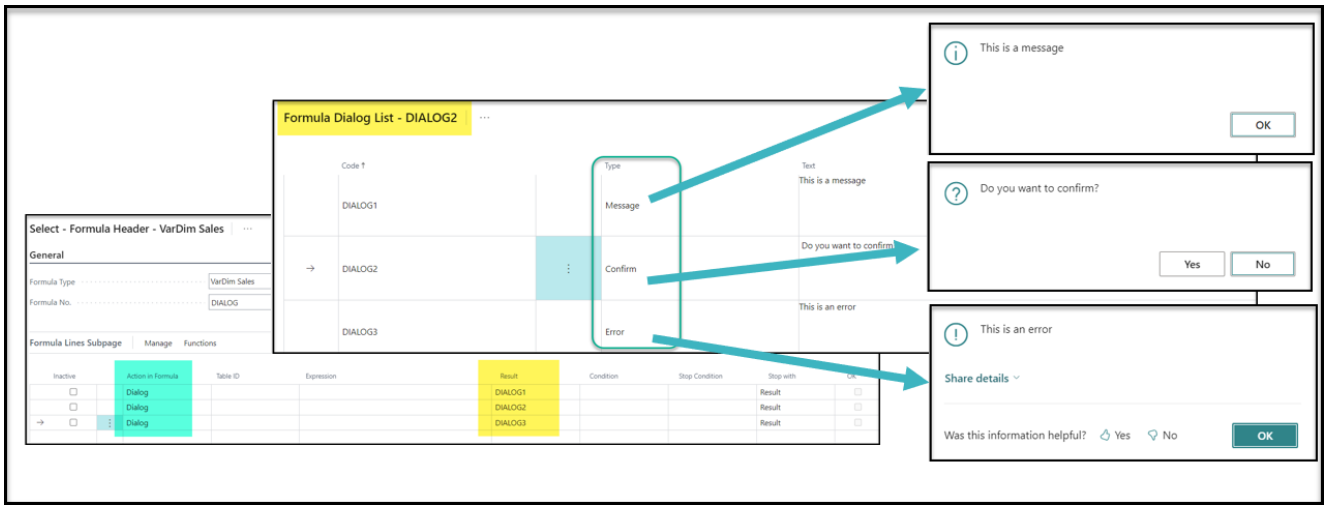
Note
The **Action in Formula** *Item Creation* can only be used in connection with Formulas on **Master/Item BOMs**.

Dialog

A Formula Line with the **Action in Formula** *Dialog* makes it possible to show a Message, an Error, or a Question for confirmation on the screen. The Dialog is shown when a particular condition (in the Field **Condition**) is met. (For more about Conditions see paragraph *Condition*)
The **Action in Formula** *Dialog* cannot be used in connection with Formulas connected to a Master/Item BOM Component Line and is typically used in connection with configuration Formulas.

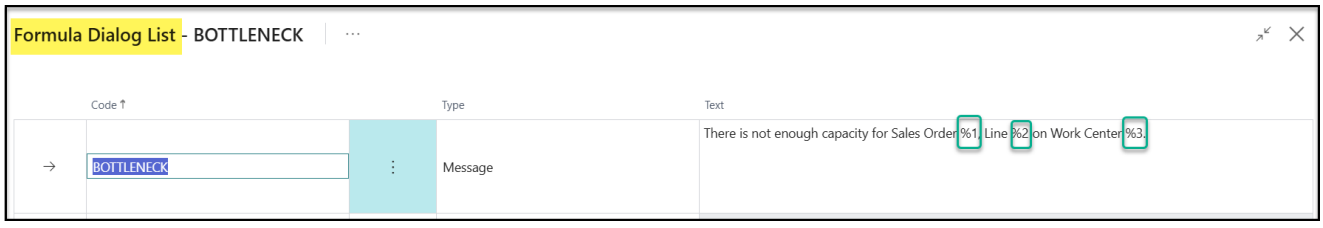
There are three types on Dialogs:

- *Message* — After clicking *OK*, you can continue.
- *Confirmation* — By clicking *Yes*, you can continue, but when clicking *No*, the process is interrupted.
- *Error* — After clicking *OK*, the process is interrupted.



Dialogs are set up in the **Formula Dialog List** (which can be found via ***Search)*** and put on the Formula Line in the Field **Result**.

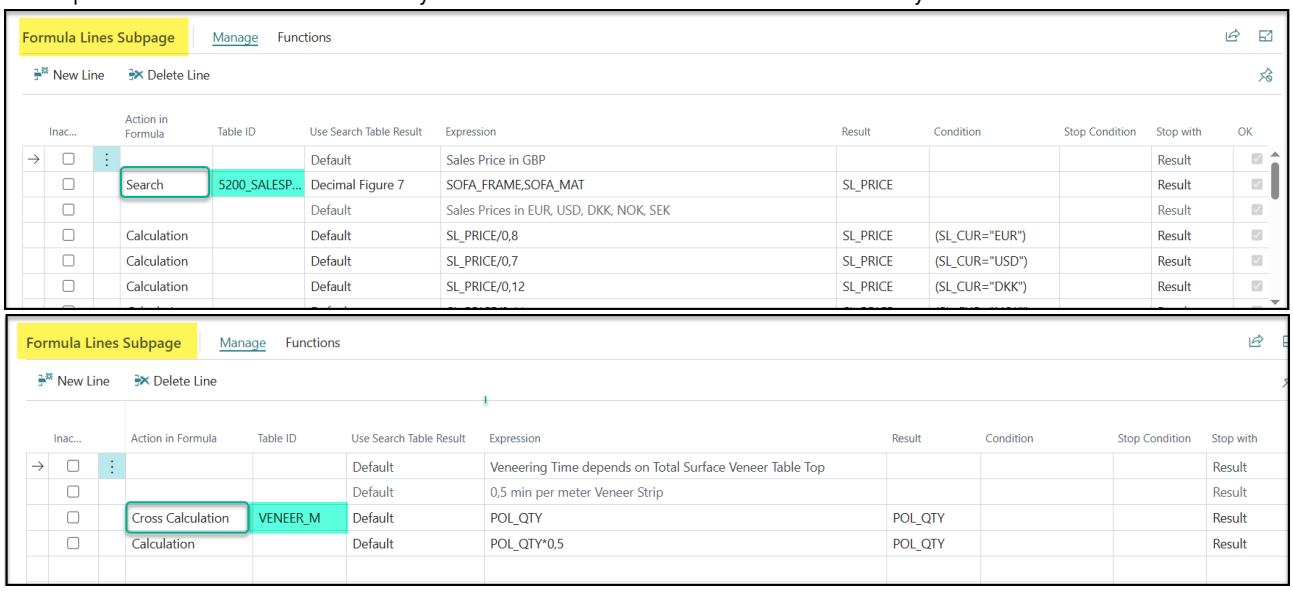
It is possible to work with reference Fields (%x) that correspond with particular Field Values. You can see an example below. (At the moment, this feature is not fully supported in TRIMIT and requires some customization)



Note
Dialogs are not supported on TRIMIT e-commerce.

Table ID

The *Code* Field **Table ID** is used in connection with the **Action in Formula** *Search* or *Cross Calculation* where you must enter the number of the **Search Table** or the **Cross Calculation,** respectively. The lookup in this Field is controlled by the Field **Action in Formula** and leads you to the correct Tables.



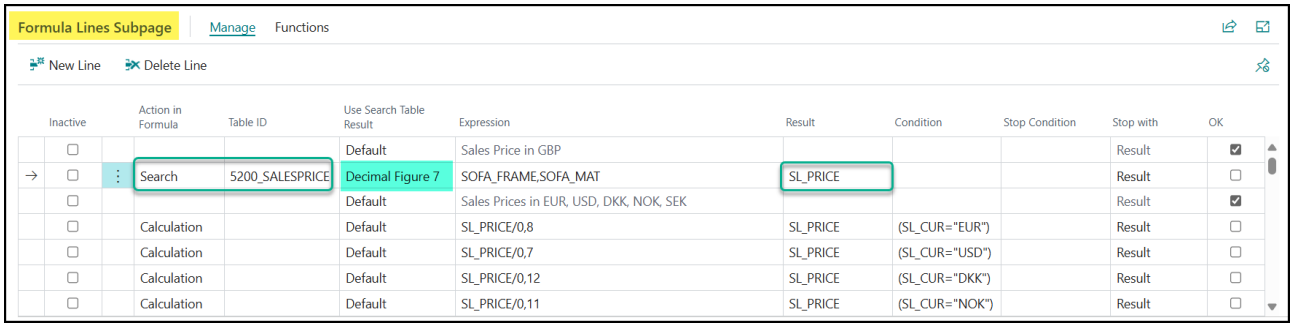
How to create a **Search Table Header** is described in the chapter *Search Table* and how to create a **Cross Calculation** is described in the chapter *Cross Calculation*.
See also the paragraphs for *Action in Formula*, *Expression*, *Result* and (in case of Search Table) *Use Search Table Result*.

Use Search Table Result

The Field **Use Search Table Result** can only be used in connection with the **Action in Formula** *Search*. It is an Option Field in which the options correspond with specific Fields in the **Search Table Setup** of the **Search Table** in Field **Table ID**. The options are:

- *Default*
- *Decimal Figure 1 – 20* (20 options)
- *Code 1 – 10* (10 options)

When creating a Formula Line, the default Value will be *Default*. This Field controls which Column Value in a **Search Table Line** will be returned into Field **Result**. The Field **Use Search Table Result** is required when there is more than one Result column in the Search Table of the same Type (*Decimal* or *Code*).



By default, the Field **Use Search Table Result** is invisible. You can get it on the page via ***Personalize***.

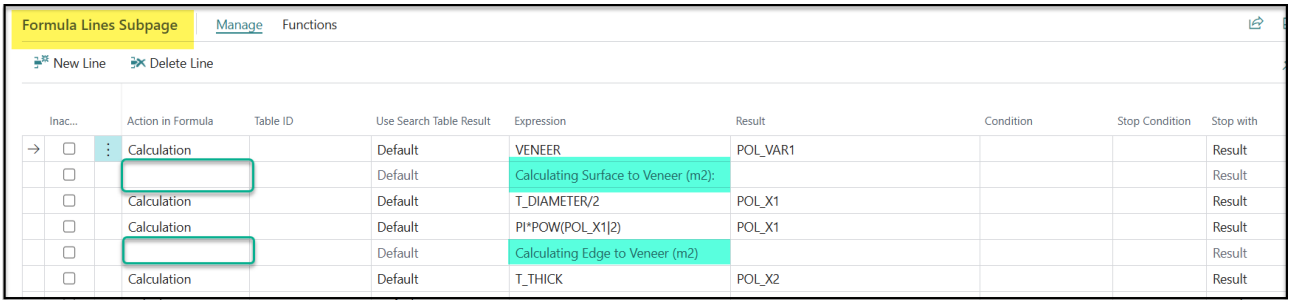
For more information about Search Tables, go to chapter *Search Tables*.
See also the paragraphs for *Action in Formula*, *Table ID*, and *Result*.

Expression

Field **Action in Formula** determines how the (*Text*) Field **Expression** is used.
You can choose between the following options below, clarified by several examples with different **Actions in Formula** beneath the Table.

Action in Formula	Expression
Blank	Free Text. (The Formula Line is 'greyed-out'.)
Calculation	Mathematical expression (consisting of single or multiple Variables and/or Mathematical Symbols or Functions) that results in a Value.
Search	Search expression with up to 5 Variables (Maximum 3 of **Type** *Decimal* and maximum 5 of **Type** *Code*, depending on setup of **Search Table Header**). Individual Variables are comma-separated.
Cross Calculation	Mathematical expression (consisting of a single or multiple Variables and/or Mathematical Symbols) that results in a Value.
Text Replacement	Expression that results in a Text string. A fixed Text must be surrounded by **"**
Item Creation	Blank (no expression allowed).
Dialog	Blank (no expression allowed).

In the example below, we can see an example of a '*Blank*' **Formula Lines** with some *free Text* in the **Expression**, which clarifies the idea behind the Formula Lines below them. Note that the '*Blank*' Formula Lines are greyed-out.

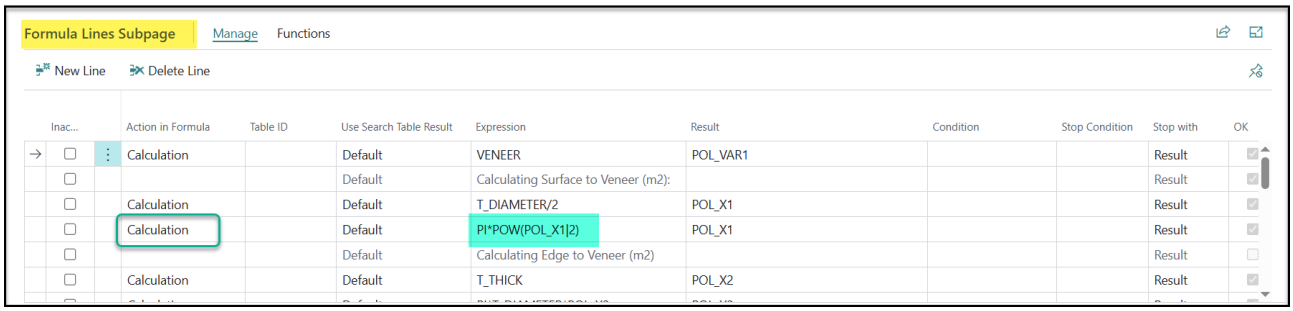


With **Action in Formula** *Calculation*, the Expression could contain one or more Variables, Mathematical Symbols, and/or Functions. Below we see some examples in Formula Lines:

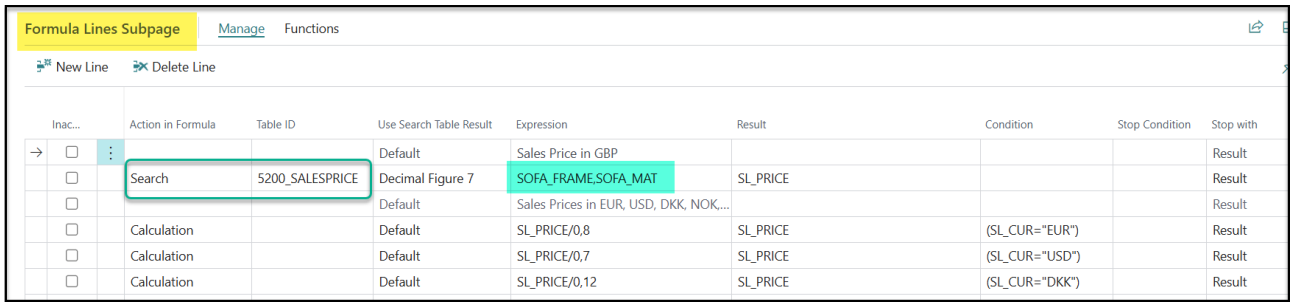
The following Mathematical Symbols can be used:

Symbol	Description		Symbol	Description
+	Addition		-	Subtraction
*	Multiplication		/	Division
(	Left Parenthesis		)	Right Parenthesis
,	Comma		POW	Raise to Power

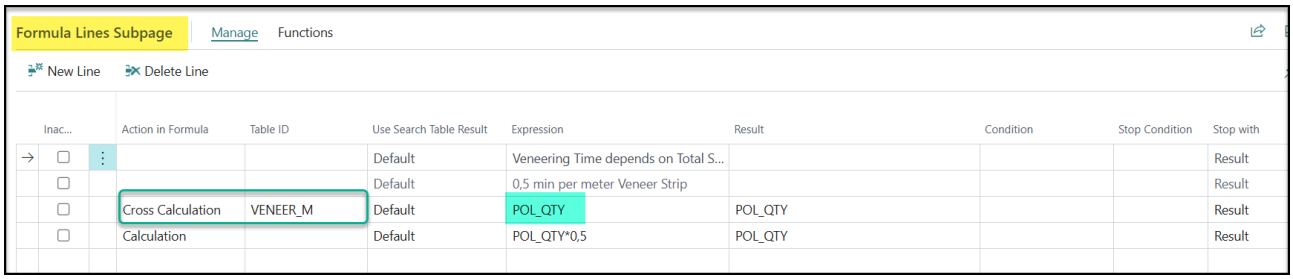
The format for *Raise to Power* is ***POW****(Base number|Exponent)*[9]. The standard internationally accepted Mathematical axioms that prioritize the actions within a calculation, are applicable in TRIMIT. Below is an example where *Raise to Power* is applied for calculating the surface of a circle (πr^2), where the radius (r) is represented by **Global Variable** *POL_X1*. In this example the squaring also goes together with a TRIMIT **Function** called *PI* that returns the number π in 15 decimals.
For more information about TRIMIT Functions: *Appendix B: Functions*.



If the **Action in Formula** is *Search*, then it is possible to go into the **Search Table** (in Field **Table ID**) with a combination of Values. These are set in the Field **Expression** comma separated. Below we see an example of this. More information about Search Tables is discussed in chapter *Search Tables*.

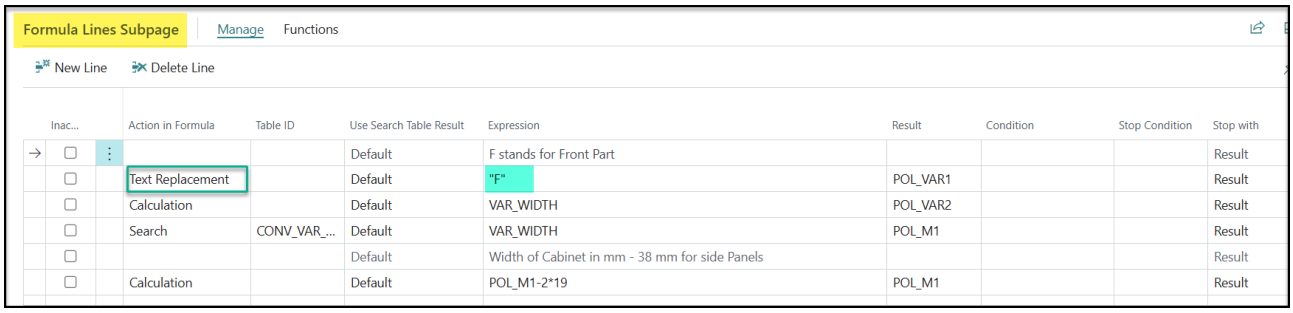


If the **Action in Formula** is *Cross Calculation*, then it is possible to get a Value from a Field of another **BOM Component Line**. In the Expression we will enter the **Global Variable** of the Field that we are looking for in this other Line. This has been set up in the Cross Calculation in the Field **Table ID**. More information about Cross Calculations can be found in chapter *Cross Calculations*.

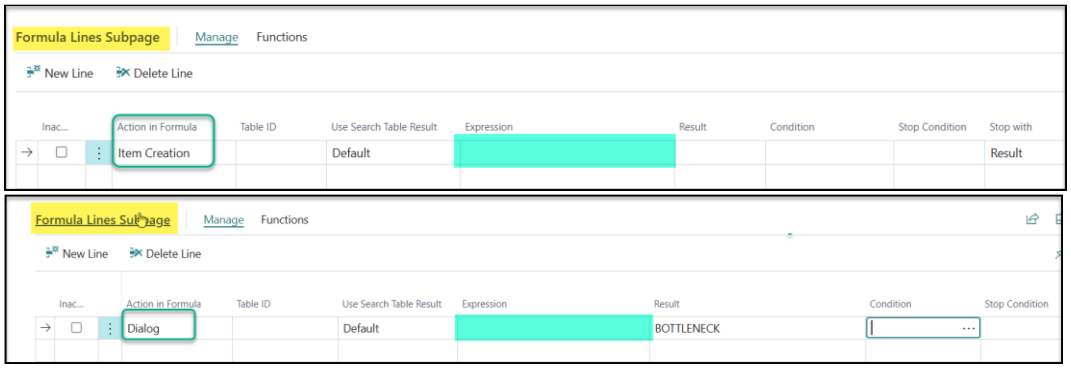


[9] This function is also used for calculating nth roots by using the reciprocal version of n: 1/n.

Below we see an example of a Formula Line with **Action in Formula** *Text Replacement*. Note that the Text string is between "double quotes".



When the **Action in Formula** is *Item Creation* or Dialog, the **Expression** needs to stay empty as shown in the examples of Formula Lines below.



Result

In connection with the **Action in Formula** *Calculation, Search, Cross Calculation* and *Text Replacement*, the Field **Result** (**Type** *Code*) is filled with the name of the Variable in which the Result of the Formula execution must be placed. After all (Active) Formula Lines are executed, the Value in **Result** will be posted to the Table Field the Variable is related to. This could be a **VarDim Type** or a **Global Variable**.

> **Note**
> This Field can only be populated with **1** Variable.

If **Action in Formula** is *Dialog* the **Dialog No.** that should be executed is entered.

If **Action in Formula** is *Item Creation* or *Blank* the Field **Result** should not be filled (blank).

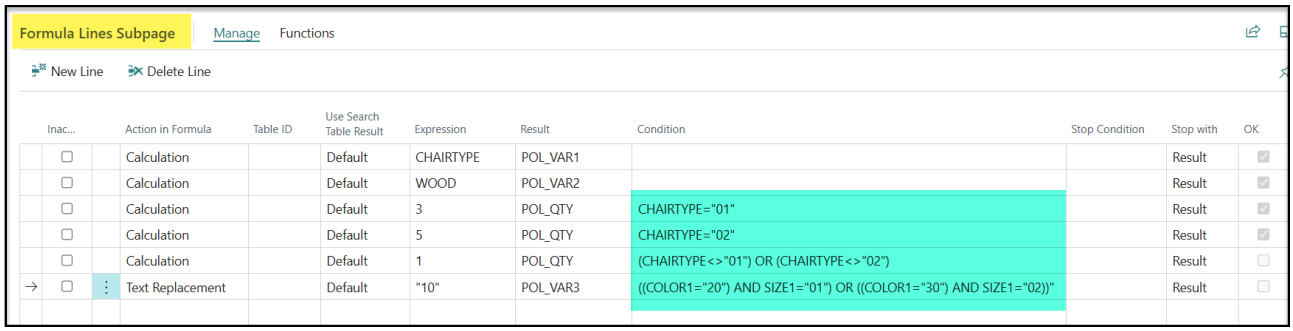
Condition

The (*Text*) Field **Condition** can be filled with an expression, which can be either *TRUE* or *FALSE* when executing the Formula Line. If the evaluation of the expression in **Condition** is *FALSE*, the Formula Line with this condition will not be executed.

An expression can consist of several elements: Variables combining with Mathematical Symbols into an equation.

Symbol	Description		Symbol	Description
=	Equal to		<>	Unequal to
>	Greater than		<	Less than
>=	Greater than or Equal to		<=	Less than or Equal to
(	Left Parenthesis		)	Right Parenthesis
AND	And		OR	Or

Multiple elements can be used. They are separated by operators *AND* and/or *OR* and encircled by parentheses (). Elements separated by *AND* needed to be both *TRUE* for continuation of the Formula Line execution. If at least one of the elements is *TRUE*, then *OR* should separate the elements. It is possible to construct the **Condition** Text in such a way that multiple separators can be used.
With the correct use of the parentheses () the appropriate elements are combined. If required, you could assemble elements within elements.



Inac...	Action in Formula	Table ID	Use Search Table Result	Expression	Result	Condition	Stop Condition	Stop with	OK
☐	Calculation		Default	CHAIRTYPE	POL_VAR1			Result	☑
☐	Calculation		Default	WOOD	POL_VAR2			Result	☑
☐	Calculation		Default	3	POL_QTY	CHAIRTYPE="01"		Result	☑
☐	Calculation		Default	5	POL_QTY	CHAIRTYPE="02"		Result	☑
☐	Calculation		Default	1	POL_QTY	(CHAIRTYPE<>"01") OR (CHAIRTYPE<>"02")		Result	☐
→ ☐	Text Replacement		Default	"10"	POL_VAR3	((COLOR1="20") AND SIZE1="01") OR ((COLOR1="30") AND SIZE1="02")"		Result	☐

In the screenshot above we see two Formula Line **Conditions** with one element and one with two elements. The last Formula Line is an example of assembled elements. Apart from the operators the elements consist of Variables. Note that it is also possible to work with **VarDim Type Groupings**. These are represented by the VarDim Extensions *[G0]* – *[G09]*[10].

The maximum length of the Text Field **Condition** is *250* characters.

Note
Due to the limited length of the Text Field, but also since too many elements in the **Condition** are bad for the performance, it is better to work with Search Tables instead.

Stop Condition + Stop with

In the previous paragraph **Conditions** are discussed that are only pointing to the **Formula Line** in which they are entered. However, it is also possible to set Conditions that are generic for the whole **Formula**. Where we talk in the previous paragraph about *Formula Line Conditions*, these Conditions can be considered as *Generic Formula Conditions*.

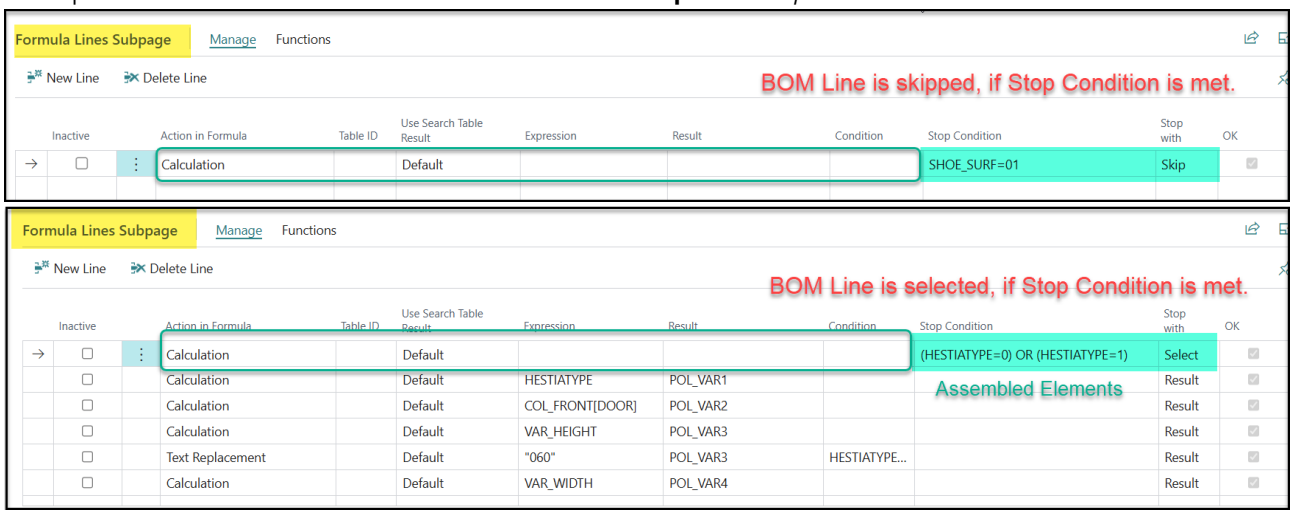
In the (text) Field **Stop Condition** we can enter an Expression, which can be either *TRUE* or *FALSE* when executing the Formula Line. If the **Stop Condition** is met (Expression = *TRUE*), the Formula will continue according to the selected option in the **Stop with** Field.
The options in **Stop with** and their effects on the Formula are:

Stop with	Effect on Formula
Result	The Formula is interrupted when the **Stop Condition** is met. The Variables calculated in Formula Lines executed earlier, are posted to the related Table Fields.
Select	The whole Formula will only be executed if the **Stop Condition** is met. If not, the Formula is interrupted as if it has not started at all. No values are posted.
Skip	The whole Formula will be interrupted if the **Stop Condition** is met. No values are posted. If not the execution of the Formula continues.
Go To	If the **Stop Condition** is met, the next Formula Line to be executed is the Line defined in the Field **Go To.** This will be discussed in the next paragraph.

[10] The **Extensions** *[G0]*, *[G1]* … *[G09]* have in fact related **Functions**. Note that there are ten more **VarDim Groupings** possible: *[G10]* – *[G19]*. However, these 10 don't have Functions related to it, so they can't be used in Formulas.
For more about TRIMIT Functions, go to *Appendix B: Functions*.

The default Value of the **Stop with** Field is *Result*. The expression in the **Stop Condition** can be constructed in the same way as the Line **Condition** in the previous paragraph: assembled from Variables and Mathematical Symbols. The maximum length of the text in the Field **Stop Condition** can contain up to *250* characters.

If a Formula is attached to a **BOM Component Line**, A Formula Line with **Stop with** *Select* establishes the BOM Component Line on the **Production Order Line**, when the **Stop Condition** is met. If the **Stop with** is *Skip*, the BOM Component Line will not be on the Production Order Lines. Below we see two examples of Generic Formula Conditions: one with **Stop with** *Skip* and the other with *Select*.

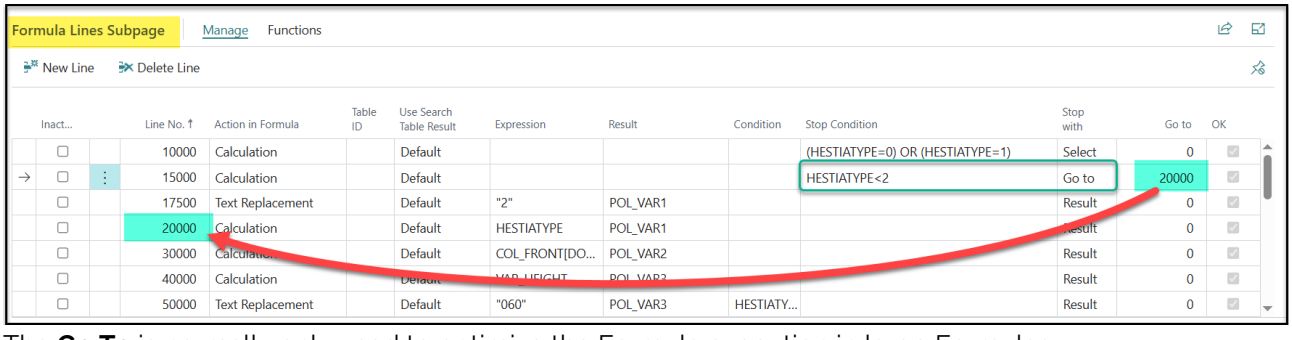


Note
Formula Lines with a *Generic Formula Condition* will only have the Fields **Stop Condition**, **Stop with** and **Go To** filled. Other Fields need to be empty except the Fields **Action in Formula** and **Use Search Table Result**, which keep their default Values. Of course, this Formula Line has its own **Line No.**

Go To

When the **Stop with** option is *Go To* and the **Stop Condition** is met, the system jumps to the **Formula Line** with **Line No.** that corresponds with the integer entered in the decimal Field **Go To**. From there the executing of the Formula continues *Top->Down* until the next **Stop Condition**.

In the example below we can see that if the **Stop Condition** in Formula Line *15000* is met, the Formula executing makes a jump to Formula Line *20000*. This way Formula Line *17500* is jumped over and therefore not executed!



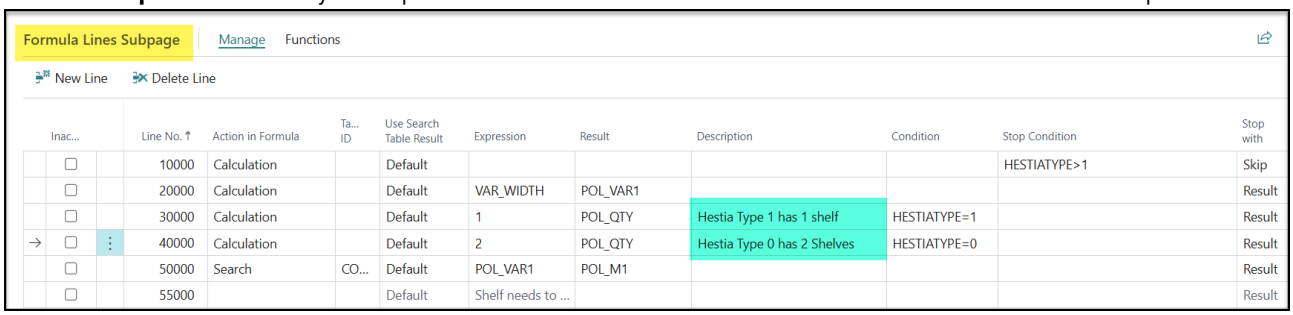
The **Go To** is normally only used to optimize the Formula execution in large Formulas.
By default, this Field is not shown on the page and can be made visible by ***Personalization***.

Note
With **Go To** you can only make a jump *forwards*. Jumping *backwards* has big risks to getting involved in an **endless loop** and is therefore not possible.

Description & Comment

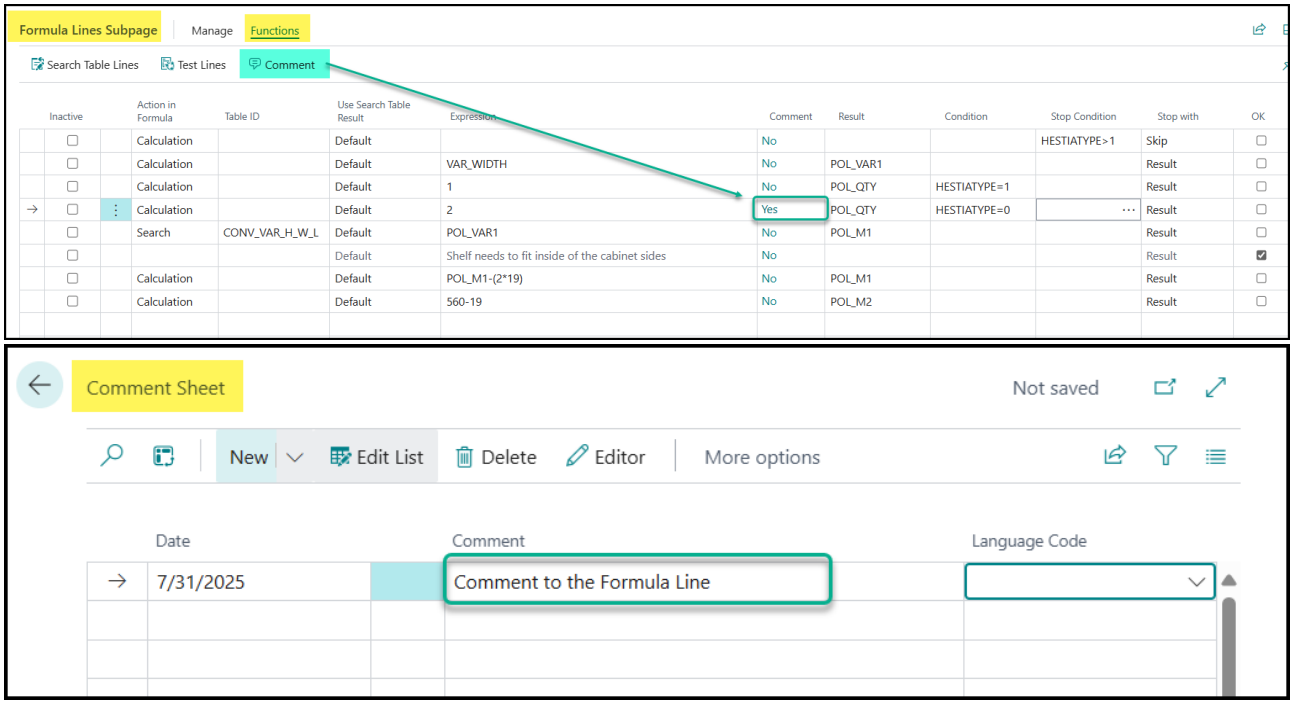
It is possible to have some information that clarifies elements in the Formula. Above is discussed that you can use the Field **Expression** for *Free Text* in case the **Action in Formula** is *Blank*. This Text is generic for the Formula. It is up to interpretation which part of the Formula is clarified.

However, it is also possible to provide information that is related to a specific **Formula Line**. One of the possibilities is entering free text in the *text* Field **Description** on a specific Formula Line.
The **Description** enables you to provide more information about this Formula Line (see example below).



Inac...	Line No.	Action in Formula	Ta... ID	Use Search Table Result	Expression	Result	Description	Condition	Stop Condition	Stop with
	10000	Calculation		Default					HESTIATYPE>1	Skip
	20000	Calculation		Default	VAR_WIDTH	POL_VAR1				Result
	30000	Calculation		Default	1	POL_QTY	Hestia Type 1 has 1 shelf	HESTIATYPE=1		Result
→	40000	Calculation		Default	2	POL_QTY	Hestia Type 0 has 2 Shelves	HESTIATYPE=0		Result
	50000	Search	CO...	Default	POL_VAR1	POL_M1				Result
	55000			Default	Shelf needs to ...					Result

If the Field **Description** does not provide enough space for the free text you want to enter, there is always the possibility to add a **Comment** to a Formula Line via ***Functions, Comment***. The ***Comment Sheet*** page can be opened and is in accordance with other Comment pages. If some text is entered in the Comment Sheet, the Field **Comment** on the Formula Line will be set to *Yes*.



Inactive	Action in Formula	Table ID	Use Search Table Result	Expression	Comment	Result	Condition	Stop Condition	Stop with	OK
	Calculation		Default		No			HESTIATYPE>1	Skip	
	Calculation		Default	VAR_WIDTH	No	POL_VAR1			Result	
	Calculation		Default	1	No	POL_QTY	HESTIATYPE=1		Result	
→	Calculation		Default	2	Yes	POL_QTY	HESTIATYPE=0		Result	
	Search	CONV_VAR_H_W_L	Default	POL_VAR1	No	POL_M1			Result	
			Default	Shelf needs to fit inside of the cabinet sides	No				Result	☑
	Calculation		Default	POL_M1-(2*19)	No	POL_M1			Result	
	Calculation		Default	560-19	No	POL_M2			Result	

The Fields **Description** and **Comment** are not shown on the Formula Lines by default.
You can add them via ***Personalize***

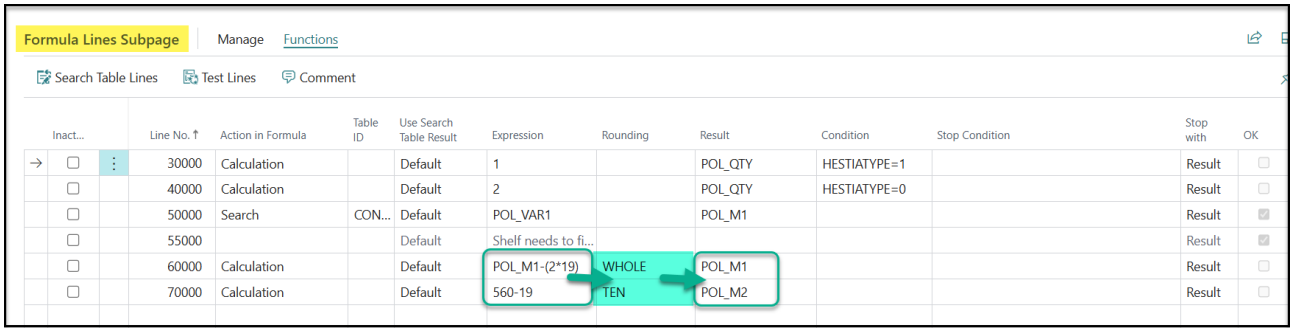
Filter VarDim Type

The Field **Filter VarDim Type** (not in the page by Default) can only be used in combination with Formulas that are attached to a **VarDim Order** or a **VarDim Item**. The Filter is used if an update of several VarDim Types in a **VarDim Order** or a **VarDim Item** is wanted. In **Result** an asterisk (*) is entered to show that the Filter should be used.

Note
The **Filter VarDim Type** will be removed in future versions and should not be used in new implementations!

Rounding

In the Field **Rounding**, a **Rounding Method** for the rounding of the Field **Result** can be entered. This means that a Value is determined (via extraction, calculation, searched for, etc.), then rounded and then placed into the Variable in the **Result** Field. The setup of the **Rounding Method** is standard BC and is not discussed in this document.



This Field is not by default shown on the page but can be added via ***Personalize***.

Compiling Formula Lines (F9)

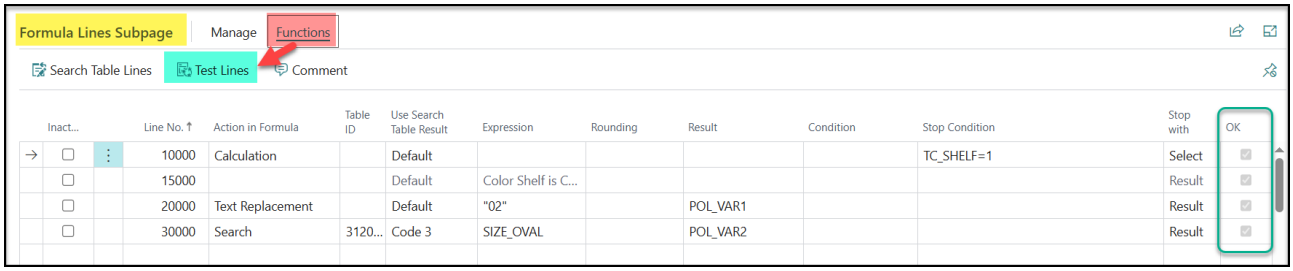
Entering the Data in the **Formula Line** Fields is a very precise job. It is easy to make a typing error. After entering a Formula Line, it is possible to compile this Formula Line to see whether the syntax is correct. It checks for instance:

- Whether the appropriate Formula Line Fields are populated.
- Do the Variables exist?
- Are Mathematical Symbols and Operators used correctly?
- Etc.

Compiling a Formula Line can be done in two ways:

1. Select the Formula Line and press Function Key **F9**.
2. Select the Formula Line, and click ***Functions, Test Lines***.

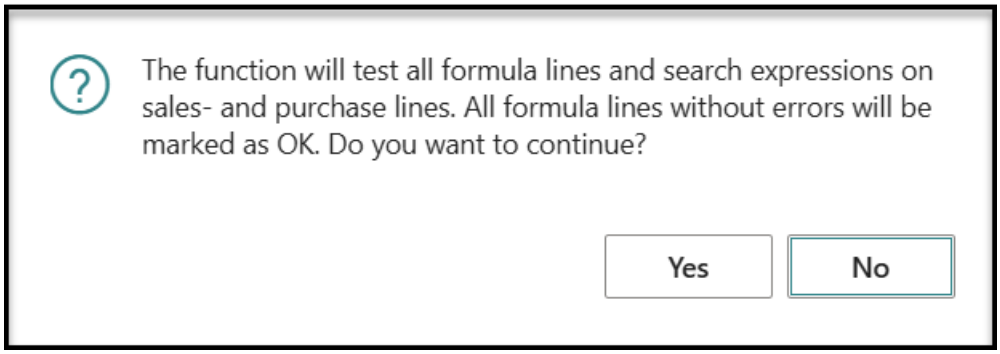
This must be done Line by Line!



If the compiling is a success, the Boolean Field **OK** will be checked. If not, you get an error message. The moment you change something in the Formula Line; the **OK** Field will be unchecked.
This Formula Line needs to be tested again.

Note
The **Test Line** function (**F9**) only investigates whether the syntax is correct. It does not check whether the Formula works or not.
You can work without testing Formula Lines. However, errors may occur while a Formula is executed. The search for the actual flaw line is much harder this way compared to the Test Line procedure when creating the Formula.

Though most of the time the Formula Lines are tested one by one, for specific circumstances it is possible to do the test as a batch. For this we need to run the report ***Test Formula Line (6036805)***. Note that only Formula Lines are tested, which are used in the **Sales Price Table** and **Purchase Price Table**. Runing this report gives you the following message.



Note
Running this report only mark Lines as *OK*, if they are correct. The ones that are not correct will not be marked as *OK*. If you want to see the list of unmarked Lines, you will have to filter them out in the *Formula Lines*.

Formula Expression AssistEdit (Alt+Arrow Down)

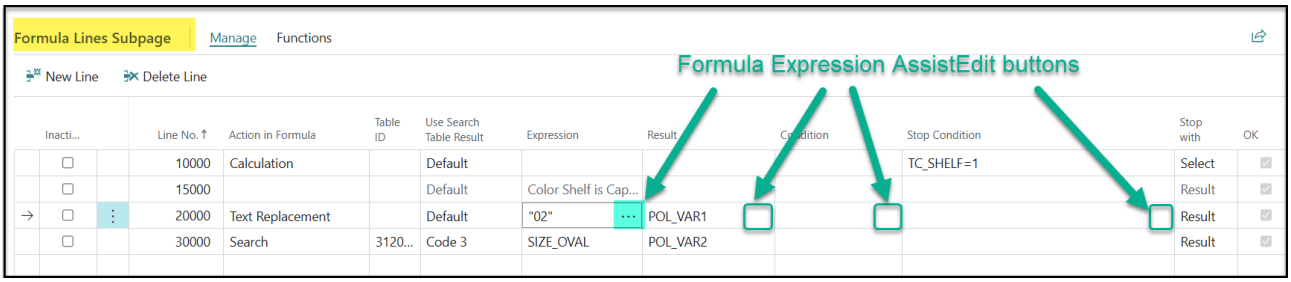
Introduction

As mentioned in the previous paragraph, the entry of the Data in Formula Lines is very precise. For this reason, TRIMIT has a tool that can help you with entering the right Variables and/or Operators. This is called the **Formula Expression AssistEdit**.

The **Formula Expression AssistEdit** can be called on the Formula Line in the following Fields:

- **Expression**
- **Result**
- **Condition**
- **Stop Condition**

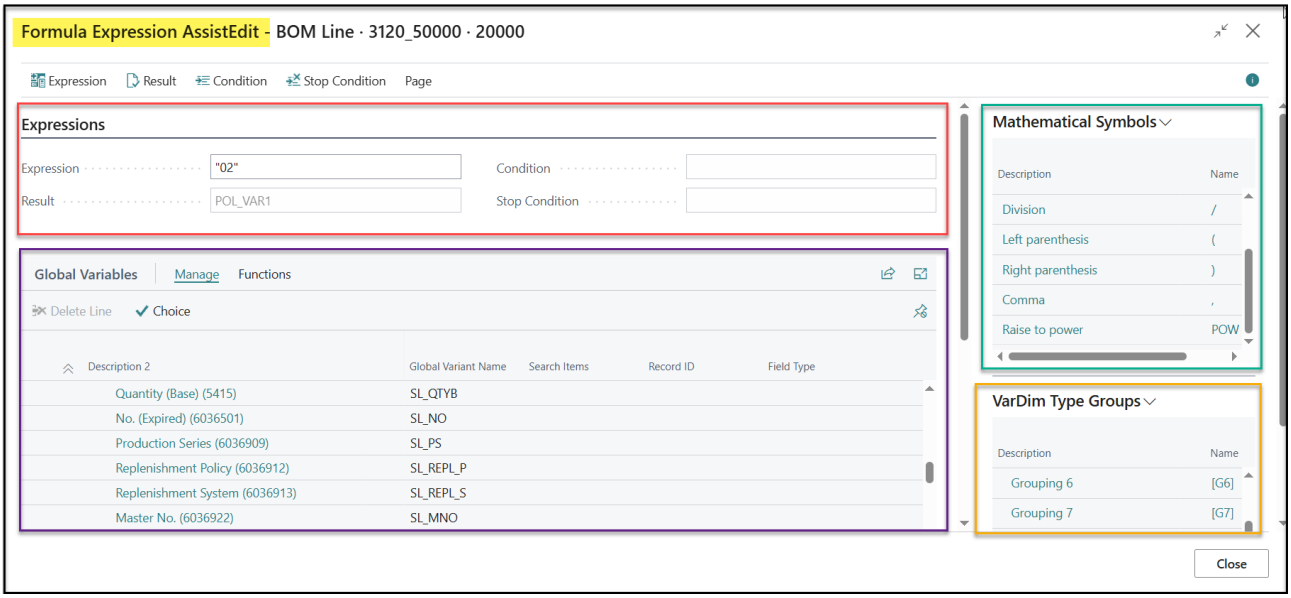
You can open this tool by clicking on the **AssistEdit** button (*"..."*) in these Fields.



The ***Formula Expression Assist Edit*** Page consists of five elements:

- FastTab ***Expressions***
- Subpage ***Global Variables***
- FactBox ***Mathematica Symbols***
- FactBox ***VarDim Type Groups***.
- FastTab ***Parameters***

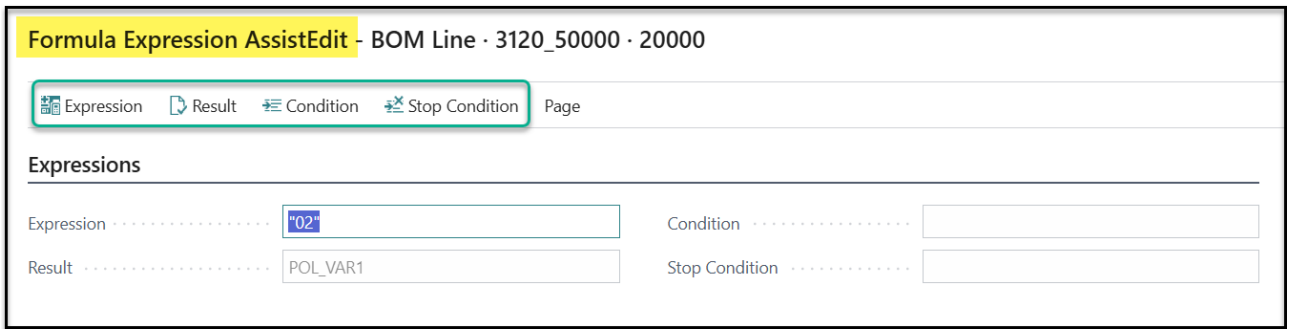
Four of them are shown below. The fifth one is below the ***Global Variables*** Subpage.



These elements are discussed in the paragraph sections below.

Expressions

In the FastTab ***Expressions*** you can see the four Fields on which the **Formula Expression AssistEdit** can be called. However, only the one from which this page is actually called, we can enter Data. The others are greyed-out. By clicking ***Expression***, ***Result***, ***Condition*** **or** ***Stop Condition***, you can switch between the Fields of the Formula Line without leaving the **Formula Expression AssistEdit** page and reopening it from another Field.



Which Data fits in the Fields in FastTab ***Expressions*** depends on whether the Field is an **Expression**, a **Result**, a **Condition,** or a **Stop Condition**. This is described in the paragraphs above concerning these Fields in this chapter.

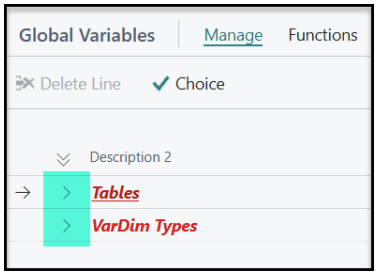
It is possible to enter the Data manually. However, you can also compose it using the other elements of the **Formula Expression AssistEdit** page. These elements are discussed in the next sections.

Subpage Global Variables

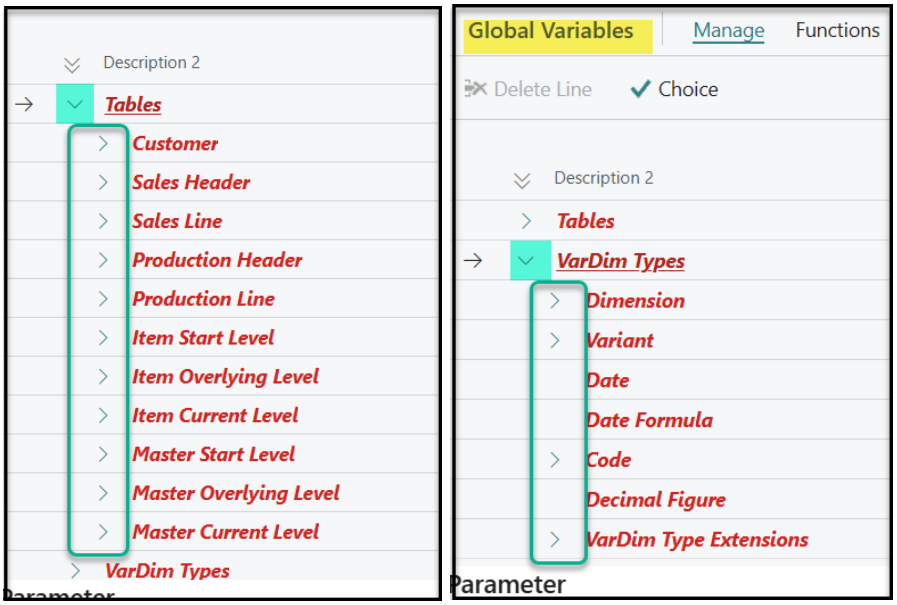
In this section we can see all **Global Variables**, **VarDim Types**, **Functions** and **VarDim Extensions** that can be used on the **Formula Line** the **Formula Expression AssistEdit** is called from. Which Variables can be selected depends on **Formula Type** of the Formula this Formula Line belongs to.

The list of Variables can be exceedingly long. Therefore, we group them based on the Tables that can be used in the actual situation (depending on the **Formula Type**). With the help of the arrow, we can expand or collapse the grouping.

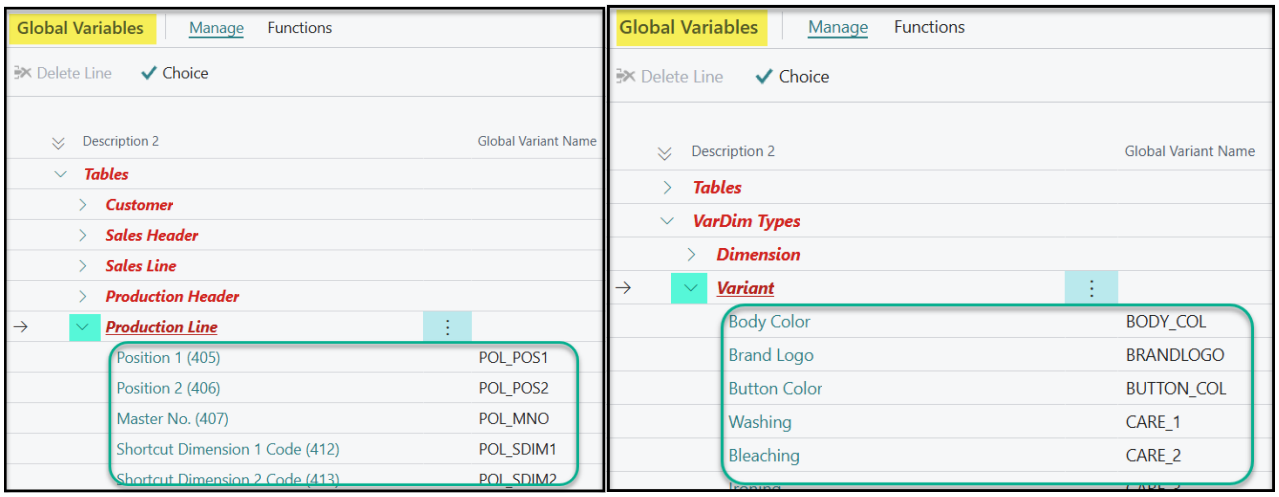
Highest Level:



Middle Level:



A part of the Lowest level:

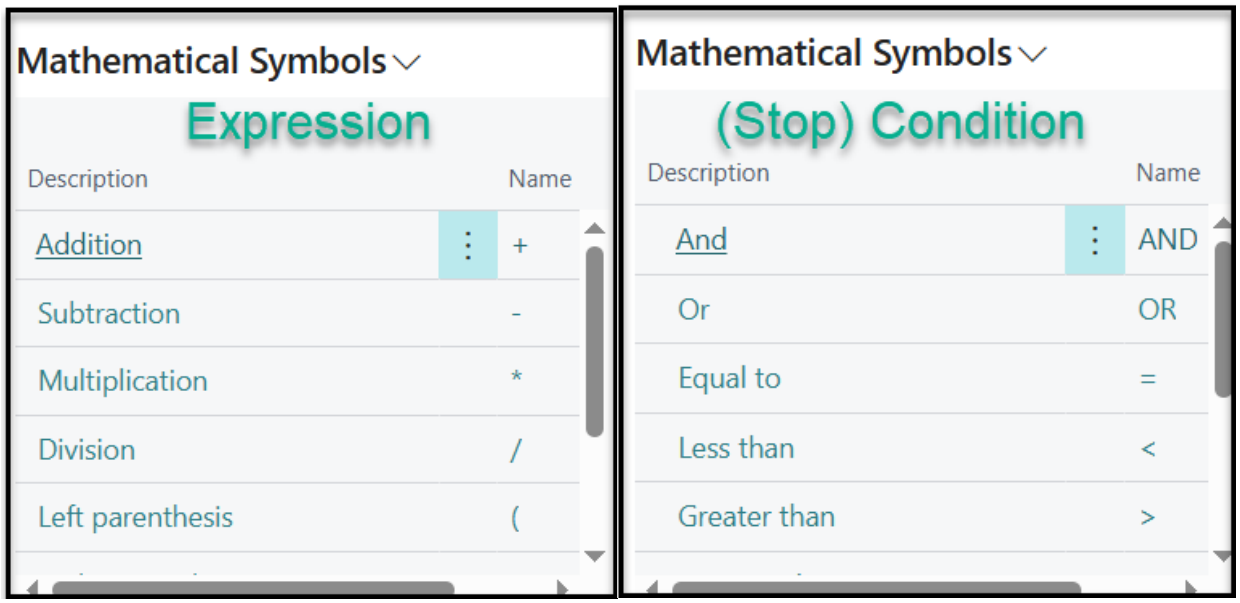


Note
The **Global Variables** are only shown in the Tables if they are activated (see chapter *Variables*)

If you select a **Global Variable** to be entered in the activated Formula Line Field, you need to push ✓ Choice to have the Variable placed in this Field. You can compose the contents in these Fields by making more choices. (This cannot be done for the Field **Result**).

Mathematical Symbols

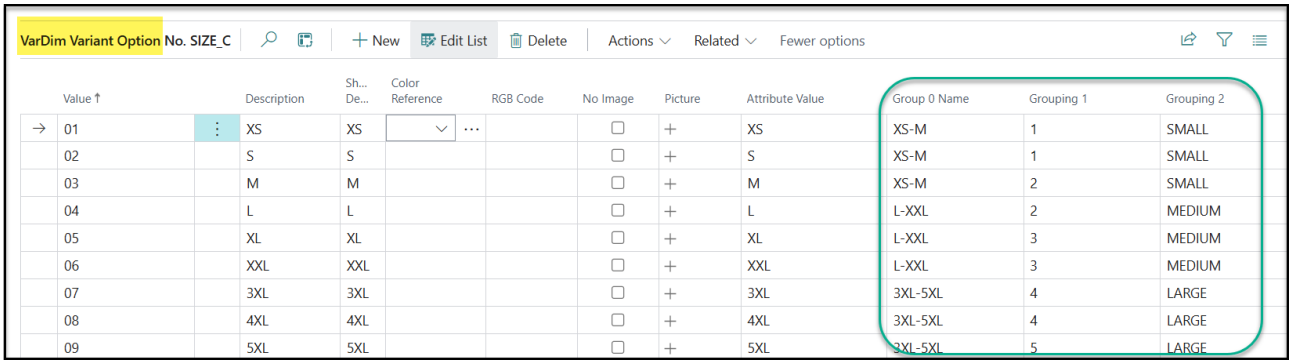
In the FactBox ***Mathematical Symbols*** you can choose which Operator you want to use. Which Symbols you can choose depends on the activated Field in FastTab ***Expressions***. The Mathematical Symbols list for the Field **Expression** is different than those for the **Condition** and **Stop Condition**.
In the Field **Result** you can only enter a Variable, so there are no Operators at all.



The Mathematical Symbols can be added in the Field activated in FastTab ***Expressions*** by clicking once on it. This way you can compose elements consisting of Variables and Operators.

VarDim Type Groups

If a **VarDim Type** is of **Type** *Variant*, it will have a Table with related **Options**. This Table may contain a large number of Options. It could be especially useful to create Groups of Options. An Option can be a part of more than one Group. Up to 20 different Groups can be created. These are represented in the columns in the **VarDim Variant Option** Table: *Grouping 0* till *Grouping 19*.



The first *10* Groups (*Grouping 0 – Grouping 9*) have a connection with a TRIMIT Function: *[G0] – [G9]*[11].
These VarDim Type Groups can be used in the Formula Lines as an extension to a VarDim Type.
Below we see an example of the Format:

SIZE_C[G1]

[11] More about TRIMIT Functions in *Appendix B: Functions*.

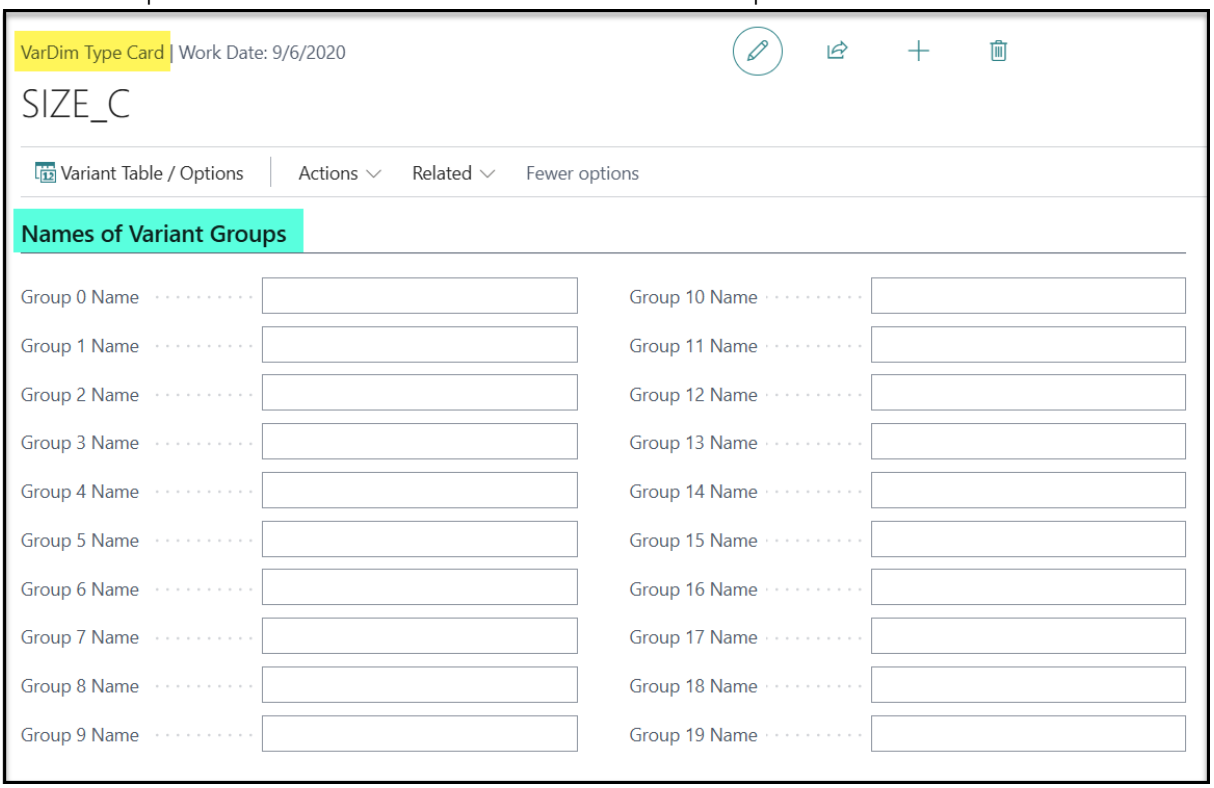
This means that the system does not look for the **Value** of **SIZE_C** but takes the Value in the column **Grouping 1** instead. Let us have a case:

Let us assume that we have a Chair that can be upholstered by Fabrics coming in different Patterns. Based on the Pattern I can have a different Consumption. This has to do with the directions you can upholster the Chair.
For instance, if you have a striped Pattern, you may want to have that the stripes of all Chairs are vertical. That means that you only have one direction. A checkered Pattern has two directions, but a plain Fabric can go in any direction. You can imagine that the more directions a Pattern has the more economic you can cut the Fabric.
However, if you have a catalogue with a lot of Fabric variants, you do not want to determine what the Consumption would be for every individual Fabric. It is better to use one of the Groupings based on the Pattern. Now you only must maintain a Table that organizes the Consumption per pattern Group. For every new Fabric Variant, you only apply the correct Pattern Group Value, and you are done.
The Formulas do not need to be adjusted anymore for every new Variant coming in, because the Pattern of it already exists.[12]

Note
The **Groupings 10 – 19** do not have a related **Function**. Therefore, they cannot be used in the TRIMIT **Formulas**. However, they can be used in combination with **Search Expression** in the **BOM Component Line** or in the Sales-/Purchase Prices.

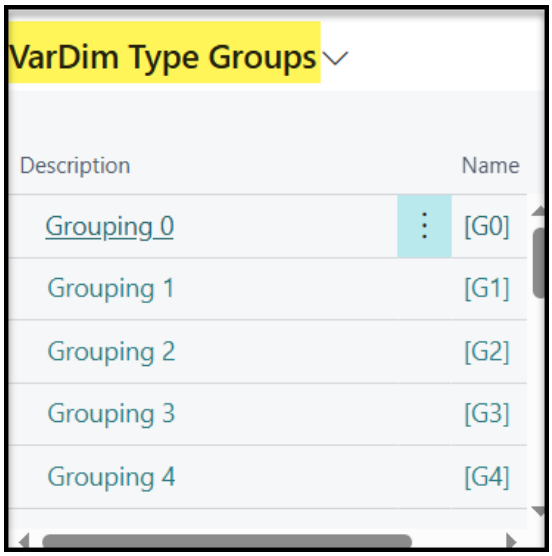
It is possible to change the **Caption** of **Grouping 0-19**. The name of the Grouping depends on the VarDim Type. Therefore, the Captions need to be set up on the **VarDim Type Card**.
There is a FastTab ***Names of Variant Groups*** in which you can enter the desired Name.

In the example above that could be for instance Pattern Group.



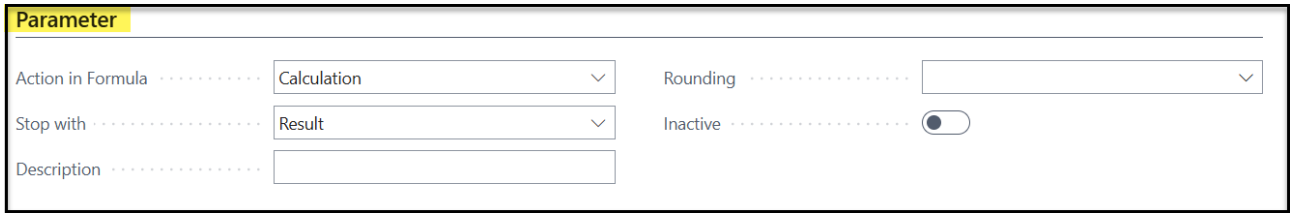
[12] Note that this functionality often goes together with the TRIMIT Search Table feature, since we must set up a Table with Pattern Groups with their Consumption.

These Groupings can also be selected in the **Formula Expression AssistEdit** via the FactBox ***VarDim Type Groups***. The **VarDim Type Group** can be added in the Field activated in FastTab ***Expressions*** by clicking once on it. This way you can compose elements that include VarDim Type Group extensions.



Parameters

The **Formula Expression AssistEdit** may help you with populating Fields within the **Formula Line**.
In the FastTab ***Expressions*** you can compose the elements in the Fields **Expression**, **Result**, **Condition** and **Stop Condition**. However, you can also enter Values of other Fields in the Formula Line, such as **Inactive**, **Action in Formula**, **Stop With**, **Description** and **Rounding**.
This can be done in FastTab ***Parameters***.



Note
For different reasons, the Fields **Table ID**, **Use Search Table Result**, **Comment**, **Go To** and **Line No.** are Fields that are not maintained in the **Formula Expression AssistEdit**.

Values of Boolean and Option Fields

Like any other Fields, Booleans and Option Fields can also be used in Formulas. However, because of the nature of these Fields, the Values of these Fields are restricted.

The Boolean can have Values *FALSE* or *TRUE*. In TRIMIT Formulas that translates to *0* or *1*.
See Table below:

Boolean Value	Boolean on Page Type Card	Boolean on Page Type Lines	Boolean Value in Formulas
FALSE	○	☐	0
TRUE	●	☑	1

Something comparable is the case with **Option** Fields. The choices belonging to an **Option** Field are represented in both Code as in TRIMIT Formulas by integers: the first Option gets Value *0*, the second Option gets Value *1*, the third gets Value *2*, and so on.

For instance, when we look at the Option Field **Replenishment System**, we have five possible choices, which we can see in the Table below with its corresponding integer.

Choice in Option Field Replenishment System	Integer used as Value in Formulas
Purchase	0
Prod. Order	1
Transfer	2
Assembly	3
'blank'	4

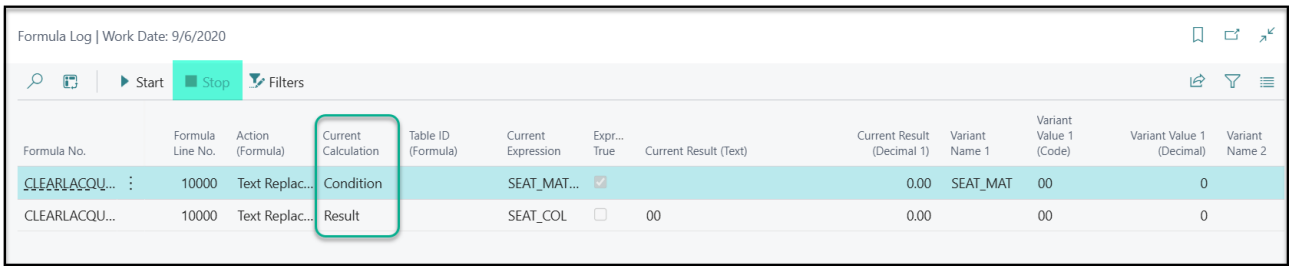
Variables representing *Boolean* and *Option* Fields can be used in the Formula Line Fields **Expression**, **Result**, **Condition** and **Stop Condition**. In the last two Fields they are a part of a test element.
For instance: GV = 0, GV = 1, etc. where GV represents the **Global Variable** pointed at the Boolean or the Option Field.

Logging Formula Lines

When the Formula is executed, it is possible to have the Formula Line and the Result that is created for each Line logged. This makes troubleshooting errors in the Formulas easier.

At any time, you can start the Formula logging in the ***Formula Log*** page by clicking on ***Start***. Then you can do an activity that triggers the execution of Formulas.

Afterwards you can go back to the ***Formula Log*** and click ***Stop.*** By clicking ***Stop,*** the page shows the logged Lines and the Results. If an error has occurred in the Formula execution, the logging stops at the error.



A **Formula Line** can have more Log Lines. For instance: on a Formula Line the **Condition** is checked first. This generates a Log Line. If the Condition is met, then the **Calculation** is done, which creates a second Log Line and finally the Variable in the **Result** column gets a Value, which creates a third Log Line.
In the example above you see two Log Lines created based on **Formula Action** *Text Replacement*.
The Type of calculation that has taken place is shown on the Field **Current Calculation**.

If the **Current Calculation** uses a **Search Table**, you can see which Table it is in the Column **Table ID (Formula)**. In the Column **Current Expression,** you can find the Input **Expression**.
In the other Columns you can see the outcome of this Current Calculation. This could be a *Text*, a *VarDim Variant,* or a *Decimal* figure.
In case of a **(Stop) Condition**, you can see in the Field **Expression True**, if the Condition is met.

If the logging is only wanted in connection with a certain **Formula Type** or/and a certain **Formula Table Header** record, this can be done by entering those in the ***Filters***. You can also choose to log on to Formulas on certain BOMs by filtering on the Fields **BOM Type, BOM No.** and **BOM Line No**,

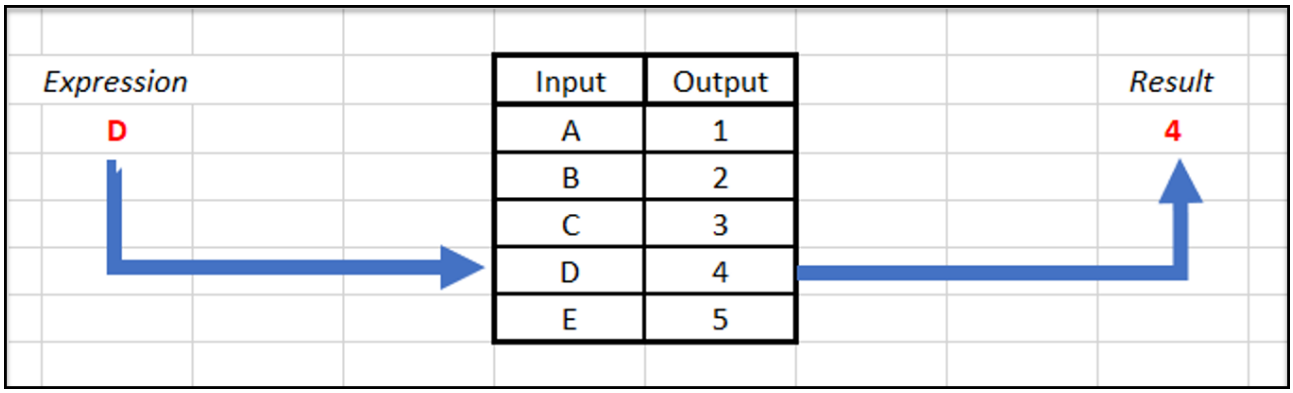
When you start a new Logging process, the existing Log Lines will be deleted.

Search Tables

Introduction

When using the **Action in Formula** *Search* a Search Table Number must be entered in the Formula Line in the Field **Table ID.** A Search Table consists of a **Search Table Header** that is created and maintained in Table ***Search Table Header*** and possibly a number of **Search Table Lines**. The Search Table Lines are created and maintained in Table ***Search Table Line***.

The principle of a Search Table is that you can link **Input** Values and **Output** Values with each other in case the Output cannot be calculated from the Input. This is schematically explained in the picture below. We see that the Input is a letter from the alphabet. The Output is the place of that letter in the alphabet. So, we make a Search Table in which we see the letters of the alphabet linked to their place.



When a Formula Line with this Search Table is executed, we can go in with a Value determined by the **Expression** in the Formula Line. In the **Search Table** we look for this letter and find the corresponding place. This Value will be put into the Field **Result** on the Formula Line.
So, in case we go in with a Value '*D,*' the result will be place '*4*'.

This means that a Search Table has two sides. An **Input** side and an **Output** side. Both could exist of more than one column. On the Input side these columns are combined with one Search Input. On the Output side we can have multiple Result columns. We must tell which of these columns has to be applied for this Formula Line. More of this later in this chapter.

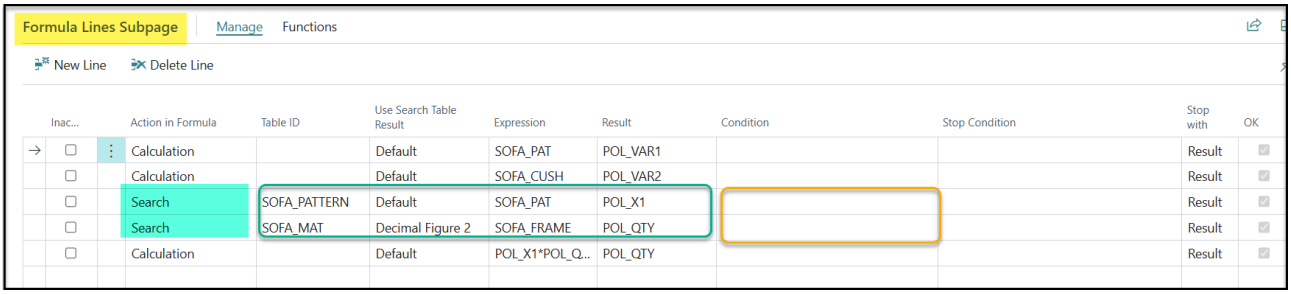
Note
Since the **Search Table** is an independent entity, you can use it on more than one **Formula Line**. It is possible that for each of these Lines another Output column is required. The Field **Use Search Table Result** determines this.

A search can be done either directly in a BC/TRIMIT Table, or in the Search Table Lines that can be created with relation to the Search Table Header. A search does not need to look for the exact Input Value. You can also look in ranges.

Use Search Table in Formula Line

Introduction Formula Line with Search Table

Let us start with having a closer look at the Formula Line with a Search Table.
In the screenshot below, you can see two Formula Lines with a Search Table. In these Formula Lines, there are several Fields that work together with the Search Table.



Formula Line Field	Value	Remark
Action In Formula	*Search*	
Table ID	*Table ID*	Name of the Search Table Header
Use Search Table Result	*Option points at Result column*	In case multiple Output columns
Expression	*Input Value*	Extracted and/or Calculated Values
Result	*Output Value*	Only one Variable
Condition	*Condition Expression (optional)*	Search Table calculation only when condition is met

The Formula Line with a Search Table process as follows:

1. First the **Condition** is tested.
   When the Condition is not met, the Formula Line is interrupted and skipped to the next Formula Line.
2. The **Expression** is determined. The **Expression** could be an extracted Value from a BC/TRIMIT Table (often a Variant), the result of a calculation or a combination of both. It could also be a string of Variables in case of a combination of Input Values.
3. With the **Expression** we go into the **Search Table** and find the appropriate (combination of) Value(s) on the Input side of the Search Table.
4. We take the corresponding Value from one of the Output columns, which is determined by the **Use Search Table Result** option.
5. Place that Output Value into the Variable in the Field **Result** on the Formula Line.

Action in Formula = 'Search'

In the case of working with a Search Table in the Formula Line, the chosen option in the Field **Action in Formula** must be *Search*. Only in that case can we enter a Search Table Number in the Field **Table ID**.

The Field **Table ID** only works with **Action in Formula** *Search* or *Cross Calculation*.
In all other situations an error message will be generated.

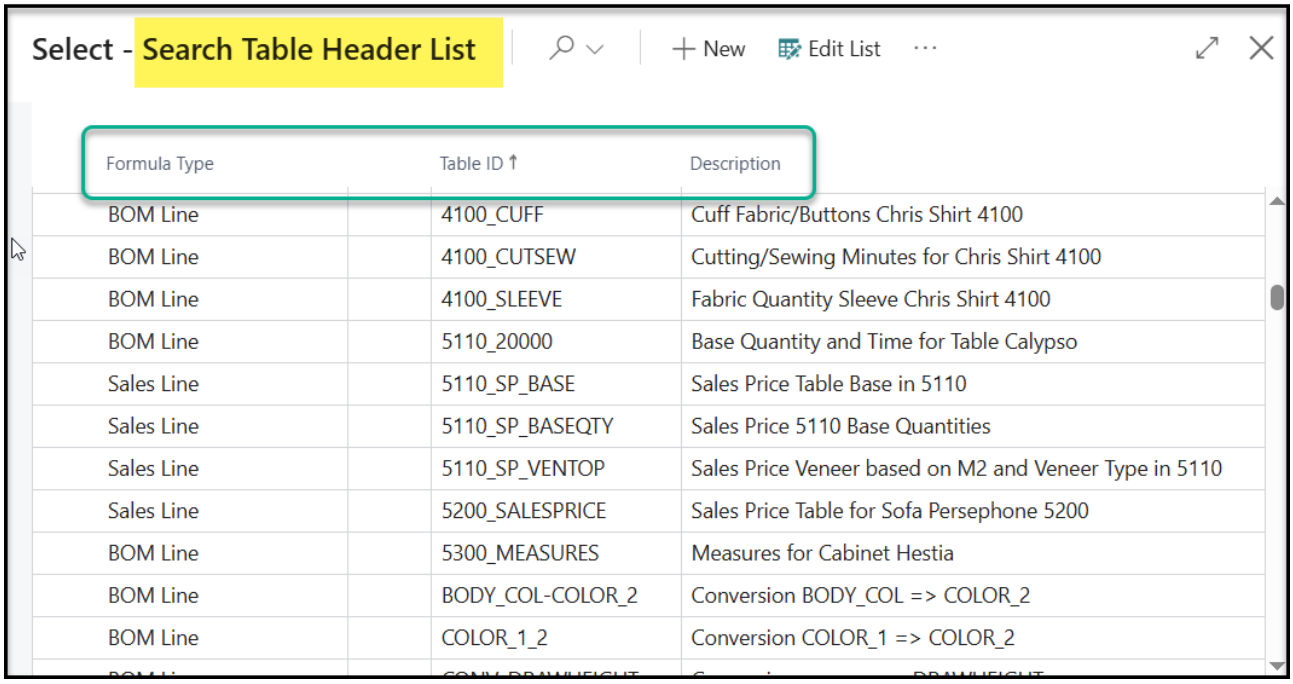
Table ID

In case **Action in Formula** is *Search*, you can enter the ID# of the Search Table Header in the Field **Table ID**. You can look for it in the **Search Table Header List**, if the Search Table already exists. If not, you can create one by clicking on ***+New***.

The Search Table Header List contains several Fields:

- Formula Type
- Table ID
- Description.

These and more Fields are discussed in another paragraph in this chapter when we discuss the *Search Table Setup*.

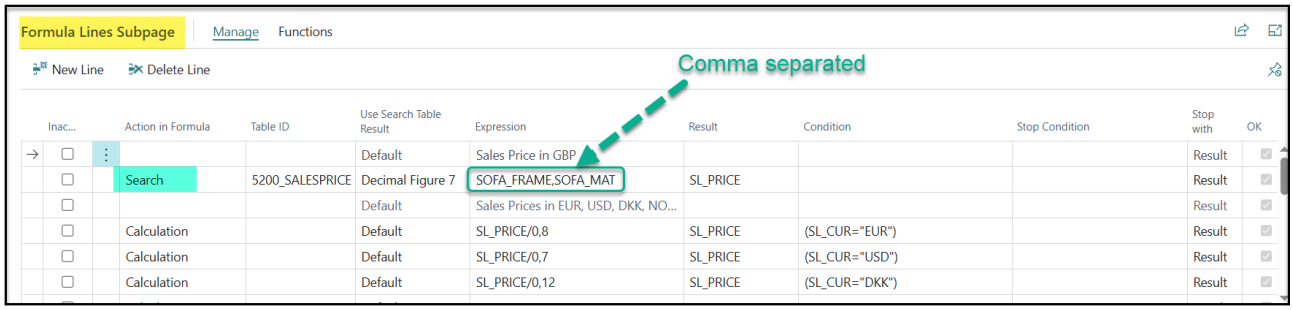


Formula Type	Table ID	Description
BOM Line	4100_CUFF	Cuff Fabric/Buttons Chris Shirt 4100
BOM Line	4100_CUTSEW	Cutting/Sewing Minutes for Chris Shirt 4100
BOM Line	4100_SLEEVE	Fabric Quantity Sleeve Chris Shirt 4100
BOM Line	5110_20000	Base Quantity and Time for Table Calypso
Sales Line	5110_SP_BASE	Sales Price Table Base in 5110
Sales Line	5110_SP_BASEQTY	Sales Price 5110 Base Quantities
Sales Line	5110_SP_VENTOP	Sales Price Veneer based on M2 and Veneer Type in 5110
Sales Line	5200_SALESPRICE	Sales Price Table for Sofa Persephone 5200
BOM Line	5300_MEASURES	Measures for Cabinet Hestia
BOM Line	BODY_COL-COLOR_2	Conversion BODY_COL => COLOR_2
BOM Line	COLOR_1_2	Conversion COLOR_1 => COLOR_2

Expression

In the Field **Expression**, we determine the Variables that are used as Input in the **Search Table**.
This Variable could be an extraction from any BC/TRIMIT Table, the result of a calculation or Function, or a Global Variable resulting from a calculation in an earlier Formula Line. It could also be a combination of Extraction, Calculation and Functions.

It is possible to have more than one Variable in the Expression.
In that case those Variables are **separated with a comma**.



For more information about the Field **Expression**, see chapter *Formula Lines Structure*.

Use Search Table Result

It is possible to have more than one Output columns in the **Search Table**. However, there could be only one applied on a particular **Formula Line**. In the option Field **Use Search Table Result**, we can indicate which of the Output columns will be placed into the Variable in the **Result** Field in the Formula Line.

In this Option Field we can choose between **20** columns with **Decimal** Values and **10** columns of Type **Code**. The default Value of this Field is *Default*. This one is chosen when there is only one Decimal column and/or one Code column.

Result

The output from a **Search Table** will be placed in the Field **Result**.

Condition

In the Field **Condition** we can place an Expression in the form of an Equation. If the Expression is tested and the outcome is *TRUE*, the Formula Line will be executed. If it is *FALSE*, the execution of the Formula Line will be interrupted. For more information about Conditions: see chapter *Formula Lines Structure*.

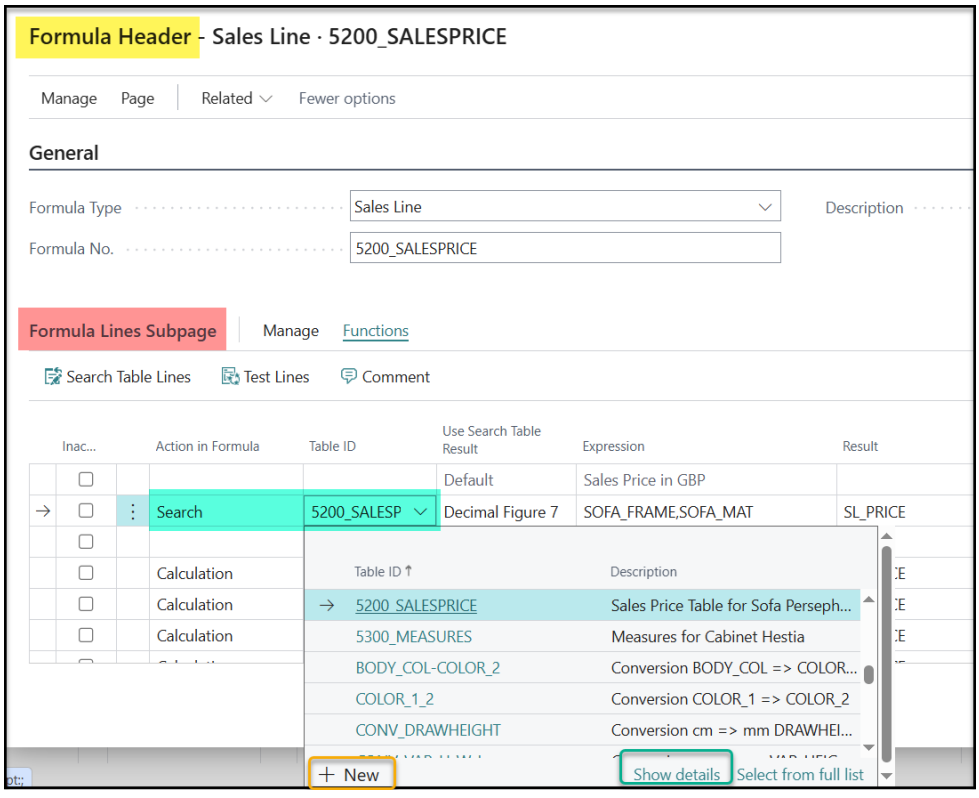
Search Table Setup

Introduction Setting up a Search Table

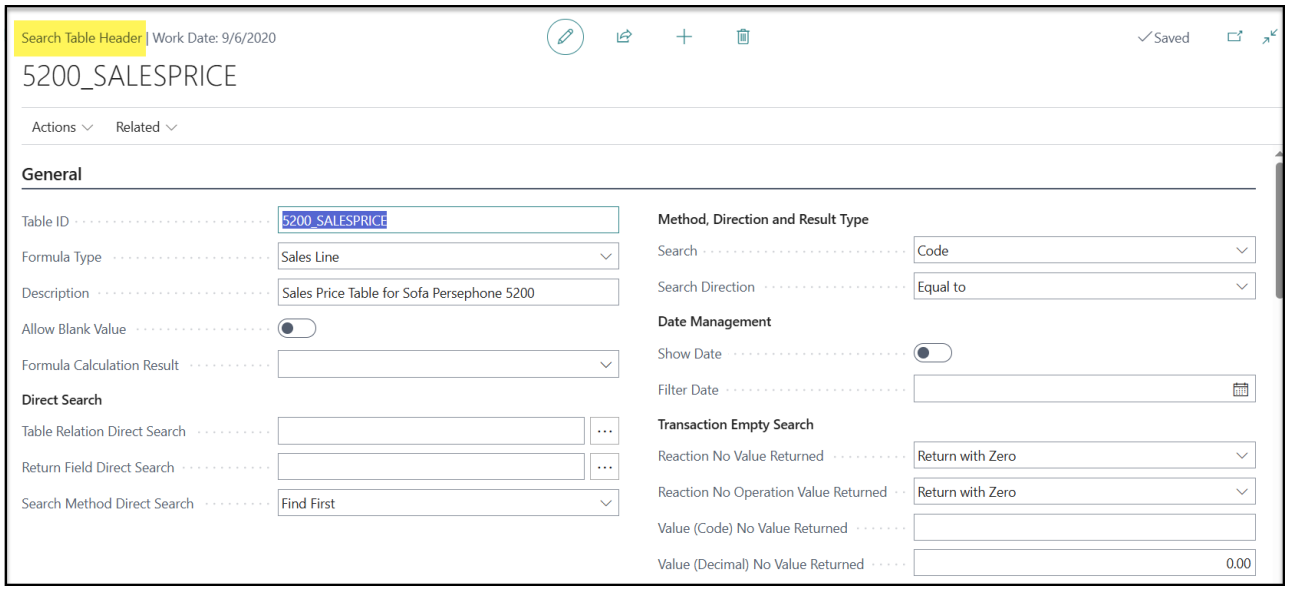
Before Search Table Data can be entered, the **Search Table** itself needs to be set up.
The Search Table Setup has four parts:

- *General Search Table settings*
- *Input Columns*
- *Output Columns*
- *Search Table Data.*

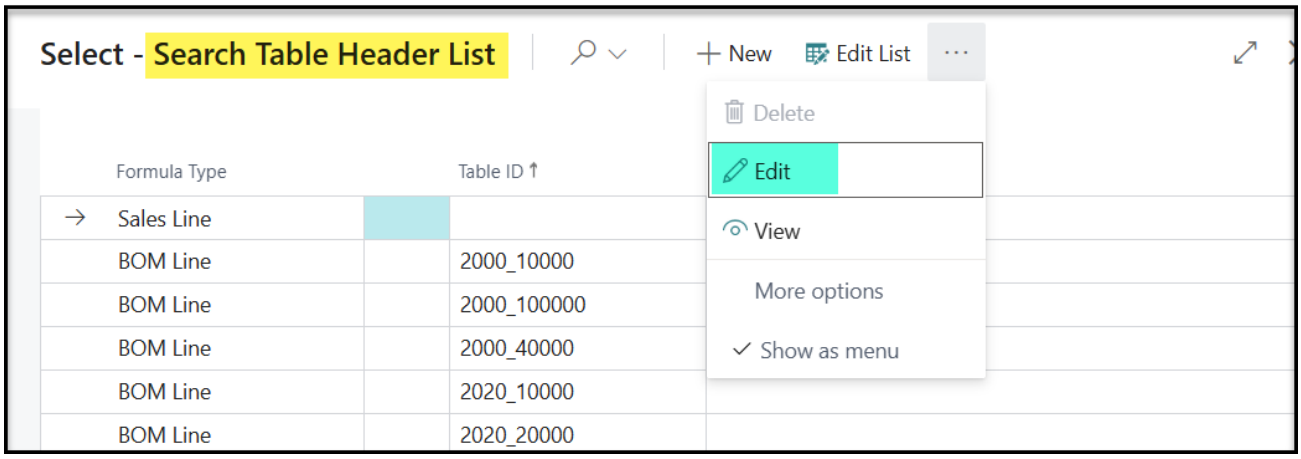
These parts will be discussed in the next paragraphs.



In a **Formula Line** we can see the setup of an existing Search Table. First you select the right Formula Line, go to the drop-down list, and click on ***Show details***. This is shown in the screenshot above. After clicking on ***Show details*** the **Search Table Header** shows up in which we can see the setup of the Search Table.



If we want to create a new Search Table, you go to the same drop-down list but click on ***+New***.
In this case the page **Search Table Header List** shows up with the selection on an empty Line. Then click on ***Edit*** to go to the Search Table Header to set up the Search Table.



If you are not in a Formula, you go to the **Search Table Header List** directly to set up a Search Table

Search Table Header List Fields

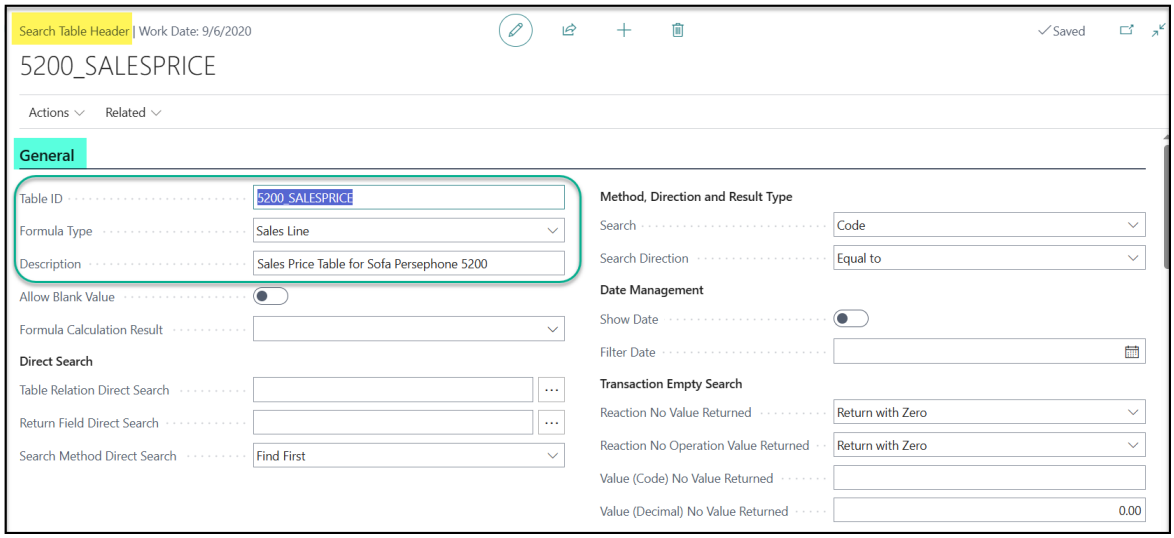
In FastTab ***General*** we can enter general information about the **Search Table**. These are the Fields that we also see in the **Search Table Header List**.

One of these Fields concerns the **Formula Type**. The option chosen in this Field must be identical to **Formula Type** in the Formula Header (or **Formula Type** *Global*)[13]. For instance, if you have a Search Table that is used in a Formula to have different **Unit Prices** for the different VarDim Type Values of a Master it is related to the **Formula Type** *Sales Line,* because it affects the **Unit Price** on the **Sales Line**.

[13] For more information regarding **Formula Type**, see paragraph *Formula Header*.

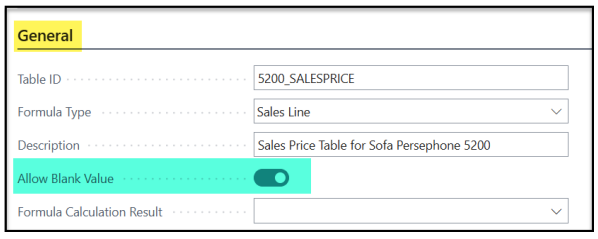
Another example could be a Search Table that is related to a Formula that is used in a BOM Component Line of a Master to combine the Colors of Materials with the Colors of a Finished Product. This Search Table would have **Formula Type** *BOM Line* because it is related to the **BOM Component Lines**.

Another Field is the **Table ID**. This defines the Name of the Search Table. The **Table ID** is the value inserted in the **Table ID** Field in the **Formula Lines**.
The last Field that we can also see in the **Search Table Header List** is **Description**. Here you can enter a free text that describes the purpose of the Search Table. This Field is optional.



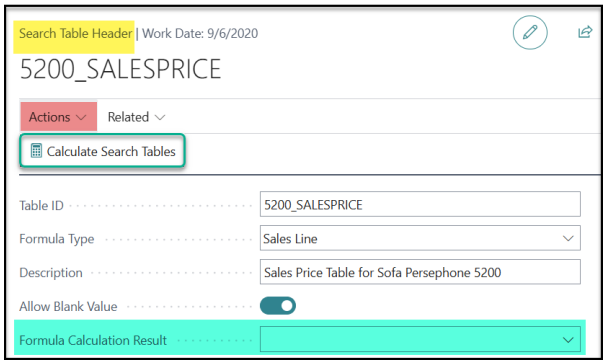
Allow Blank Value

It could be possible that one or more input Values in the search expression in the Field **Expression** (on the Formula Line) is *Blank*. If that is the case the Boolean **Allow Blank Value** should be set to *TRUE*.



Formula Calculation Result

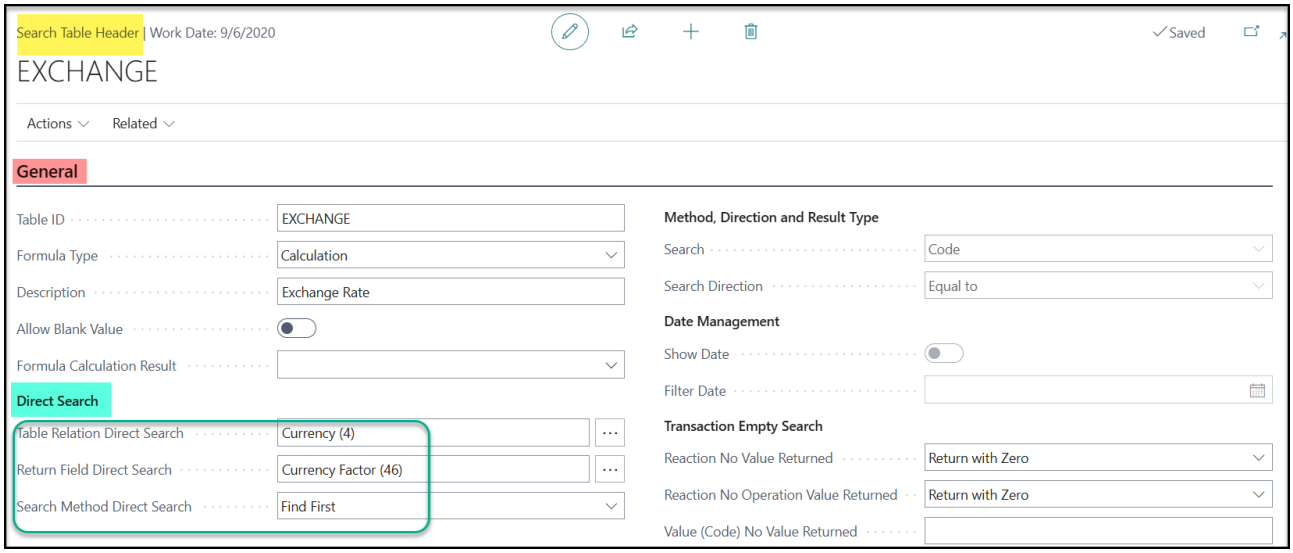
A Formula can be attached with **Formula Type** *Search Table* that can calculate Output columns in the **Search Table** based on Search Fields in the Input columns or other Output columns in the **Search Table**.



Execution of this Formula is only triggered by pressing ***Actions, Calculate Search Tables*** in the **Search Table Header**.

Direct Search

A Search Table could be created manually but could also be any Table in BC/TRIMIT. If the latter is the case, we need to enter the **Direct Search** Fields.



In the Field **Table Relation Direct Search,** we must enter the BC/TRIMIT Table in which we want to search. The Search Input is the (combination of) **primary key** Field(s). This means that the Variables in the **Expression** on the **Formula Line** must match the primary key Field(s). This means that the **Expression** contains comma-separated elements. The **maximum** is *5* elements, so only Tables can be used for **Direct Search** if they have *5* elements in the primary key or less. If one or more elements does not play a role in the search, you can replace these by having an *asterisk* "*" in the expression in the appropriate place in the element string. If no Value is entered into an element, the system considers it as '*Blank.*'

Note
With **Direct Search** we use an existing Table that already has Data. For that reason, **Search Table Lines** do not need to be set up. This means that the other Fields in the **Search Table Header** are ignored except (for obvious reasons) the Fields **Table ID**, **Formula Type**, **Description** (optional) and **Allow Blank Value**.

Now we have determined which Table we use for Direct Searching. Next thing is that we must decide from which column we want to have the Output Value. This can be established by entering the appropriate Field of the **Table Relation Direct Search** in the **Return Field Direct Search**. We can look for that Field using the *AssistEdit* that opens a list of Fields that are related to the Table we entered in the **Table Relation Direct Search**. You can only fill this Field if the Field **Table Relation Direct Search** is filled.

So, if we look at the screenshot above, we see that we want to search for a *Currency Factor* based on the *Currency* in the **Currency** Table. The **Currency** Table has only *one* primary key with *1* element, so the **Expression** in the **Formula Line** exists of only one Variable that matches the Currency.

However, in case of multiple elements in the primary key, we can leave some out in the Expression (by using the *asterisk**). This could mean that there are more Lines on the Currency Table that match the primary key that are left. In that case we must decide the direction in which we want to search. That is defined in the Field **Search Method Direct Search**.

There are two options:

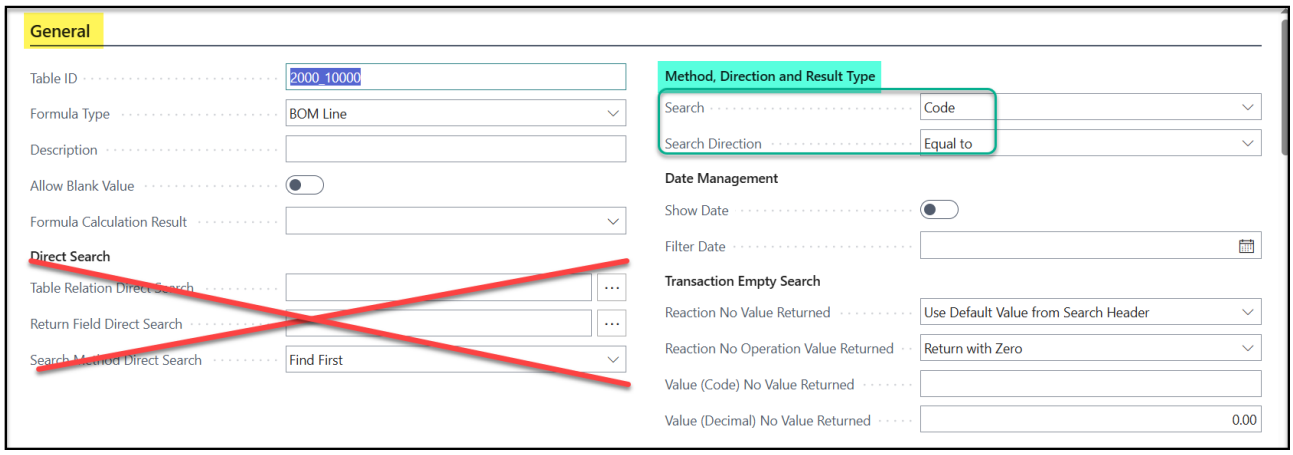
Find First	The code uses a *SETRANGE* on the Values in the search expression where after a FINDFIRST is used to find the record for which the Field Value of the Field set in **Return Field Direct Search** is returned.
Find Last	The code uses a *SETRANGE* on the Values in the search expression where after a FINDLAST is used to find the record for which the Field Value of the Field set in **Return Field Direct Search** is returned.

The default value of **Search Method Direct Search** is *Find First*.

Note
If no record is found, the Field **Reaction No Value Returned** decides how the system reacts. This is discussed later in this paragraph.

Method, Direction and Result Type

If we do not use **Direct Search** in existing tables, we must create a Search Table ourselves. The first thing we need to do is determine some search parameters.



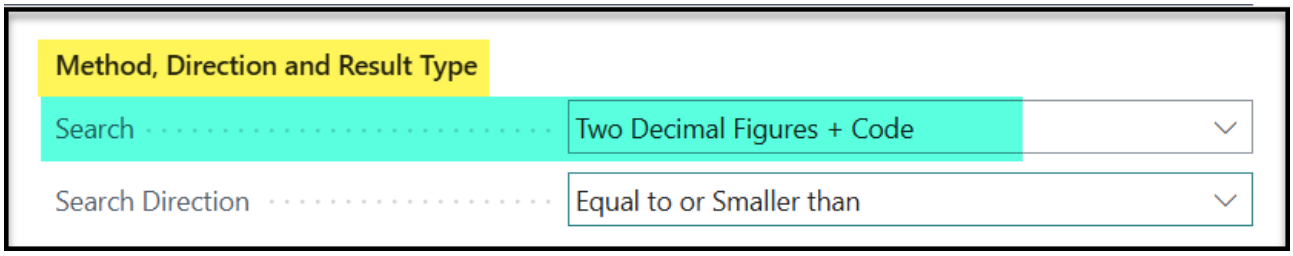
Note
If we want to make a **Search Table** ourselves and not use Direct **Search**, the Fields **Table Relation Direct Search** and **Return Field Direct Search** need to stay *empty*.

The first Field we want to decide is **Search**. This Field decides what Field *Types* are entered in the **Expression** on the **Formula Line** we want to search with.
You can choose between the following seven options:

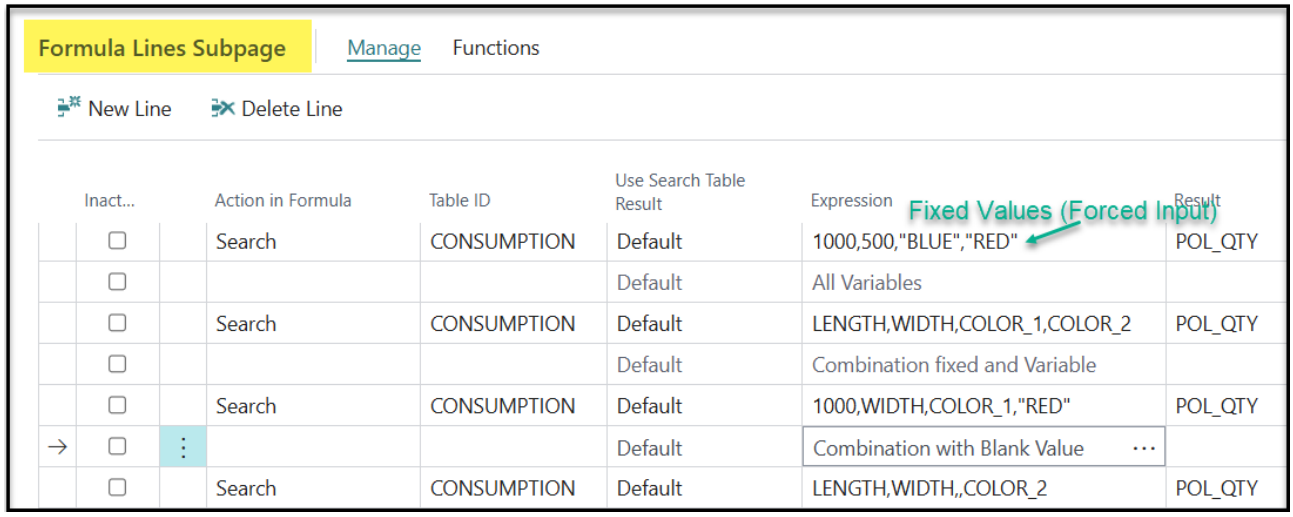
One Decimal Figure	1 Decimal Figure in search expression
Two Decimal Figures	2 Decimal Figures in search expression
Three Decimal Figures	3 Decimal Figures in search expression
One Decimal Figure + Code	1 Decimal Figure and up to 5 Codes in search expression
Two Decimal Figures + Code	2 Decimal Figures and up to 5 Codes in search expression
Three Decimal Figures + Code	3 Decimal Figures and up to 5 Codes in search expression
Code	up to 5 Codes in search expression

All options with use of *Code* make it possible to use a maximum of *5* Code Values in the **Expression** Field on the ***Formula Line***. On the Formula Line in the Field **Expression,** all elements (Decimals and/or Codes) must be addressed as either **Global Variables**, **VarDim Types** or plain **Text** separated with a comma, if it contains more than one element.

Plain Text needs to be between hyphens (i.e., "BLUE")
If a "*blank*" Value is allowed as part of the search expression, this is represented in the element string as two commas (,,) with no space in between.
Let us assume that we have a Search Table with an Input of *4* elements: two Decimal Figures and two Codes. The **Search** Field must have the option *Two Decimal Figures + Code.*



In the screenshot below, we see some examples of how the elements are positioned in the **Expression** on the **Formula Line**.



Inact...	Action in Formula	Table ID	Use Search Table Result	Expression	Result
☐	Search	CONSUMPTION	Default	1000,500,"BLUE","RED"	POL_QTY
☐			Default	All Variables	
☐	Search	CONSUMPTION	Default	LENGTH,WIDTH,COLOR_1,COLOR_2	POL_QTY
☐			Default	Combination fixed and Variable	
☐	Search	CONSUMPTION	Default	1000,WIDTH,COLOR_1,"RED"	POL_QTY
→ ☐ ⋮			Default	Combination with Blank Value ···	
☐	Search	CONSUMPTION	Default	LENGTH,WIDTH,,COLOR_2	POL_QTY

The second Field is **Search Direction**. This Field decides the Search Direction in the Search Table Lines. You can choose between the following three options:

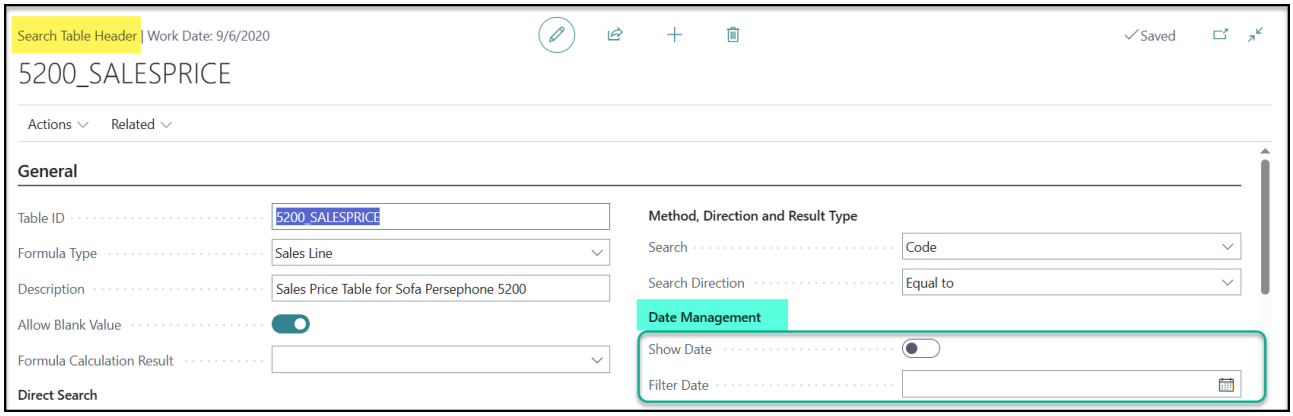
Equal to or Smaller than	Search for an Input Value in the Table that is equal or smaller than the Value in the **Expression** on the Formula Line.
Equal to or Larger than	Search for an Input Value in the Table that is equal or larger than the Value in the **Expression** on the Formula Line.
Equal to	Search for an Input Value in the Table that is equal to the Value in the **Expression** on the Formula Line.

The first two options can only work in combination with ranges (Decimal Figures). Searching for a *Code* always looks for the exact same Value. So, if only Code Fields are in the search expression, the option *Equal to* must be chosen. With Decimal Fields it is quite common that you look for smaller or larger Values in the **Input part** of the **Search Table**.

An example:
Let us assume that we have Bars in stock in different **Lengths** (**cm**): 20, 50, 100, 200 and 500. We can cut the Bars make-to-measure. A Customer wants to have a Bar of 180 cm. That means that we must look for a standard Bar Length that is bigger than 180 cm, so that we can cut it down to the requested Length. The closest standard Length is 200 cm. To find this we need a **Search Table** with all the standard Lengths with a **Search Direction** *Equal to or Larger than*.

Note
When more than one Decimal is included in the search and the **Search Direction** is *Equal to or Smaller than* or *Equal to or Larger than*, the **TRIMIT Formulas** makes sure that the record found is true for all the Decimal columns.

Date Management

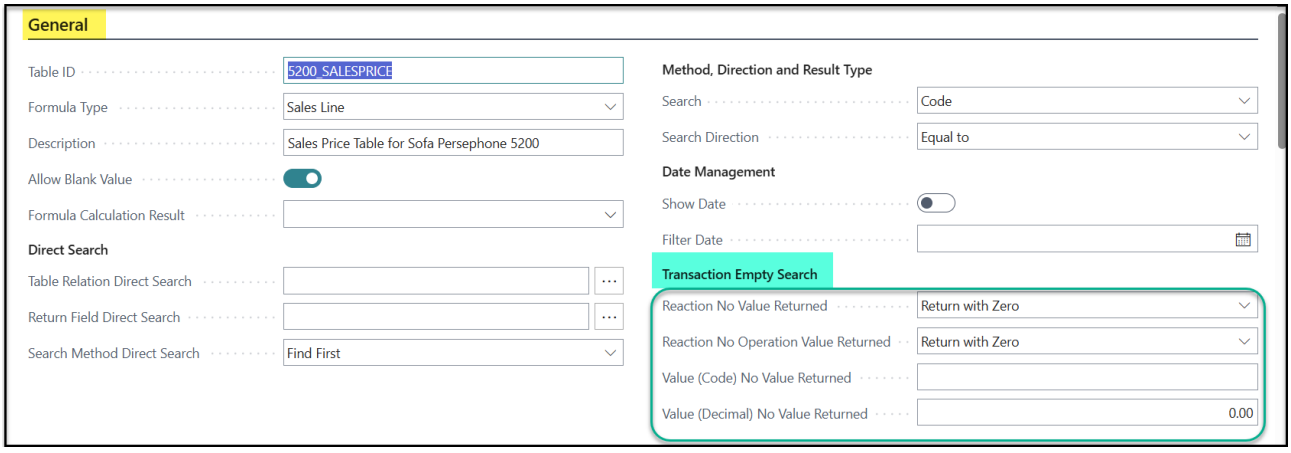


The **Search Table** also allows us to work with Dates. If the Boolean **Show Date** is set to *TRUE*, an extra (initial) search Field is added to the Search Table in the form of an extra column of **Type** *Date*. The **Show Date** can be used in connection with Formulas with **Formula Type** *Sales Line* or *Purchase Line*. The Date used is the same as the Date for the Sales Price and Unit Cost Search.

The **Filter Date** Field can be used to limit the call of the **Search Table Lines** to a certain Date range to help get a better overview in connection with maintaining Data on the **Search Table Lines**.

Transaction Empty Search

In this section we discuss Fields that define what the system should generate as Output if we cannot find a proper Input Value.



With the option Field **Reaction No Value Returned** determines which Output Value will be placed in the Variable in the Field **Result** on the **Formula Line**. If the **Search Table Header** is in use in connection with a **BOM Component Line**, it only influences the execution on Formulas that are connected to a BOM Component Line of **Type** <> *Operation* or **Type** <> *Setup*.
For these two BOM Component Line Types, we have another setting discussed later in this section.

You can choose between five options:

Return with Zero	If the **Result** Variable is of Type *Code*: value *Blank ("")* is returned. If the **Result** Variable is of Type *Decimal*: value *0* is returned.
Return without Overwrite	The current Value of the Variable in the **Result** Field on the **Formula Line** will be unchanged.
Return with Search Base	This option can be used if the Search Base contains one to three Decimals. If the **Result** Variable is of Type *Decimal*, the Decimal Value that is initially used as input is returned as Output into the **Result** Field. The Value returned depends on which **Use Search Table Result** Field is chosen on the **Formula Line** and the number of Decimals that are chosen in the **Search** Field in the **Search Table Header**. The system places the return Values in the following **Result** Fields. • *Search Decimal 1 -> "Result 1 (Decimal)"* • *Search Decimal 2 -> "Result 2 (Decimal)"* • *Search Decimal 3 -> "Result 3 (Decimal)"*
Error	The Formula execution is interrupted and will raise an Error message.
Use Default Value from Search Header	If the **Result** Variable is of Type *Code,* the Value in the Field **Value (Code) No Value Returned** is used as result (discussed later in this section). If the **Result** Variable is of Type *Decimal* the value in the Field **Value (Decimal) No Value Returned** is used as result (discussed later in this section).

With the option Field **Reaction No Operation Value Returned,** it is determined which Output Value will be placed in the Variable in the Field **Result** on the **Formula Line** in connection with **BOM Component Lines** of **Type** *Operation* or *Setup*.
You can choose between the same options as described in the Table above.

Note
Because we separate the action to be taken on different BOM Component Line **Types**, it is possible to use the same Formula on more than one BOM Component Line. Depending on the BOM Component Line **Type**, we can have different Return actions.

If in the Fields **Reaction No Value Returned** and/or the **Reaction No Operation Value Returned** Option *Use Default Value from Search Header* is chosen, we need to establish these Values on the **Search Table Header**. This can be done in the Fields **Value (Code) No Value Returned** in case the Variable in the Field **Result** in the Formula Line is of Type *Code* and/or **Value (Decimal) No Value Returned** in case that Variable is of Type *Decimal*.

Input Column Type Decimal

Above we have described some general settings about the setup of a Search Table Header. The next thing is defining the Input columns. There are two Types of input: *Decimal* and *Code*.
In this section it is discussed how to set up the Input columns for Decimals.

If on the Formula Line Global Variables and/or Values in the **Expression** are of Type *Decimal*, we go to the FastTab ***Column Decimal Figures***. In this FastTab we see three Text Fields representing the Captions of the Decimal columns in the **Search Table**.

*Earlier in this chapter we discussed a case with make-to-measure Bars to be cut from Bars with a standard Length. The **Expression** has a Variable for the **Bar Length**, which is a Decimal. Therefore, we need a Search Table Input column called: Bar Length (cm).*

Tip:
In case of Decimals, it is useful to have the **Unit of Measure** in the **Caption** of the **Search Table** column. So, the user knows exactly what content should be visible in the Search Table. For instance, a Length of *200* does not mean anything until you know that it is in *mm*, *cm*. *inches*, etc.

The Fields in this FastTab need to be used in combination with the Field **Search** on FastTab ***General*** according to the Table below:

Search	Which Fields in FastTab Column Decimal Figures
One Decimal Figure	Column 1 (Name)
Two Decimal Figures	Column 1 (Name) and Column 2 (Name)
Three Decimal Figures	Column 1 (Name), Column 2 (Name) and Column 3 (Name)
One Decimal Figure + Code	Column 1 (Name)
Two Decimal Figures + Code	Column 1 (Name) and Column 2 (Name)
Three Decimal Figures + Code	Column 1 (Name), Column 2 (Name) and Column 3 (Name)
Code	None of the Column Names are used

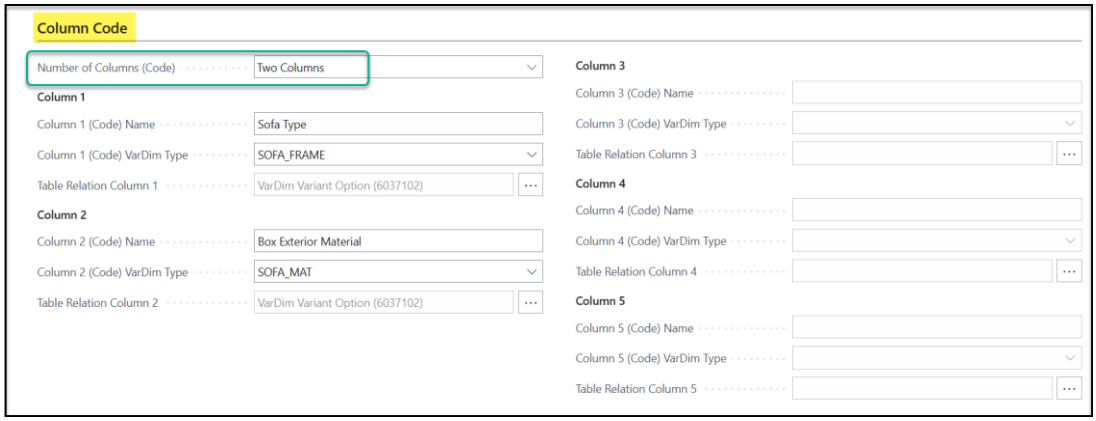
Note
The maximum number of Input columns of Type *Decimal* is 3!

Input Column Type Code

When the Formula Line in the Field **Expression** shows Code in the form of Global Variables of the Type *Code* and/or only plain *Text*, we go to the FastTab ***Column Code***. For these Types of columns, it is not enough to only establish the Captions, but we also need to relate those columns to an option Table.

The Fields in this FastTab can only be used in combination with the Field **Search** in the FastTab ***General.*** These Fields need to be entered in combination with the following Options:

- *One Decimal Figure + Code*
- *Two Decimal Figures + Code*
- *Three Decimal Figures + Code*
- *Code*



In the Field **Number of Columns (Code)**, we determine how many Input columns of Type *Code* are required. This number depends on the number of **Code** elements in the Field **Expression** on the **Formula Line**.

> **Note**
> The maximum number of Input columns of Type *Code* is 5!

For every **Column** of Type *Code*, there are three Fields to set up. The Field **Column 1-5 (Code) Name** can be filled with a **Description** representing the Captions of the columns in the **Search Table**.

The Fields **Column 1-5 (Code) VarDim Type** can be filled with a **VarDim Type** from the VarDim Type Table. If you do so, the Field **Table Relation Column 1-5** will be linked automatically with the ***VarDim Variant Option (6037102)*** Table. This means that we can create Data in the **Search Table Lines** with the help of **VarDim Variants Options** belonging to this VarDim Type.

> **Note**
> The Values of corresponding expression elements need to be in accordance with the Values to be found in the Input column. This often means that the **VarDim Type** in the **Expression** and in the **Column Code** are identical. But that is not a requirement.

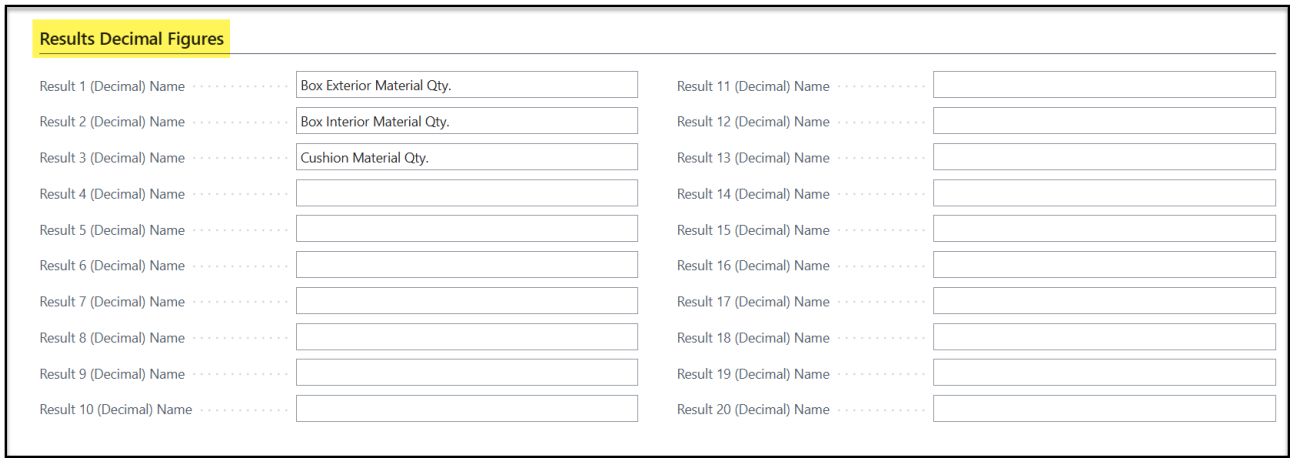
You can also leave the Field **Column 1-5(Code) VarDim Type** empty. That gives you the opportunity to create Search Table Lines with Values from other Tables in BC/TRIMIT. However, you can only use Tables with one element primary key with a maximum Length of *20* characters.
The ***Item (18)*** and ***Master (6037103)*** Tables are mostly used. Besides these Tables, you can also choose the following four Tables:

349	***Dimension Value***
6037015	***Work Center (TRIMIT)***
6037102	***VarDim Type*** (automatically entered after entering **Column 1-5 (Code) VarDim Type**)
6037118	***Assortment***

Output Column Type Decimal

In the previous sections the general setting on a **Search Table Header** is described and the Input part of it. In the next two sections the Output part is discussed.

In the FastTab ***Results Decimal Figures***, we can define the Decimal columns for a **Search Table**.
For every column for which we have a Decimal Value or Global Variable in the **Expression** on the **Formula Line**, we need to define the Caption for the column in the Search Table. These can be set up in the **Result 1-20 (Decimal) Name**. The Output Value of the which column that will be returned to the Variable in the Field **Result** depends on the Field **Use Search Table Result** (Option *Decimal Figure 1-20*).

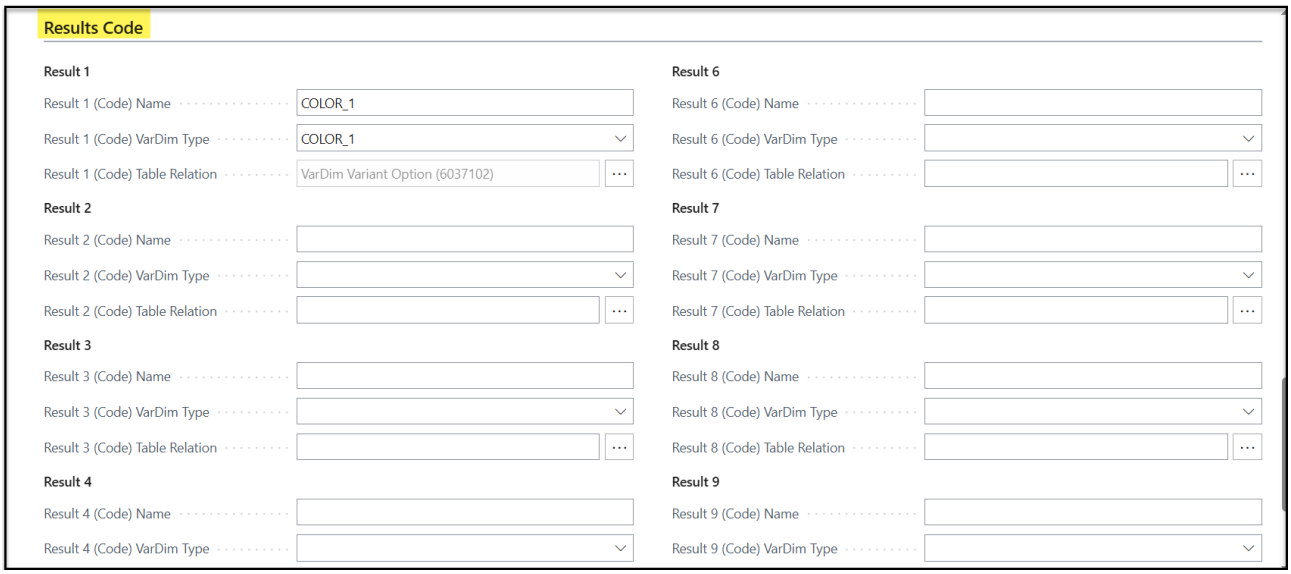


Note
The maximum number of Output Decimal columns is *20*.

Output Column Type Code

In the FastTab ***Results Code***, we can define the columns of Type *Code* in a **Search Table**. For every column for which we have a Text or Global Variable of Type *Code* in the **Expression** on the **Formula Line**, we need to define the Caption for the column in the Search Table. These can be set up in the **Result 1-10 (Code) Name**.

For every **Column** of Type *Code*, there are two additional Fields to set up. The Fields **Result 1-10 (Code) VarDim Type** can be filled with a **VarDim Type** from the VarDim Type Table. If you do so, the Field **Result 1-10 (Code) Table Relation** will be linked automatically with the ***VarDim Variant Option (6037102)*** Table. This means that we can create Data in the **Search Table Lines** with the help of **VarDim Variants Options** belonging to this VarDim Type.



You can also leave the Field **Column 1-5(Code) VarDim Type** empty. That gives you the opportunity to create Search Table Lines with Values from other Tables in BC/TRIMIT. However, you can only use Tables with one element primary key with a maximum Length of *20* characters.
The ***Item (18)*** and ***Master (6037103)*** Table are mostly used.

The Output Value of which of these columns that will be returned to the Variable in the Field **Result** depends on the Field **Use Search Table Result** (Option *Code 1-10*).

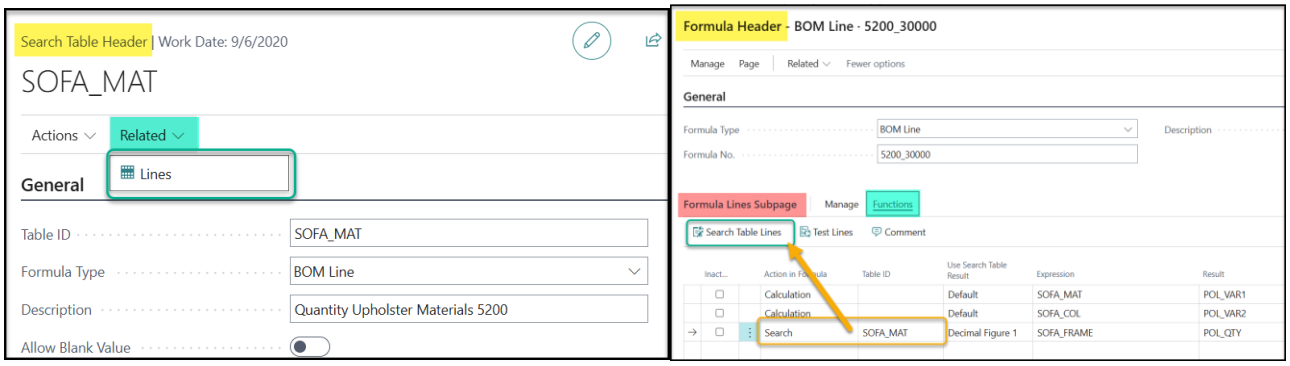
> **Note**
> The maximum number of Output columns of Type *Code* is *10*!

Data in Search Table

In the previous paragraph we discussed how a Search Table Header needs to be set up:

1. General Search Table settings as Search Directions, how to handle if there are no results, etc.
2. Determine the Input columns with Captions on the columns and related Tables if required.
3. Determine Output columns with Captions and related Tables if required.

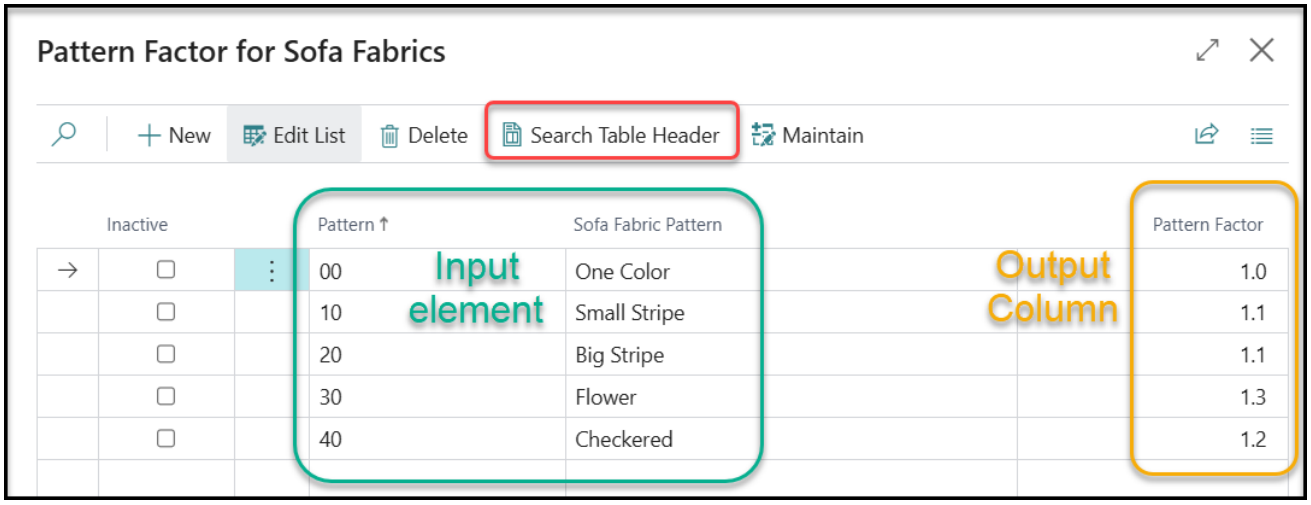
If the Search Table is not related via **Direct Search**, we need to enter the Search Table Data in the **Search Table Lines**. That can be done in two places.
The first one is on the **Search Table Header** via ***Related, Lines***.
The other place is on the **Formula Header** in the Line Actions via ***Functions, Search Table Lines***.



> **Note**
> Going to the **Search Table Lines** via the **Formula Header** requires that the appropriate **Formula Line** with **Action in Formula** *Search* and a **Table ID** must be selected.

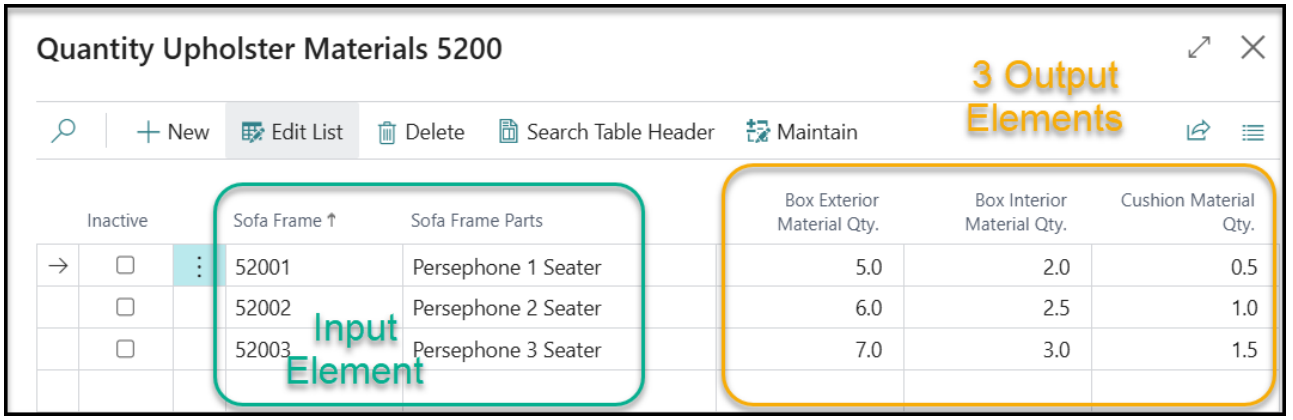
As discussed earlier in this chapter, the **Search Table Lines** consist of an *Input* part and an *Output* part. Both parts can consist of more than one element. An element of Type *Decimal* is represented by a single column expressing a Decimal Figure. An element of Type *Code* is represented by two columns: one with the *Code* of the option and one with the *Description* of the option.

Below we discuss several examples of **Search Table Lines**.
The first one is simple with one Input element (*Code + Description*) and one Output part; in this case a *Decimal* column. (You could also have an Input element of Type *Decimal* and/or an Output element of Type *Code*. Every combination is possible.)



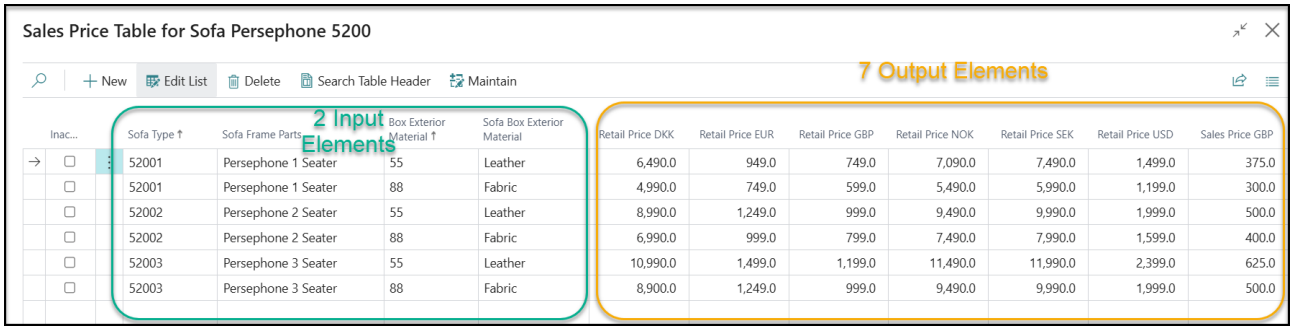
Note
It is always possible to add extra Input or Output columns or make other changes. You can go directly to the **Search Table Header** from the **Search Table Lines**.

In the next screenshot we see an example of **Search Table Lines** that has one Input element (Type *Code*), but multiple Output columns (all Type *Decimal*). Note that the Output part could consist of any combination of elements of Type *Code* and/or *Decimal*.



Note
In case of multiple Output elements, do not forget to point at the right Output element via the Field **Use Search Table Result** in the Formula Line.

The last example of Search Table Lines we show has multiple Input elements and multiple Output elements. However, any combination of Input elements Type *Code* or *Decimal* is possible.
That is also the case for any combination on the Output part. Also note that a situation is possible with multiple Input elements (any combination of Types) and only one Output element of any Type.

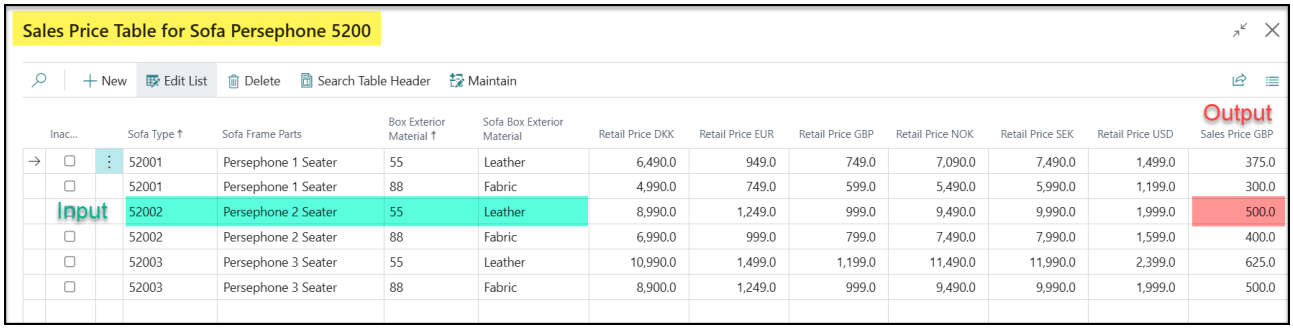


Inac...	Sofa Type	Sofa Frame Parts	Box Exterior Material	Sofa Box Exterior Material	Retail Price DKK	Retail Price EUR	Retail Price GBP	Retail Price NOK	Retail Price SEK	Retail Price USD	Sales Price GBP
	52001	Persephone 1 Seater	55	Leather	6,490.0	949.0	749.0	7,090.0	7,490.0	1,499.0	375.0
	52001	Persephone 1 Seater	88	Fabric	4,990.0	749.0	599.0	5,490.0	5,990.0	1,199.0	300.0
	52002	Persephone 2 Seater	55	Leather	8,990.0	1,249.0	999.0	9,490.0	9,990.0	1,999.0	500.0
	52002	Persephone 2 Seater	88	Fabric	6,990.0	999.0	799.0	7,490.0	7,990.0	1,599.0	400.0
	52003	Persephone 3 Seater	55	Leather	10,990.0	1,499.0	1,199.0	11,490.0	11,990.0	2,399.0	625.0
	52003	Persephone 3 Seater	88	Fabric	8,900.0	1,249.0	999.0	9,490.0	9,990.0	1,999.0	500.0

Example Search Process

Let us elaborate in this example a little more how the execution of a Formula works with a Search Table that has a combination of Input elements and a combination of Output elements.

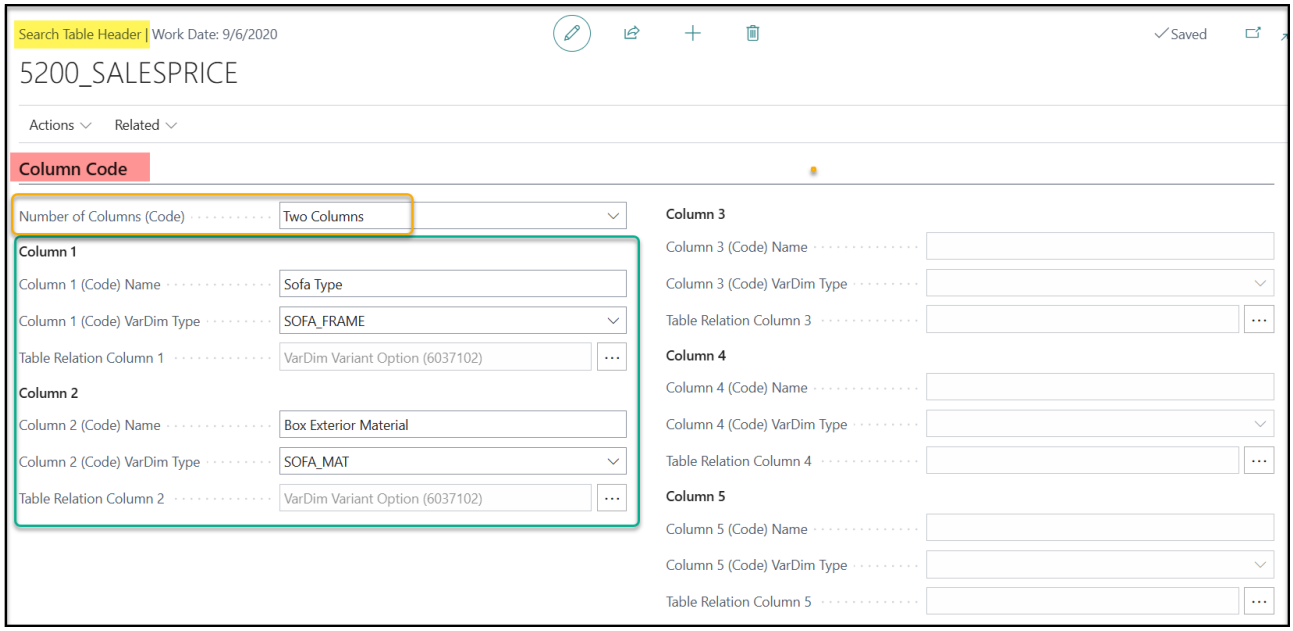
The case:
We want to sell a *2-Seater Sofa* with *Leather* upholstery to a Customer in the UK.
This Customer is not a Retail Customer, and we need the **Unit Price** in *GBP*.
We have set up a **Search Table Header** with two Input elements:
The number of Seats (SOFA_FRAME (Sofa Type): *1,2* or *3* Seats) and the Type of Upholstery (SOFA_MAT (Box Exterior Material): *Leather* or *Fabric*).
As Output elements we have all kinds of *Retail Prices* in different Currencies and one *Sales Price* in *GBP* for Wholesalers.



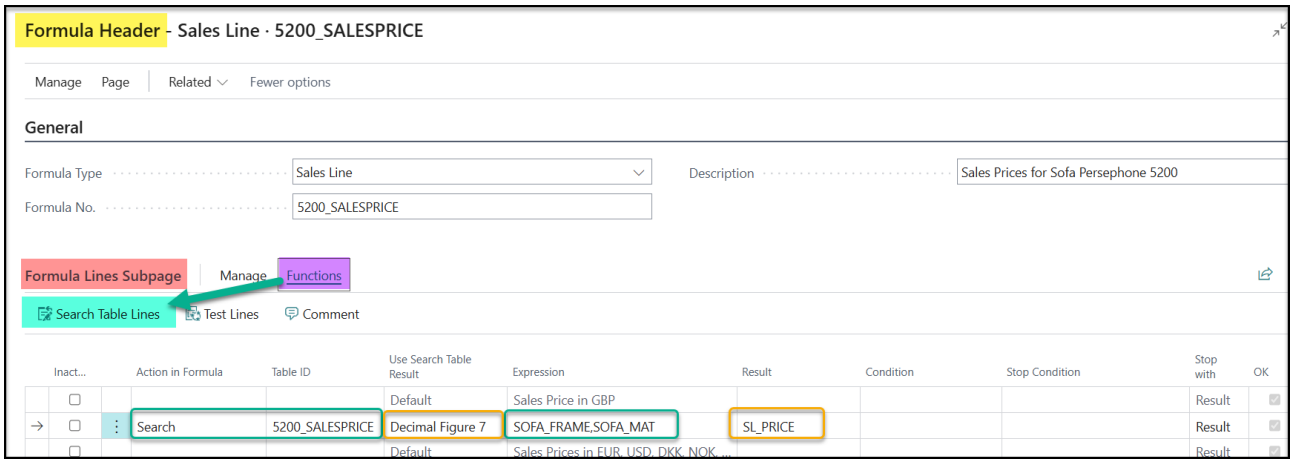
Inac...	Sofa Type	Sofa Frame Parts	Box Exterior Material	Sofa Box Exterior Material	Retail Price DKK	Retail Price EUR	Retail Price GBP	Retail Price NOK	Retail Price SEK	Retail Price USD	Sales Price GBP
	52001	Persephone 1 Seater	55	Leather	6,490.0	949.0	749.0	7,090.0	7,490.0	1,499.0	375.0
	52001	Persephone 1 Seater	88	Fabric	4,990.0	749.0	599.0	5,490.0	5,990.0	1,199.0	300.0
	52002	Persephone 2 Seater	55	Leather	8,990.0	1,249.0	999.0	9,490.0	9,990.0	1,999.0	500.0
	52002	Persephone 2 Seater	88	Fabric	6,990.0	999.0	799.0	7,490.0	7,990.0	1,599.0	400.0
	52003	Persephone 3 Seater	55	Leather	10,990.0	1,499.0	1,199.0	11,490.0	11,990.0	2,399.0	625.0
	52003	Persephone 3 Seater	88	Fabric	8,900.0	1,249.0	999.0	9,490.0	9,990.0	1,999.0	500.0

The **Search Table** will look as in the screenshot above. In this Search Table we want to find the Sales Price (GBP) for a Persephone 2-Seater with leather upholstery, so search in the Table for the combination *Persephone 2-Seater* and *Leather* (marked green). The **Sales Price** that belongs to it can be found in the column of the *7th* Output element. For this case we find *GBP 500,-* (marked red).

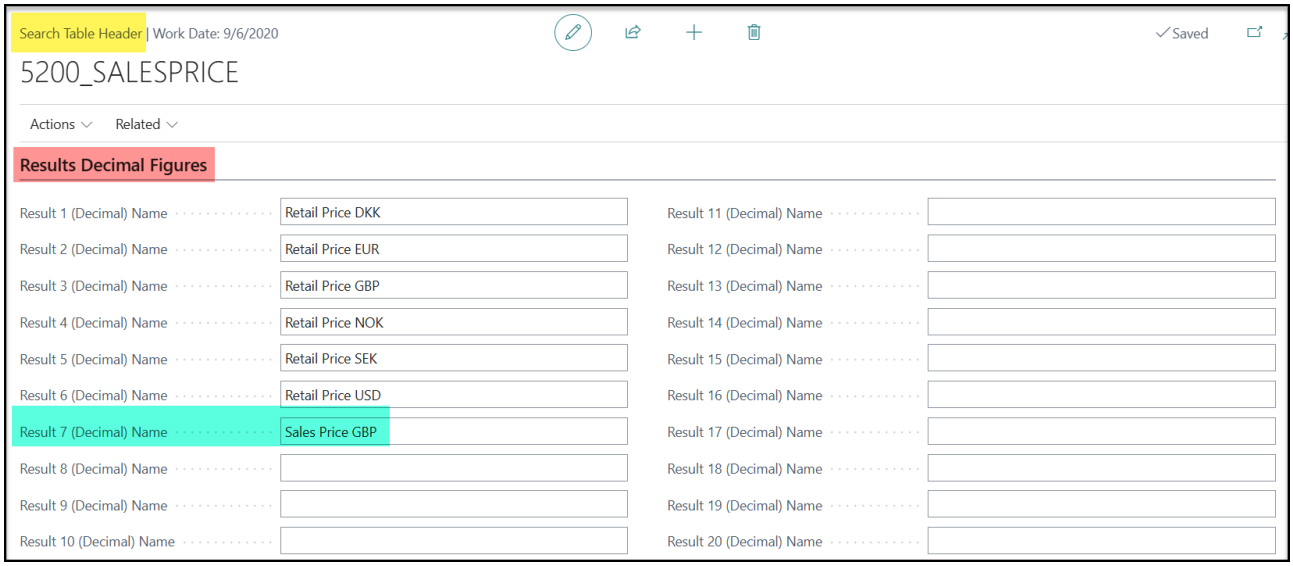
The Input part of the Search Table consists of two elements: *Sofa Type* and *Box Exterior Material*. These are linked to the **VarDim Types** *SOFA_FRAME* and *SOFA_MAT* as can be seen in the **Search Table Header**.



In the **Formula Line** we must establish that we want to look into a **Search Table**. Therefore, we enter *Search* in **Action in Formula** and the appropriate **Formula ID** for the Sales Price Table for the Sofa Persephone: *5200_SALESPRICE*. In the Field **Expression** we must put the elements we want to search for. In this case **VarDim Types** *SOFA_FRAME* and *SOFA_MAT*. When we execute this **Formula Line**, these VarDim Types get the **Values** *52002 (Persephone 2-Seater)* and *55 (Leather)*.
Note that the two VarDim Types in the Field Expression are comma-separated.



Via the Field **Use Search Table Result**, we tell the system that we want to look for *Decimal Figure 7*, which is represented by the *7th* Output element of Type *Decimal*. In our case *Sales Price GBP* as set up in the Search Table Header.



After we find the appropriate *Sales Price (GBP)* of *500,-*, this Output Value is returned to the Variable in the Field **Result** on the **Formula Line**: *SL_PRICE*. This is the **Global Variable** related to the **Unit Price** on the **Sales Line**. At the end of the execution of all Formula Lines the eventual Value of this Variable will be entered on the Sales Line.

Search Table Automations

Introduction Search Table Automations

Previously in this chapter we discussed how to use **Search Tables** in **TRIMIT Formulas** and how these Search Tables need to be set up. However, in some cases it is possible to have Search Table Automations.

The idea behind **Search Table Automations** is to enable a user not acquainted with TRIMIT Formulas to work with Search Tables. That is especially the case in Fashion Companies where several times a year a new Collection of new Products must be designed and put into BC/TRIMIT.
The users doing that do not want to be bothered by creating Formula Lines with Search Tables. Search Table Automations only works under specific circumstances. A Furniture/Configuration Product is often more complex, but of course Search Table Automations can also be useful for that Type of Products.

In the following sections, two Types of Search Table Automations are discussed.

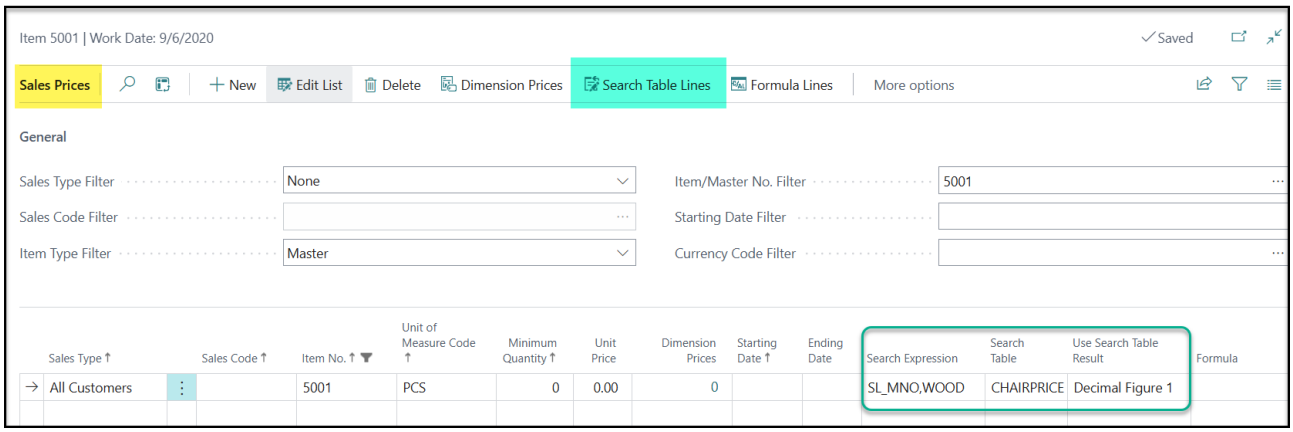
- Direct linking a Search Table without creating the Formula Line by the user.
- Automatic creation of the Search Table and the Formula Line.

Direct Linking to a Search Table

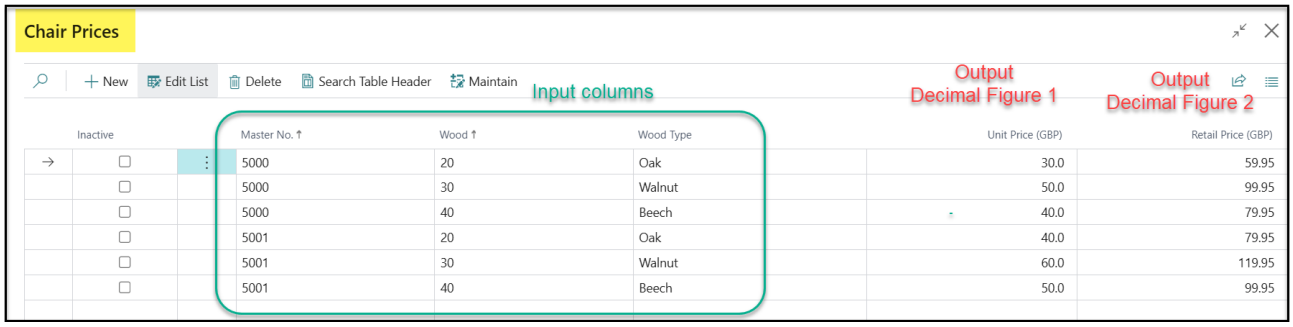
This functionality makes it possible to do a search on a **Search Table** without having to create a Formula to carry out the search. It is used in connection with a Price search and therefore connected to the **Sales Price Table (7002)** and **Purchase Price Table (7012)**.

With this Feature you still need to set up a Search Table as it would be for normal use. However, there are some restrictions. This Functionality only works with a maximum of ***5*** elements in total (maximum *5* of Type *Code* and *3* of Type *Decimal*). You can only have Output elements of Type *Decimal*, as the returned Value is placed in the Field **Unit Price** on the **Sales Line** or **Direct Unit Cost** on the **Purchase Line**.

This time we do not enter the **Table ID** of the Search Table on a Formula Line, but directly on a Line in the Price Table in the Field **Search Table**. In that Line we also need to enter the Variables in the **Search Expression** used as search criteria. These Variables are comma-separated in case there are more than one. In the **Use Search Table Result** we can define the Output (Decimal) column for the Price Table Line.



Via **Search Table Lines** you can look at or enter the Data in the Search Table Lines. Note that in this example one of the search criteria is **Master No.** on the **Sales Line** (*SL_MNO*), which is inherited from the **Item** entered/configured. This example shows you that it is possible to set up a Search Table that is only created once and can be used for multiple Masters.



Note
If you link a **Search Table** directly to a Price Table Line, no **Formula** is to be created. **Search Table Automation** on the **Sales-** and **Purchase Price** Tables cannot coexist with Formula at all.

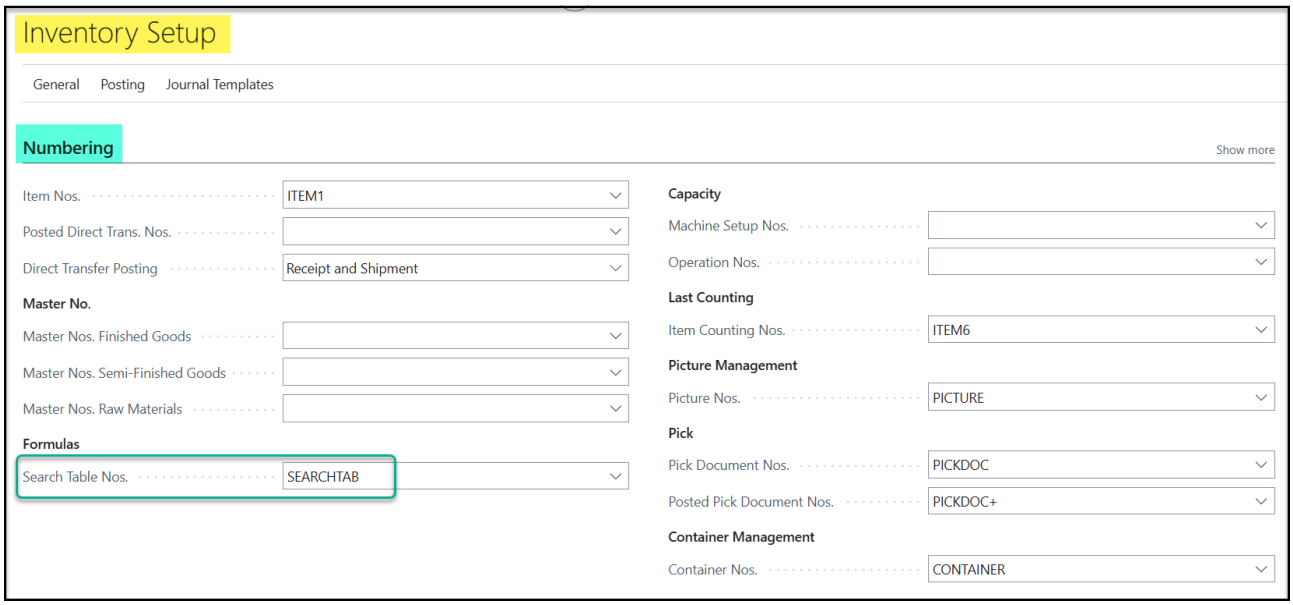
Automatic Creation of Search Tables in Price tables

In the section above it is discussed how an existing Search Table can be connected directly to a Line on Price Table Lines. In TRIMIT it is also possible to go a step further: the automatic creation of the Search Table Header. This Feature cannot only be applied in the Price Tables, but on Table **BOM Component Line (6037341)** as well.

In this paragraph several scenarios are discussed:

- *Auto-created Seach Table on **Price** table per **Size**.* (this section)
- *Auto-created Search Table on **Price** table per **Grouping** of Sizes.* (this section)
- *Auto-created Search Table on **BOM Component Line** without **Quantity per**.* (next section)
- *Auto-created Search Table on **BOM Component Line** with **Quantity per**.* (next section)

Since Search Tables will be auto-created, we need to have a **No. Series** for identifying the Search Table Header. This **No. Series** is set up in the Parameter **Seach Table Nos.** in the **Inventory Setup**.

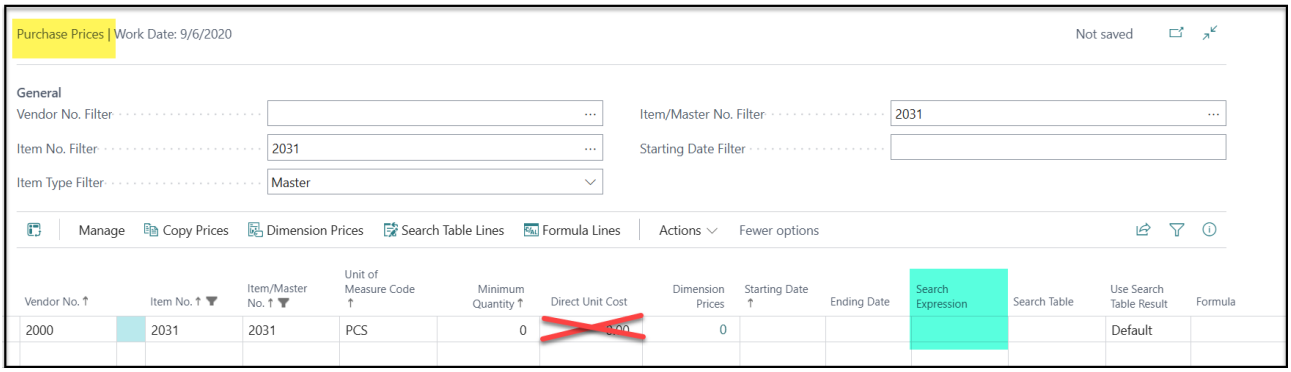


Case:
We have a Shirt that comes in different Colors and Sizes. Let us assume that this Shirt has different Purchase Prices depending on the Size, while the Sales Price is set up by a Group of Sizes.

Let us start looking at the auto-creation of Search Tables on Pricing Tables. This functionality can be used if the **Unit Price** or **Direct Unit Cost** depends on one or more **VarDim Types** in the **VarDim Master,** or by a **Grouping** on the **VarDim Variant Options** related to the **VarDim Type**.

Regarding the Case described above: the Purchase Price is depending on the Size.

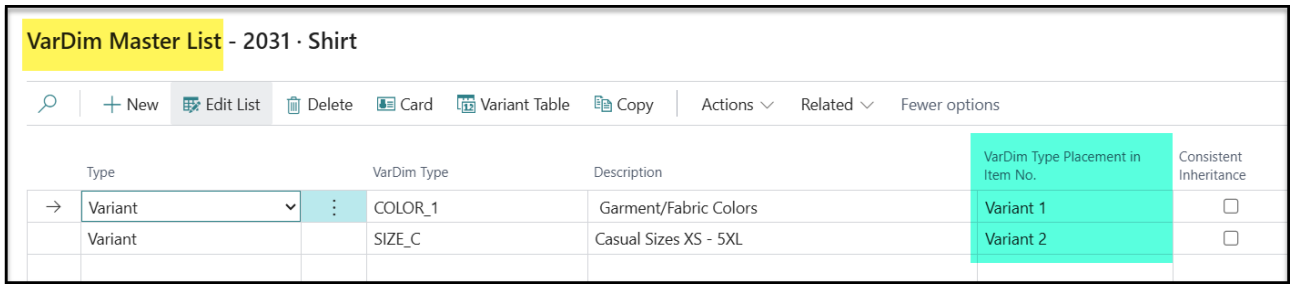
In the Purchase Price Table, we create a new **Line** as we would do in standard BC. However, if we have Purchase Prices that depend on one or more Variables on the **Item**, we do not need to enter the Purchase Price in the Field **Direct Unit Cost**[14] on the Purchase Price Table Line.



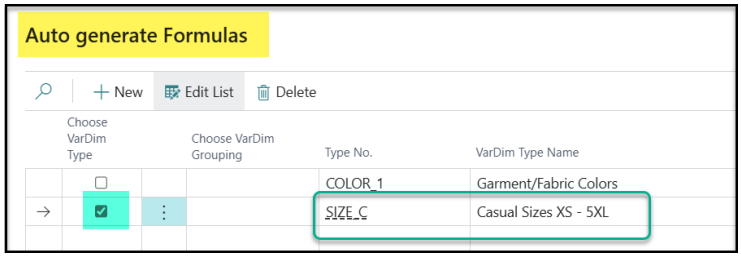
Instead, we open the page behind **Search Expression** by clicking on the *AsistEdit* button. A List of available **VarDim Types** that is found in the **VarDim Master** related to the **Master** you are creating a

[14] You can enter the **Direct Unit Cost**, but it will be overwritten by the **Direct Unit Cost** defined in the **Search Table**, if the Search Criteria are met.

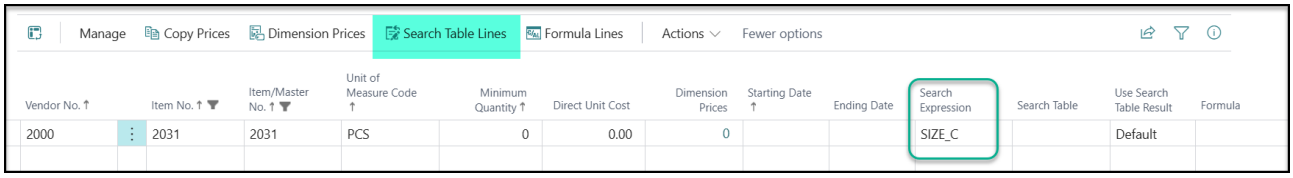
Sales- or Purchase Price Line for is shown. In this List we can only see **VarDim Types** for which the Field **VarDim Type Placement in Item No.** is entered.



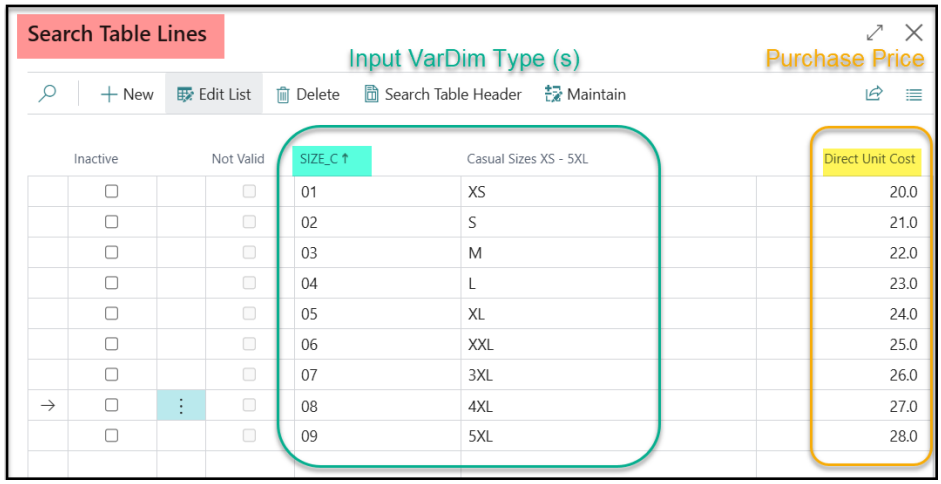
On the lookup page behind Search Expression, we determine the Variant(s) to which the Purchase Price is depending by placing a checkmark in the *Boolean* **Choose VarDim Type**. In this case: only the **VarDim Type** *SIZE_C* because we have established that the Purchase Price of the Shirt only depends on the Size.



The Field **Search Expression** on the Purchase Price Line will be populated by the checked **VarDim Types**. In case of auto-creation of Search Tables, the VarDim Types in the Search Expression are not only acting as Input elements but also define the Input columns in the Search Table to be created.

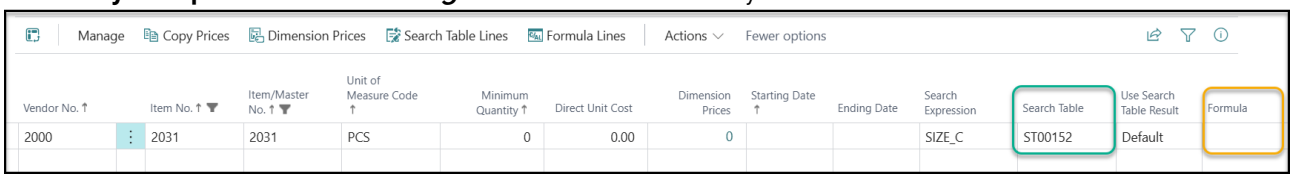


Let us have a look at the **Search Table Lines**. The Input column is (a combination of) selected **VarDim Type**(s) of the **VarDim Master**. The Output column is the **Direct Unit Cost**.



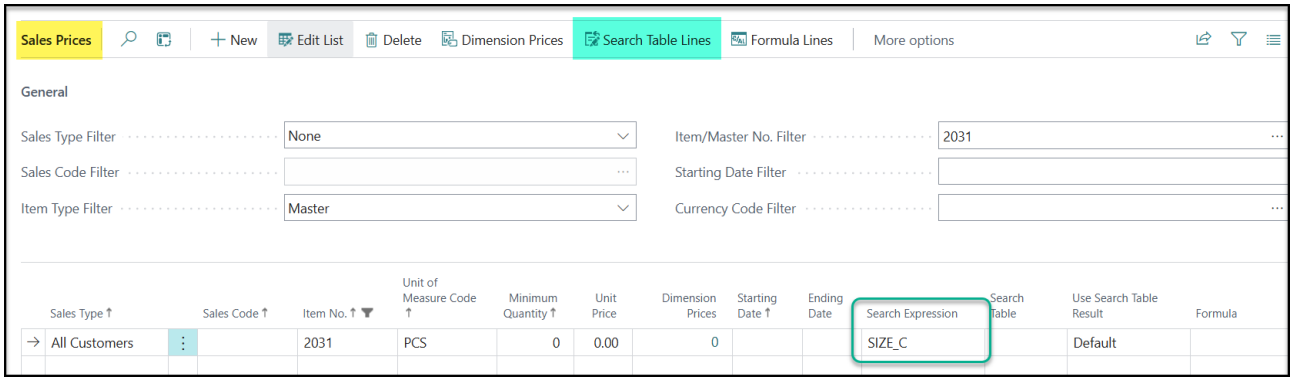
You can see in this case that every Size of the Shirt has a different Purchase Price.

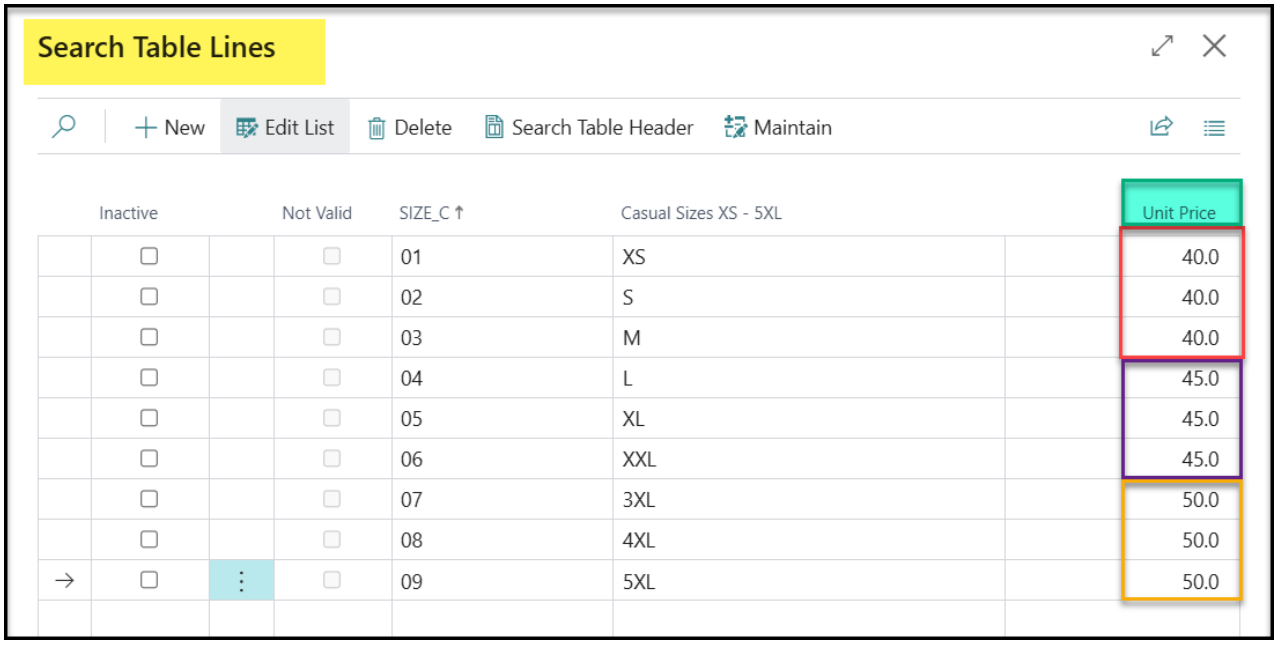
After you have entered the Data and close the Search Table, you can see on the Purchase Table Line the Field **Search Table** has been populated with a **Table ID** generated from the **No. Series** setup in the **Inventory Setup** FastTab ***Numbering***. The Field **Formula** stays '*Blank*.'



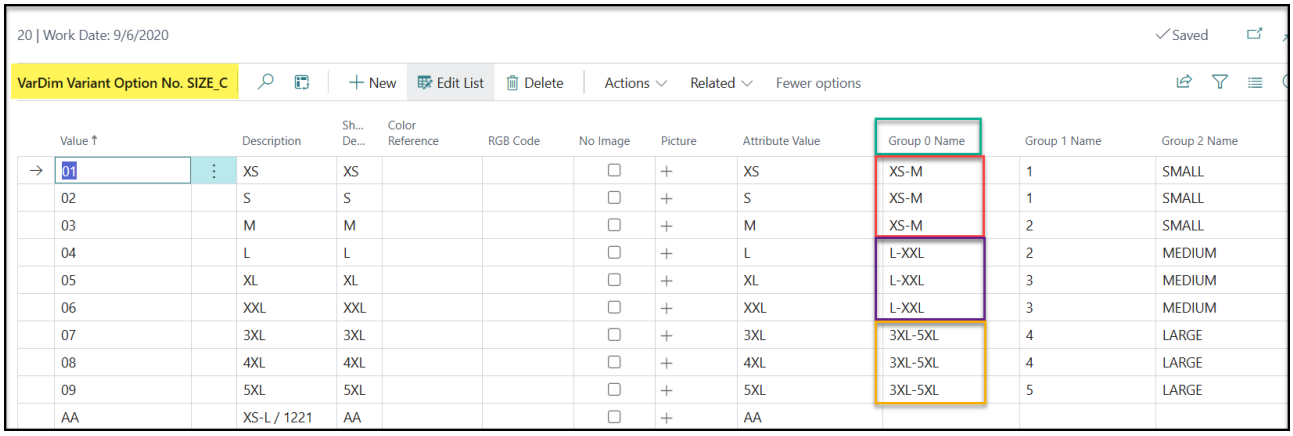
It could be very well that one of the Input elements is an exceptionally large Table. In that case, it is possible to make Groups of the Variant Options. Every Group has its own Price. For making these Groups we use the **Grouping** Feature as described in chapter *Formula Lines Structure* paragraph *Formula Expression AssistEdit* section *VarDim Type Groups*.

Also, for our Shirt we stated that we have different Sales Prices for Sizes. If we would have the SIZE_C in the **Search Expression**, we could have **Search Table Lines** as below.



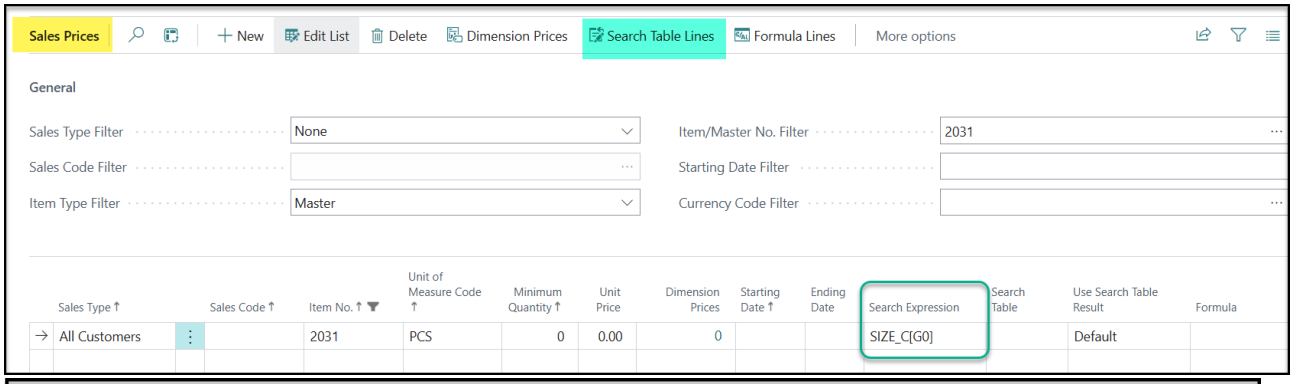


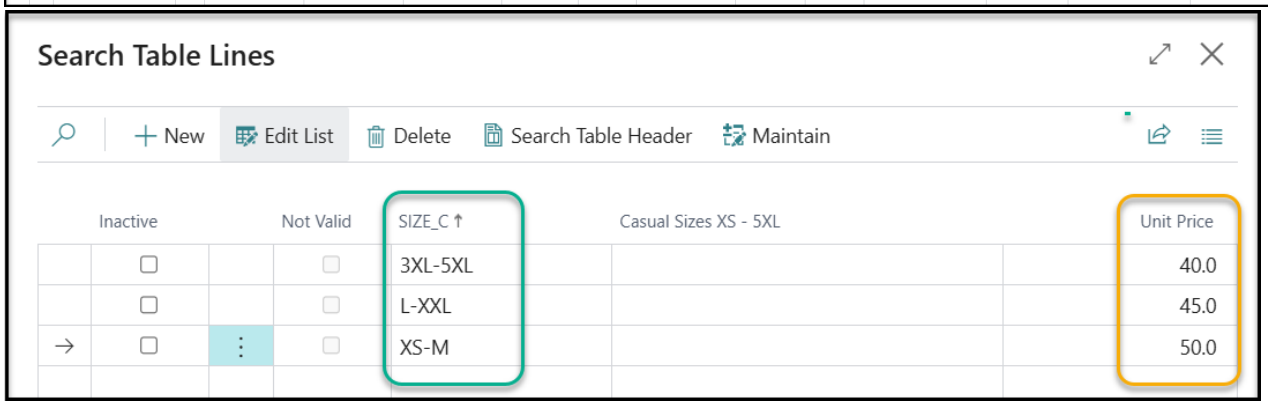
However, you could see, that there are three different **Unit Pices**: *€40,-* for the Sizes *XS-M*, *€45,-* for the Sizes *L-XXL* and *€50,-* for the Sizes *3XL-5XL*. So, you could say that we have three different Prices based on grouping of Sizes. In the **VarDim Variant Options** of *SIZE_C* these Groupings are defined in the column **Group 0 Name** (see below.)



Sizes with the same Price needs to have the same content in **Group 0 Name**.

Group 0 Name can be represented as an **Extension** *[G0]* to *SIZE_C* in the **Search Expression**. If we enter this extension on the **Sales Price Line**, we can now create **Search Table Lines** based on the Size Groupings. This way we can limit the number of Search Table Lines.





Note

As you can see in this example, it is important to have the right **Text** in the **Group 0 Name**, otherwise it will not be sorted the right way based on the Sizes it concerns, and you will make mistakes by entering the **Unit Price** – as shown in above picture.

Note

Since this concerned a direct Search Table with no Formula Lines involved, we can use the extensions [G0] – [G19] although [G10] – [G19] have not got a Function related to it. That means that you can make 20 different Group Columns with all kinds of Groupings.

Above we have described how a **Search Table** can be automatically created on **Sales-/Purchase Price** Tables. The difference with *Direct Linking,* an existing Search Table, is that for the latter we need to have created a Search Table before we can link it.
However, it has more flexibility regarding the Variables in the **Search Expression**.
In the case of the auto-creation of a Search Table on the **Sales-/Purchase Price** Line, you can only select **VarDim Types** from the **VarDim Master** with a **VarDim Type Placement in Item No.** of the **Master** to which the **Sales-/Purchase Price** Line is connected.

Note
When a **Search Table** is auto-created on a Price Table Line no **Formula** is to be created. **Search Table Automation** on the **Sales-** and **Purchase Price** Tables cannot coexist with Formula at all.

Automatic Creation of Search Tables on BOM Component Lines

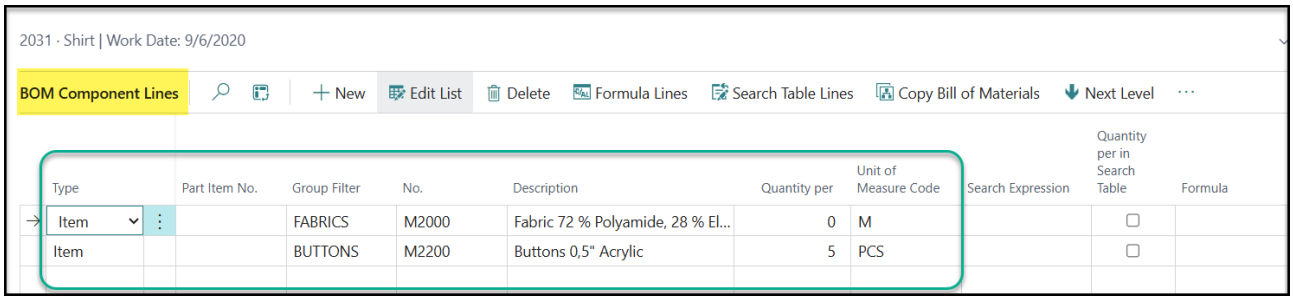
On the **BOM Component Lines** a **Search Table Header** can also be auto-created. There we have some additional Features. The functionality on the **BOM Component Line** can be used to set **VarDim Type Values** on the **Item** on the *Current* Level depending on **VarDim Type Variant Values** on the Item on the *Overlying* Level, as well as filling the **Quantity per** on the **BOM Component Line**.

The automatic Search Table creation on the **BOM Component Line,** works in a similar way as the functionality on the Sales- and Purchase Price Lines, with the difference that the functionality automatically creates a **Formula** with **Formula Lines** including a **Search Table Header** that is attached to the **BOM Component Line** instead of a direct link of the Search Table Header to the BOM Component Line.

This feature is commonly used for **Matrix** Products in the Fashion Industry, where designers create Styles every Season with a BOM Structure that is simpler than in other industries. This way manually creation of **Formulas** and **Search Tables** can be avoided in most cases. This is of course not the case for more complicated Apparel Products, such as Customized Workwear, Tailor-Made Apparel, or Orthopedic Shoes.

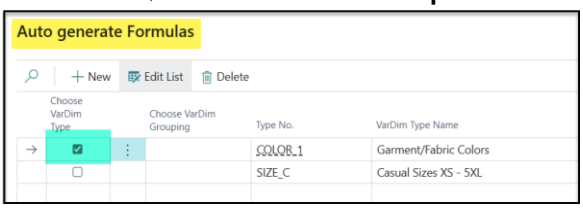
We will discuss this Feature with the same Shirt as mentioned in the section above.

The Shirt comes in different Colors and Sizes. The Shirt consists of two Raw Materials: a Fabric and a Button. You could imagine that the bigger the Size, the more Fabric needs to be used.
The Color of the Fabric is identical to the Color of the Shirt.
So, the Color of the Fabric could be inherited from the Color of the Shirt. (see paragraph *Consistent Inheritance*)
For the Buttons we have other rules. The number of Buttons is the same for each Size. However, the Buttons have a 'contrast' Color[15].
The BOM looks like this before entering the Configuration Rules.

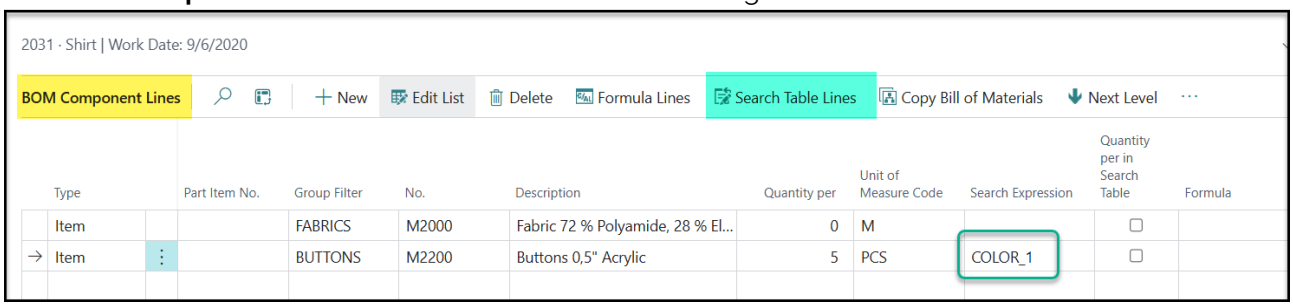


[15] If a Component has a contrast Color, it means that the Color of the Component deviates from the Color of the main Product

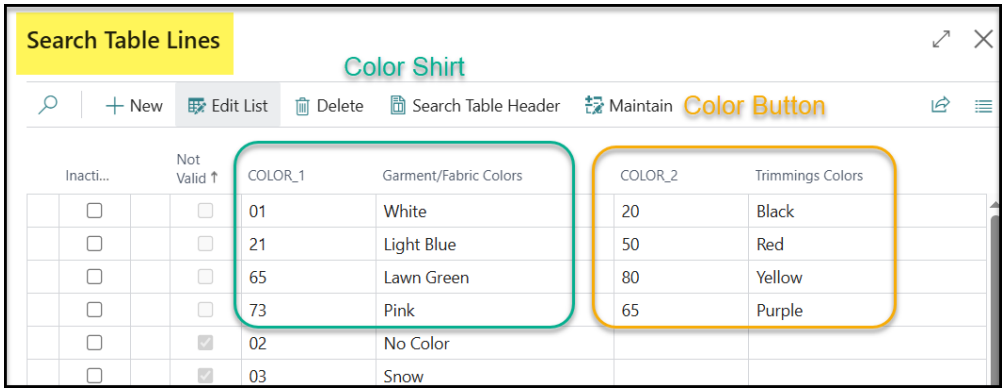
Let us look at the Buttons first. It is always *5* Pieces, but the Color of the Button depends on the Color of the Shirt. So, in the Field **Search Expression** we select only Color.



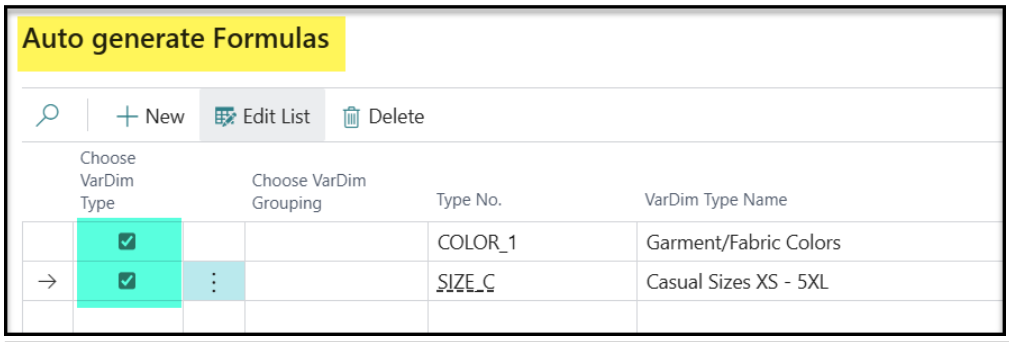
The **BOM Component Line** looks like below. From there we go to **Search Table Lines**.

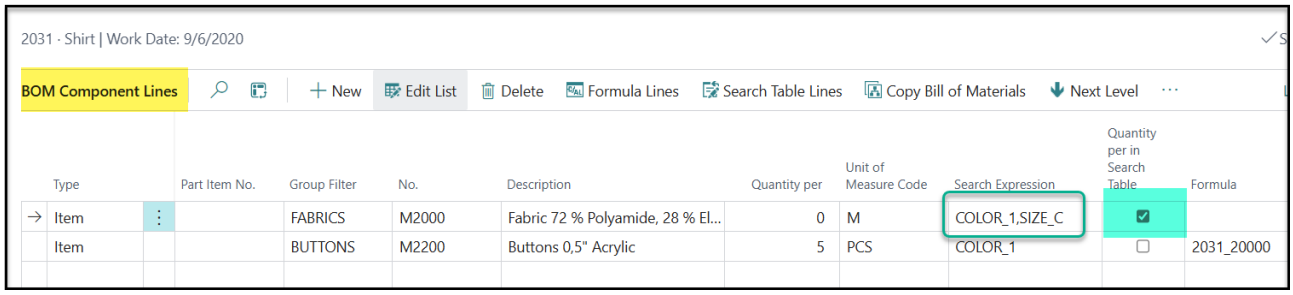


The **Search Table Lines** opens, and we can define the relationship between the Colors of the Shirt and the Color of the Button. For instance, like this:



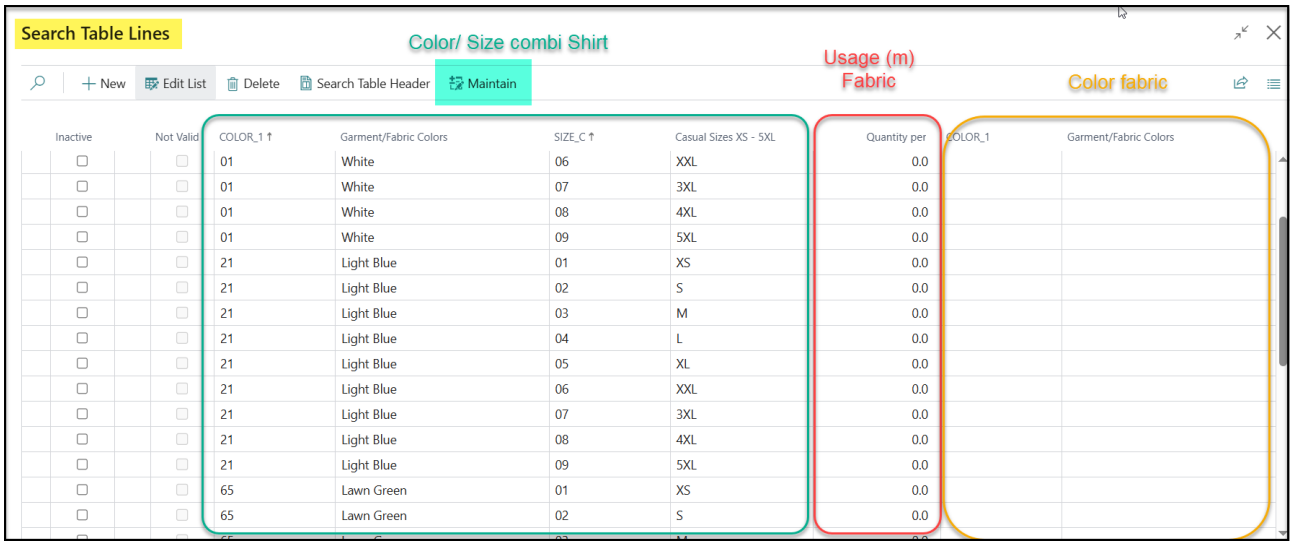
Now let us look at the Fabric. The Fabric depends on the Color and the Size. One of the possibilities is to have both VarDim Types checked in the Search Expression. See the result below:





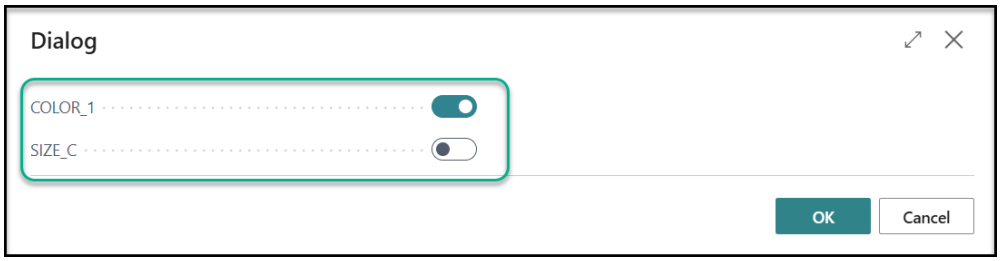
Since the usage of the Fabric depends on the Size, we need to place a checkmark in the Field **Quantity per in Search Table**. If this Boolean is set to *TRUE*, there will be a column in the **Search Table Lines** for entering the **Quantity per** for the Fabric.

You can see that the two **VarDim Types** in the **Search Expression** are *comma-separated*. The search criteria is now the combination of Colors and Sizes. This could be a long List and can be avoided by using the **Consistent Inheritance** Feature in TRIMIT. Later of that in the next paragraph. For now, let us have a look at the **Search Table Lines**.



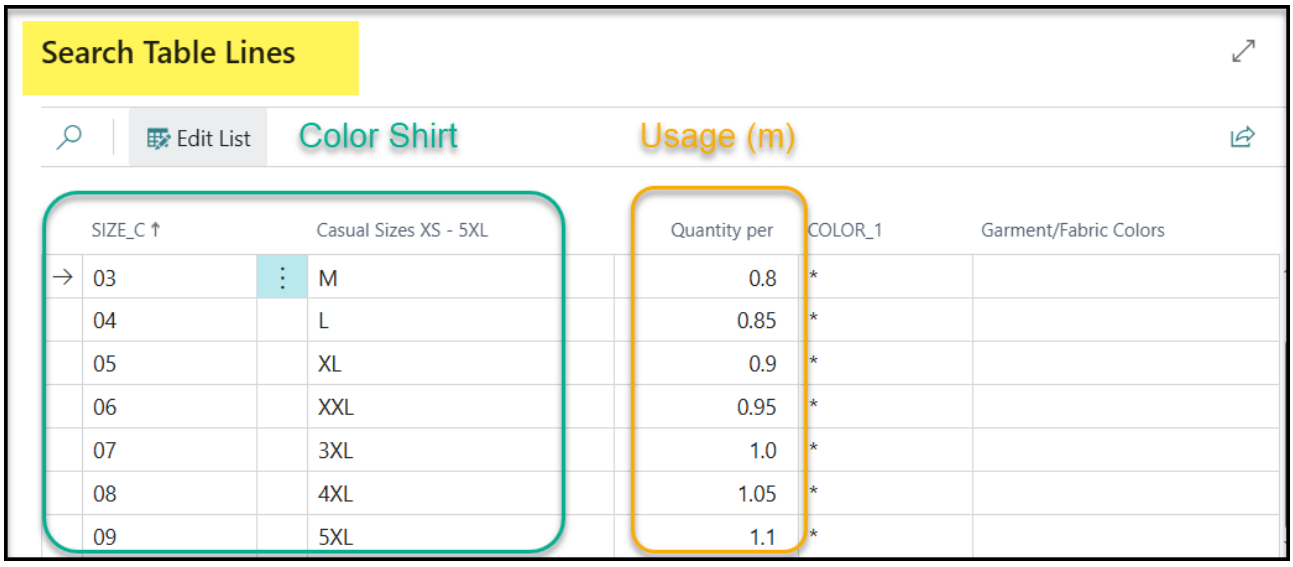
Inactive	Not Valid	COLOR_1	Garment/Fabric Colors	SIZE_C	Casual Sizes XS - 5XL	Quantity per	COLOR_1	Garment/Fabric Colors
		01	White	06	XXL	0.0		
		01	White	07	3XL	0.0		
		01	White	08	4XL	0.0		
		01	White	09	5XL	0.0		
		21	Light Blue	01	XS	0.0		
		21	Light Blue	02	S	0.0		
		21	Light Blue	03	M	0.0		
		21	Light Blue	04	L	0.0		
		21	Light Blue	05	XL	0.0		
		21	Light Blue	06	XXL	0.0		
		21	Light Blue	07	3XL	0.0		
		21	Light Blue	08	4XL	0.0		
		21	Light Blue	09	5XL	0.0		
		65	Lawn Green	01	XS	0.0		
		65	Lawn Green	02	S	0.0		

As you can see this is a big Table. Fortunately, TRIMIT has a feature that helps maintain big **Search Tables**. For that you click ***Maintain*** and choose a VarDim Type on the **Dialog** page.
First, we select *COLOR_1* to determine the dependencies on the Color. A cross-section of the Search Table opens, in which we can enter the Colors of the Fabric, which are identical in this case as the Color of the Shirt. The Color has no influence on the **Quantity per**, so we do not enter anything in that column.



COLOR_1	Garment/Fabric Colors	Quantity per	COLOR_1	Garment/Fabric Colors
01	White	0.0	01	White
21	Light Blue	0.0	21	Light Blue
65	Lawn Green	0.0	65	Lawn Green
73	Pink	0.0	73	Pink

We do the same thing, but now we select in the **Dialog** page *SIZE_C*.
Now we must enter Quantities in the column **Quantity per** since the Size does have its effect on the usage of the Fabric. In this case we leave the column for the Color empty. The *asterisk* * in that column tells us, that per Size there are already different Colors entered, which we entered earlier.

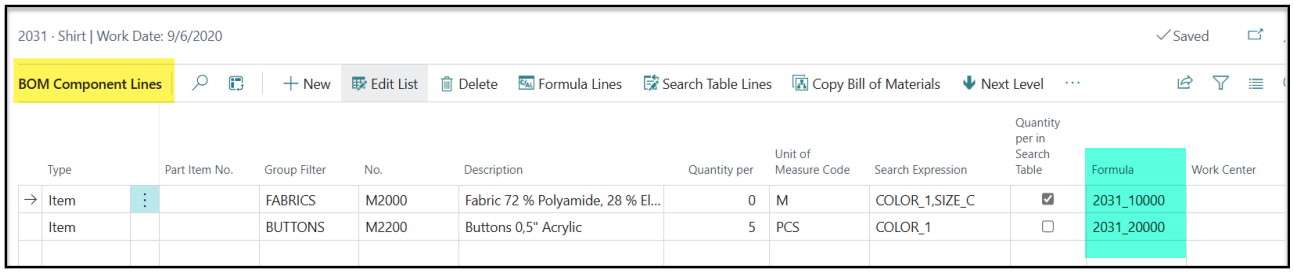


SIZE_C	Casual Sizes XS - 5XL	Quantity per	COLOR_1	Garment/Fabric Colors
03	M	0.8	*	
04	L	0.85	*	
05	XL	0.9	*	
06	XXL	0.95	*	
07	3XL	1.0	*	
08	4XL	1.05	*	
09	5XL	1.1	*	

If we go back to the **Search Table Lines**, we can see now that all Color/Size combinations of the Shirt have an appropriate Fabric Color and Fabric Usage.

Inactive	Not Valid	COLOR_1	Garment/Fabric Colors	SIZE_C	Casual Sizes XS - 5XL	Quantity per	COLOR_1	Garment/Fabric Colors
		01	White	06	XXL	0.95	01	White
		01	White	07	3XL	1.0	01	White
		01	White	08	4XL	1.05	01	White
		01	White	09	5XL	1.1	01	White
		21	Light Blue	01	XS	0.0	21	Light Blue
		21	Light Blue	02	S	0.0	21	Light Blue
		21	Light Blue	03	M	0.8	21	Light Blue
		21	Light Blue	04	L	0.85	21	Light Blue
		21	Light Blue	05	XL	0.9	21	Light Blue
		21	Light Blue	06	XXL	0.95	21	Light Blue
		21	Light Blue	07	3XL	1.0	21	Light Blue
		21	Light Blue	08	4XL	1.05	21	Light Blue
		21	Light Blue	09	5XL	1.1	21	Light Blue
		65	Lawn Green	01	XS	0.0	65	Lawn Green
		65	Lawn Green	02	S	0.0	65	Lawn Green

If we close the Search Table Lines, we can see that now for both the Fabric and the Button a **Formula** has been created automatically. The **Table ID** is created from the **No. Series** set up in **Inventory Setup** (see at the beginning of this paragraph).



Type	Part Item No.	Group Filter	No.	Description	Quantity per	Unit of Measure Code	Search Expression	Quantity per in Search Table	Formula	Work Center
Item		FABRICS	M2000	Fabric 72 % Polyamide, 28 % El...	0	M	COLOR_1,SIZE_C	☑	2031_10000	
Item		BUTTONS	M2200	Buttons 0,5" Acrylic	5	PCS	COLOR_1	☐	2031_20000	

Let us have a look at the **Formula Lines** of both Formulas.
In both Formulas we see a Formula Line with a **Search Table**.
The Formula of the Fabric has an extra Formula Line, since also the **Quantity per** is determined.

Formula Lines Subpage | Manage | Functions

New Line | Delete Line

Formula on Button

Inact...	Action in Formula	Table ID	Use Search Table Result	Expression	Result	Condition	Stop Condition	Stop with	OK
→ ☐ ⋮	Search	2031_20000	Code 1	COLOR_1	POL_VAR1			Result	☐

Formula Lines Subpage | Manage | Functions

New Line | Delete Line

Formula on Fabric

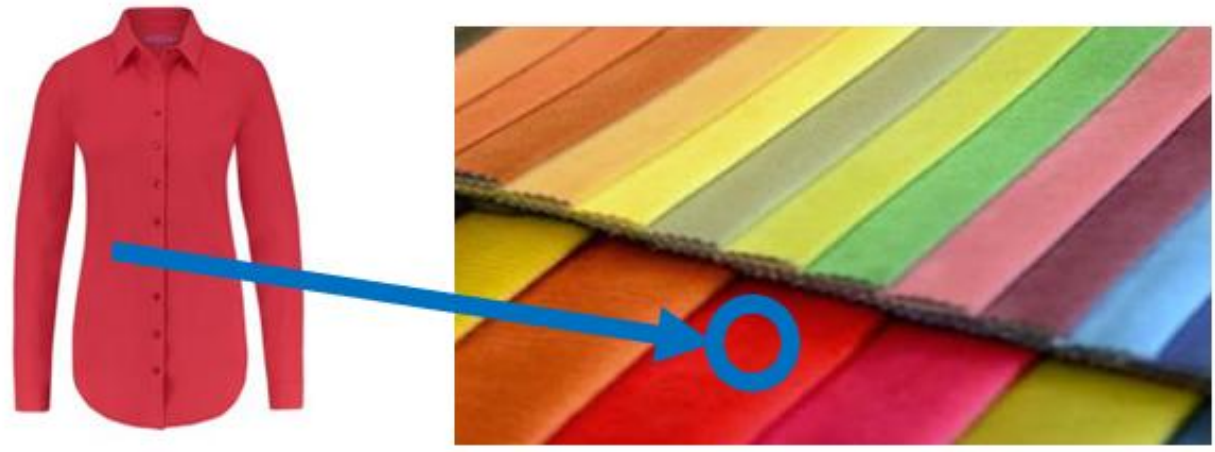
Inact...	Action in Formula	Table ID	Use Search Table Result	Expression	Result	Condition	Stop Condition	Stop with	OK
→ ☐ ⋮	Search	2031_10000	Code 1	COLOR_1,SIZE_C	POL_VAR1			Result	☐
☐	Search	2031_10000	Decimal Figure 1	COLOR_1,SIZE_C	POL_QTY		POL_QTY=0	Skip	☐

Note
This **Search Table Automation** method on **BOM Component Lines** does not work together with manually created **Formula**.

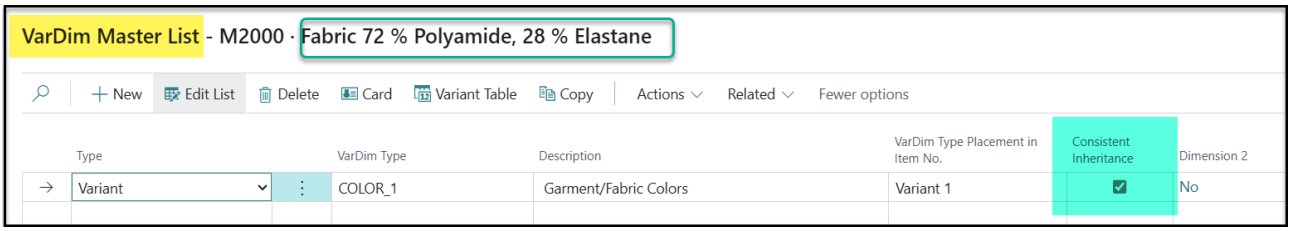
Consistent Inheritance

As discussed in the previous paragraph, it is possible for simple BOM relations having auto-created Formulas. However, linking the Variants of the Main Product and the Variant of the Component could result in huge Search Tables. Especially when those Variants are identical.
In that case the TRIMIT Feature **Consistent Inheritance** can be used.

An example:
A Shirt may come in different Colors. That Color is exactly the same as the Color of the Main Fabric, and both Colors are from the same VarDim Type.
For instance: if the Shirt is *red* of VarDim Type *COLOR_1*, the main Fabric is also *red* of VarDim Type *COLOR_1*.

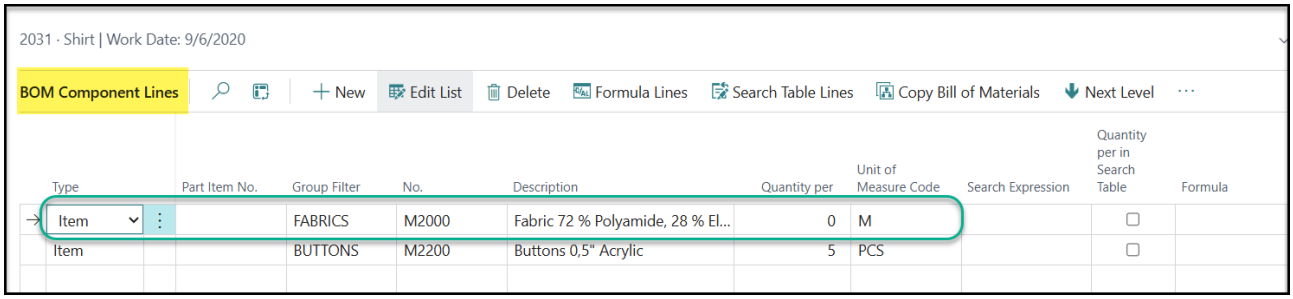


In those cases, it is not necessary to have a Search Table on the main Fabric for the Colors.
We could simply inherit the Color from the Main Product to the Component.
If you want this, you only must place a *checkmark* in the Boolean **Consistent Inheritance** on the **VarDim Type** in the **VarDim Master** of the Component for which you want to inherit the Value from an Overlying Level.

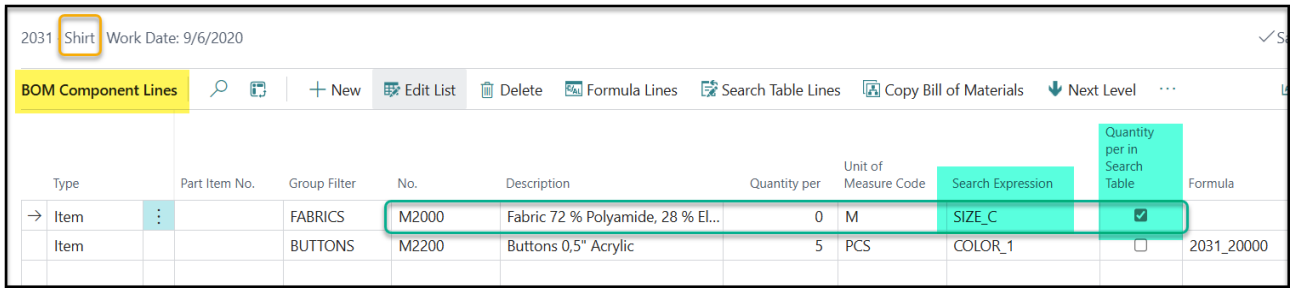


Let us have a look at the case we used in the previous paragraph and show the alternative setting using the **Consistent Inheritance** as set in the VarDim Master above.

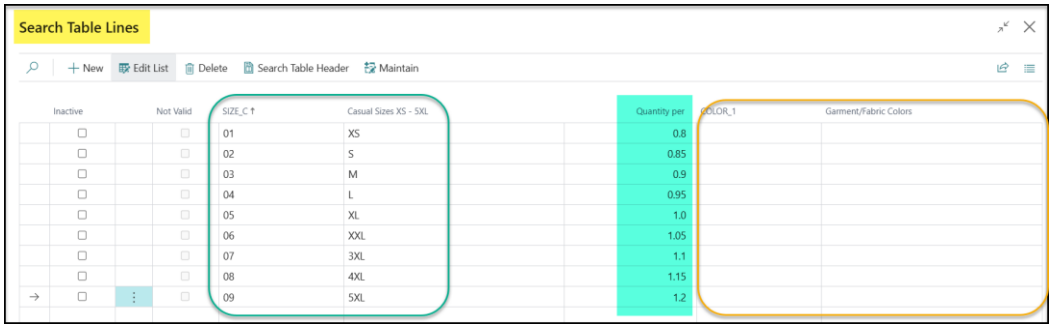
The Shirt comes in different Colors and Sizes. The Shirt consists of two Raw Materials: a Fabric and a Button: the bigger the Size, the more Fabric needs to be used. The Color of the Fabric is identical to the Color of the Shirt. So, the Color of the Fabric could be inherited from the Color of the Shirt.
The BOM looks like this before entering the Configuration Rules (only focus on the Fabric).



Since we have a checkmark in **Consistence Inheritance** on *COLOR_1* in the VarDim Master of the Fabric (which is identical to the Color variant in the Shirt), we only need a Search Table managing the Quantity depending on the Size. So, we must place a *checkmark* in the Field **Quantity per in Search Table**. In the **Search Expression** we only need to select *SIZE_C*.



If you do so, the **Search Table Lines** will look like the screenshot below in which we see only the Sizes in the Input side of the Search Table. We only must enter the appropriate **Quantity per**. The columns for the Fabric Color need to stay *empty*, because these will be inherited from the Shirt Color.



Note 1
The **Consistent Inheritance** only works for identical VarDim Types on the main Product and its Component. Besides that, it also requires that the **VarDim Type Placement in Item No.** should be there on the same **VarDim Type** in both **Masters** (Finished Product and Component).

Note 2
There is a hierarchy: The system looks first if there is **Consistent Inheritance** and later to an eventual Search Table. This means that the Value in the **Search Table Lines** overwrites the inherited Value.

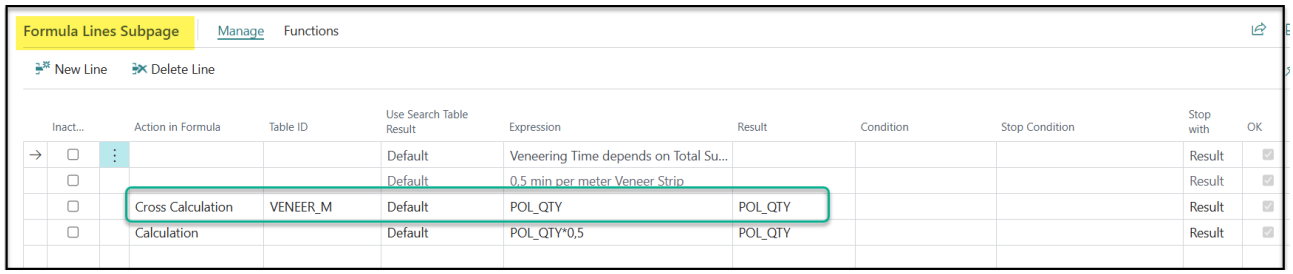
Note 3
Apparently, there are alternative ways to set up Configuration Rules for Components. However, the best advice is to be consistent throughout all BOM Component Lines.

Cross Calculation

Introduction

With a Formula on a **BOM Component Line** it is possible to extract Input Data from other BOM Component Lines. This can be done with a **Cross Calculation** Feature. **Cross Calculation** is typically used to calculate Quantities that are determined by other Components in the BOM.

A **Cross Calculation** is initialized by attaching a **Formula Line** with the **Action in Formula** = *Cross Calculation* to the **BOM Component Line** that must be updated. The basis for the Calculation must be entered in Table ***6037207*** **Formula Cross Calc Setup**. The **Formula Cross Calculation** that is used has its own unique Code and is entered in the Field **Table ID**.



The **Cross Calculation** is done when the production BOM is generated by running through all **BOM Component Lines** within the filters and for each Line executes the **Expression** that is entered in the Field **Expression** in the **Formula Line**. The calculated Value is accumulated and, in the end, becomes the basis for the update of the Field **Result** on the Formula Line**.**

In the next paragraph the Fields on the Cross Calculation are discussed.

Cross Calculation Card

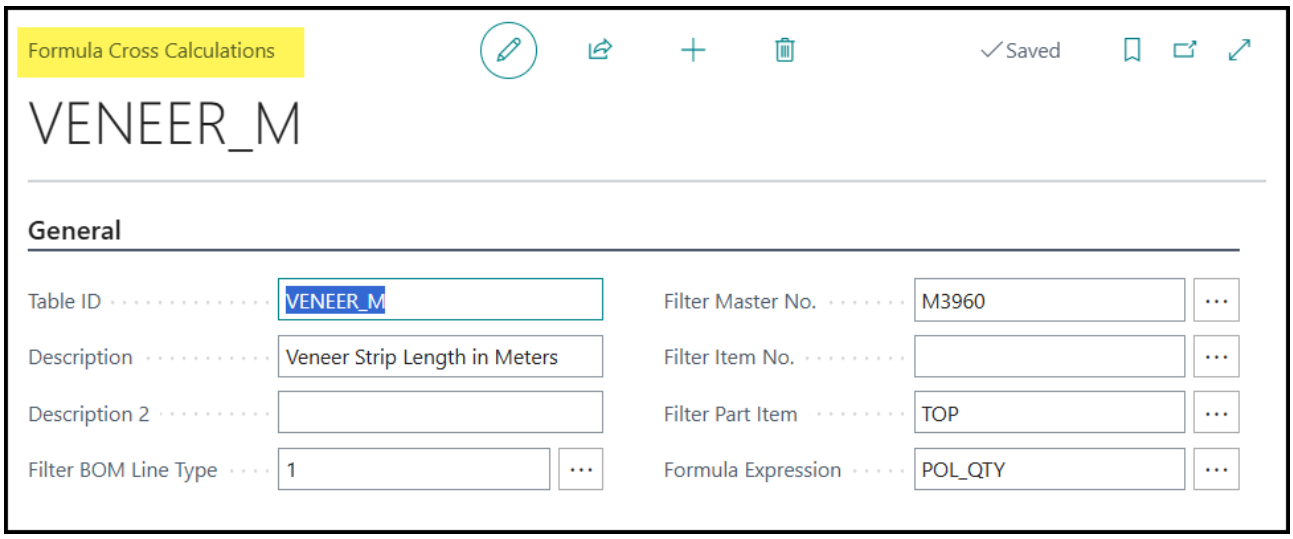


Table ID + Description + Description 2

In the Field **Table ID**, we enter the Code that represents the Name of the **Formula Cross Calculation.**
In the Field **Description** you can add a Text that describes the Cross Calculation.
If some additional Text is required, it can be placed in Field **Description 2**.

Filter BOM Line Type

The Field **Filter BOM Line Type** makes it possible to Filter the **Production Order Lines** on a **Type** for which you want to execute the **Formula Expression**.
You can choose between the following options: (Written as an Integer on the page):

"Blank"	No filters applied
Item	Filter the **Production Lines** on **Type** *Item*, shown as **1**
Machine Setup	Filter the **Production Lines** on **Type** *Setup*, shown as **2**
Operation	Filter the **Production Lines** on **Type** *Operation*, shown as **3**

Filter Master/Item/ Part Item No.

In the Field **Filter Master No.**, you can enter criteria that filter the **Production Order Lines** on the Field **Master No**. Note that this is not the **Item No.**, although an Item of **Type** *Master Item* is identical to its Master No. If you want that, you will have to enter filter criteria in the Field **Filter Item No.** We can also set a Filter on **Filter Part Item** that filter on the Field **Part Item No.** of the Production Order Line.

The result of the right filtering is that one or more Production Order Line(s) should be in the selection.

Note
Filters are executed in the following order:
Filter BOM Line Type, **Filter Master No.**, **Filter Item No.**, **and Filter Part Item.**

Formula Expression

In this Field, you create the **Expression** you want to execute on all the **Production Order Lines** within the filters set in the Filter Fields. The Value of the one Variable of **Type** *Decimal* is the accumulated Value of all Variables of these Production Order Lines. So according to the screenshot above, it is the sum of the Field **Quantity** in the Production Order Lines. Other mathematical calculations can be applied.

The **Formula Expression** is automatically inserted in the Field **Expression** on the **Formula Line** where the **Cross Calculation** is called from. It can be maintained/changed from both places.
Using the **AssistEdit** button (or press **Shift+Alt+Arrow Down**) behind the Field can help you in creating the Formula Expression by opening the **Formula Expression AssistEdit**.

The outcome of executing the **Formula Expression** will be placed in the Variable in Field **Result** on the **Formula Line**.

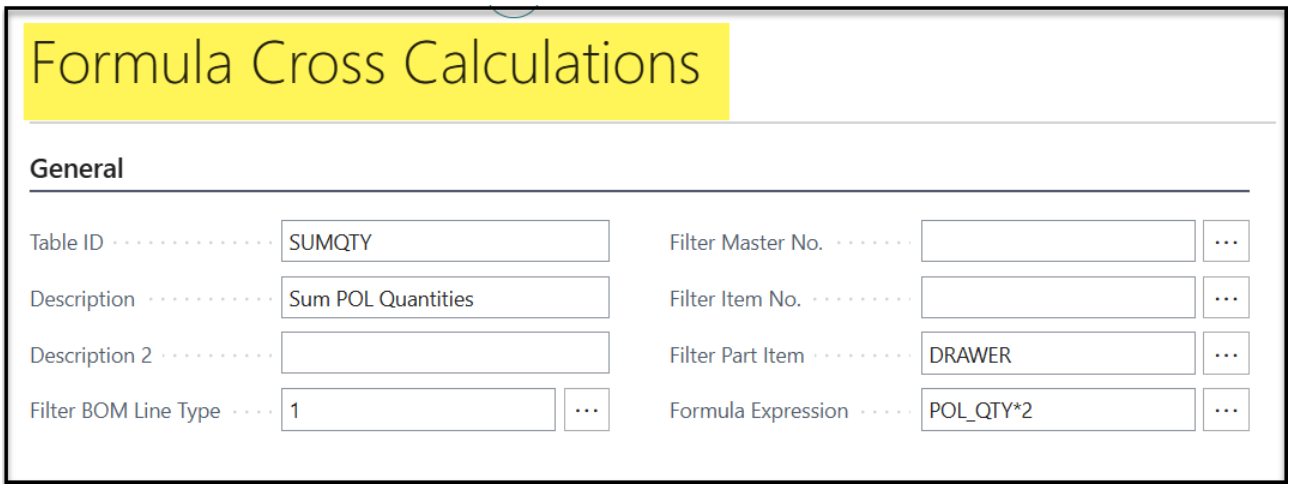
How to Use Cross Calculation

Cross Calculations can be used to achieve different goals.

The first one is mentioned in the paragraph above. You can look for an accumulated Value for the *Decimal* Field found on the **Production Order Lines** within the **Cross Calculation** filters.

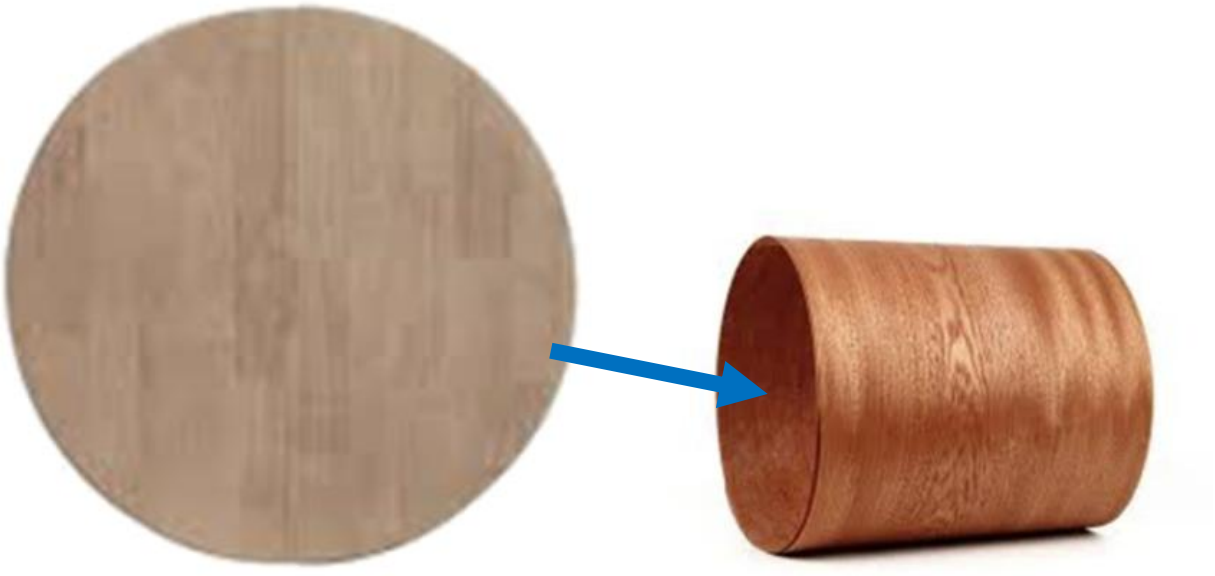
Case 1:
Let us assume that we have a Cabinet that may have different Types of Drawers (**Part Item No.** = *DRAWER*) on the Production Order Lines. Every Drawer has two Handles. With the help of the Cross Calculation Feature we want to calculate how many Handles we need to have in the Production Order Line of the Handles.

So, in that case, we make a **Formula Cross Calculation** based on Items with a filter on **Part Item No.** = *DRAWER*. This will result in a selection of Production Order Lines for which we take the Production Order Line **Quantity per** (*POL_QTY*) and accumulate them.
The sum of these Quantities will be multiplied by *2* according to the **Formula Expression**. The result will be placed into the **Result** Field on the **Formula Line** from which the Cross Calculation is called.



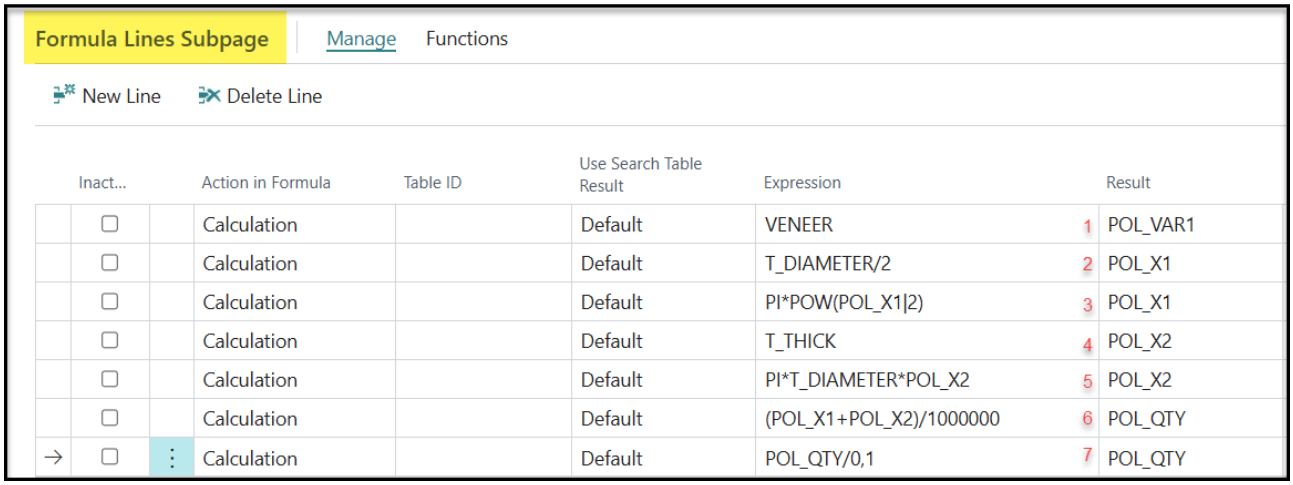
Cross Calculation can also be used for complex calculations that you will only do once.

Case 2:
Let us assume that we have a Round Tabletop that needs to be covered with Veneer. In order to do so we need to calculate the surface of the Tabletop to be veneered including the edges. Based on the Diameter and the Thickness we can calculate the total surface in Formula Lines using the **PI** function.
The Veneer comes in roles of a certain Width (or Thickness), so if we know the total Surface we can calculate that into the total Length of the Veneer Strips we need.



However, the Veneer Strip is not the only component for which the Quantity depends on the surface of the Tabletop. This is the case with the Veneering Operation as well. The time used for glueing the Veneer to the Tabletop can be calculated from the Length of Veneer Strip that is required.
If we do not want to have the same calculation twice in two different Formulas, we can calculate it in one and use the Cross Calculation Feature to get this Value also for the other BOM Component Line.

So, to calculate the Length (m) of the Veneer Strip (*10 cm wide*) we need the *Diameter* (*mm*) and *Thickness* (*mm*) of the Table Top and we have the following **Formula** on the **BOM Component Line** of the Veneer Strip:

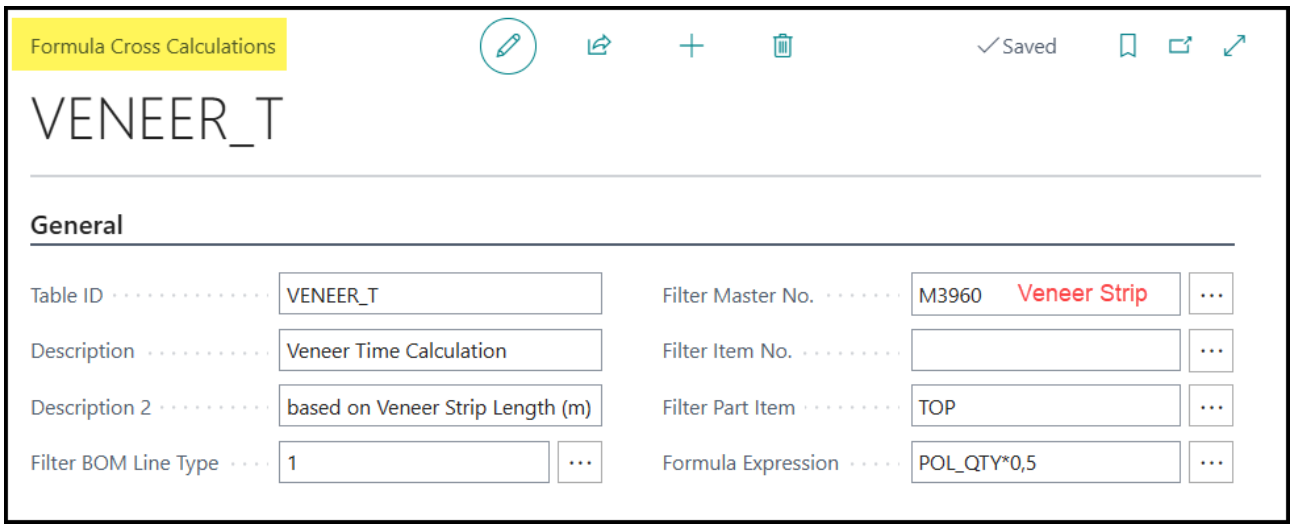


	Inact...		Action in Formula	Table ID	Use Search Table Result	Expression		Result
	☐		Calculation		Default	VENEER	1	POL_VAR1
	☐		Calculation		Default	T_DIAMETER/2	2	POL_X1
	☐		Calculation		Default	PI*POW(POL_X1|2)	3	POL_X1
	☐		Calculation		Default	T_THICK	4	POL_X2
	☐		Calculation		Default	PI*T_DIAMETER*POL_X2	5	POL_X2
	☐		Calculation		Default	(POL_X1+POL_X2)/1000000	6	POL_QTY
→	☐	⋮	Calculation		Default	POL_QTY/0,1	7	POL_QTY

Explanation of the *7* Formula Lines

1. Inherit the choice of Veneer on the Tabletop to the Veneer Strip.
2. Convert the Diameter (mm) of the Tabletop into the Radius (mm)
3. Calculate the surface of the Tabletop (mm2)
4. Extract the Thickness (mm) of the Tabletop.
5. Calculate the surface to the edge of the Tabletop (mm2)
6. Totalize the two surfaces and convert that into m2.
7. Calculate the Quantity per (m) of the Veneer Strip , which is 10 cm (=0,1m) wide.

The Veneering speed is *2 m/min (=0.5 min/m)*. So, if we know the Length of the required Veneer Strip, we can calculate the Time needed for the Veneering Operation.
We can do the whole calculation again as above, or we use the **Cross Calculation** to use the Quantity that has already been calculated for the Veneer Strip.
For the latter we need to set up the following **Cross Calculation**:

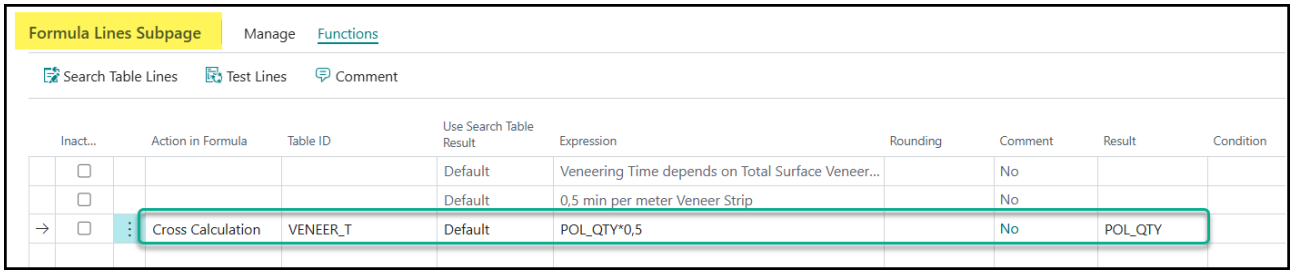


According to this **Cross Calculation** we filter the **Production Order Lines** on **Items** originated from **Master** *M3960* (Veneer Strip) used for the **Part Item No.** *TOP* – in case we also have Veneer used for the Table Base, which we do not want to select. If we find the calculated Length of the Veneer Strip

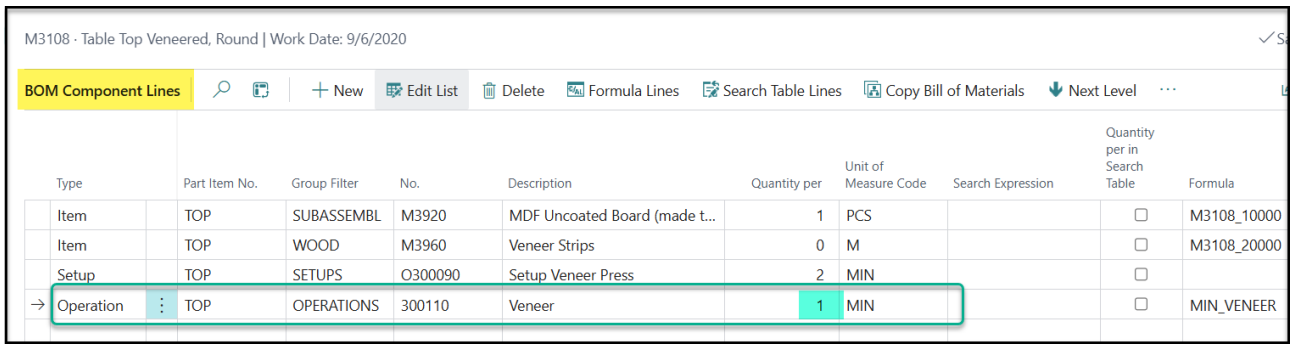
(*POL_QTY*) in that Production Order Line, we use that Quantity in the Search Expression to calculate the required estimation time (*POL_QTY x 0,5*) that we place into the Field **Result** on the **Formula Line**.

Note
Keep in mind that the Decimal separator must comply with the **Formula Decimal Separator** in the Basic Setup!!

The **Formula Line** on the Operation *Veneering* would be like this:



Note
In **BOM Component Lines** we see the **Quantity per**, which will be placed on the **Production Order Line**.
Often, we can leave the **Quantity per** '*zero,*' if we calculate the **Quantity per** via Formula Calculation.
However, that is <u>not</u> the case, if the basis of this calculation is a **Cross Calculation**.
In those cases, we need to enter a **Quantity per** on the **BOM Component Line**.
The Formula will overwrite this Quantity. An example is shown below.



Analyzing BOM Structure

Introduction

Before you start creating BOMs and entering Formulas on the BOM Components Lines, it would be wise to analyze the BOM Structure first by following the steps below:

1. Determine the Variations of the Main Product.
2. Determine the 1st Level (= Start Level) of the BOM Component Lines: Materials and Operations
3. Establish whether and how the Variations on the BOM Component Lines depend on the Variations of the Main Product.
4. If none of the 1st Level BOM Component Lines are Sub-assemblies, the analyzing will stop here. Go to step 9!
5. Determine the BOM Component Lines for these Sub-assemblies: Materials and Operations
   Do this for every Sub-assembly individually.
6. Establish whether a Component is before or after the Customer Order Decoupling Point. (*CODP*, to be explained later in this chapter)
7. Establish whether and how the Variations on these BOM Component Lines depend on the Variations of the Overlying BOM Level or the Start Level. (The latter only makes sense with Components before the CODP.)
8. If you are finished with a Component Level and some of these Components are Sub-assemblies, you can analyze the next Level of Components for these Sub-assemblies:
   go back to step 5.
   Otherwise the analyzing stops here. In that case, go to step 9!
9. Establish in which production departments the Operations take place. (This will not be discussed in this document. For this we refer to ***White Paper - TRIMIT Production*** (under construction)

Please note that the advice is to analyze Level by Level. This is the same order advised for entering Formula Lines on BOM Component Lines.
In the next paragraphs we go into more detail regarding the analysis of the BOM Structure.

Variations on Masters: Main and Components

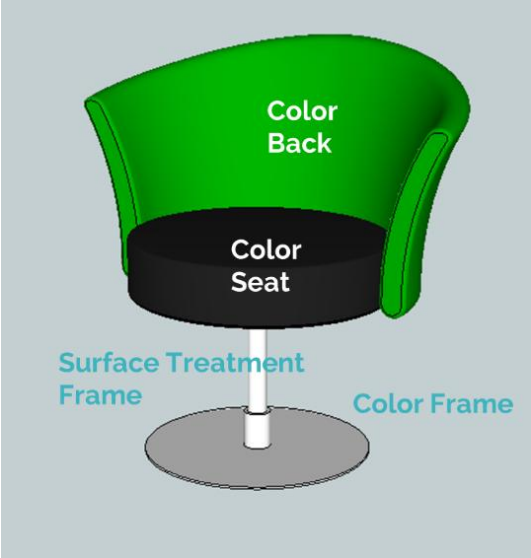
In the Introduction it is stated that it is important to determine what the Variations are of both the Main Product and its Components. The variations could be:

- *Variants* (table with Options)
- *Measurements* (TRIMIT Dimensions with a specific range within a Min. and Max. Value)
- *The presence or absence of particular parts on the Products to be produced.*
- *The positions of particular parts*
- *The Quantity of particular parts*

To keep the Formulas as simple as possible, the advice is to have as much as possible to use the same VarDim Types on both the Main Product and its Components. This often means, that renumbering of the Items might be required when implementing TRIMIT in a project.

Example:
Let us assume that we create a new **Master** called *Chair Hercules* (see picture below).
We establish that this Product has four different Variations:

- Color of the Back: selection of Colors (**VarDim Type** of **Type** Variant)
- Color of the Seat: selection of Colors (**VarDim Type** of **Type** Variant)
- Surface Treatment of the Frame: *Polished* or *Paint (Lacquer)* (**VarDim Type** of **Type** Variant)
- Color Frame: None (if *Polished*) or selection of RAL Colors (when *Paint (Lacquer)*).
  - (Also, VarDim Type of **Type** Variant)



Component Levels + Variance Dependencies

Next thing is determining the first Component Level. This can include both Materials as Operations.
The Variances of First Level Components can depend on the Variances of the Main Product.
If that is the case, you need to put the Components in a Master BOM with Configuration Rules that establish the dependencies between the Main Product and its Components.

When all Components on the first Level do not depend on the Variances on the Main Product, you can choose to have the Components in a (fixed) Item BOM instead.

On our Hercules Chair we have the following Components on the first Level:

- Back -> depending on the chosen Color for the Back.
- Seat -> depending on the chosen Color for the Seat
- Frame with Surface Treatment. -> depending on the chosen Surface Treatment/Frame Color

For these Components, it is useful to have the identical VarDim Type on the Components as on the Main Product. In that case, quite simple inheritance Formulas can be used.

- Chair Assembly -> Operation that does not depend on any Variance.

This Operation does not need any Formula that manipulates the Production Order Line.

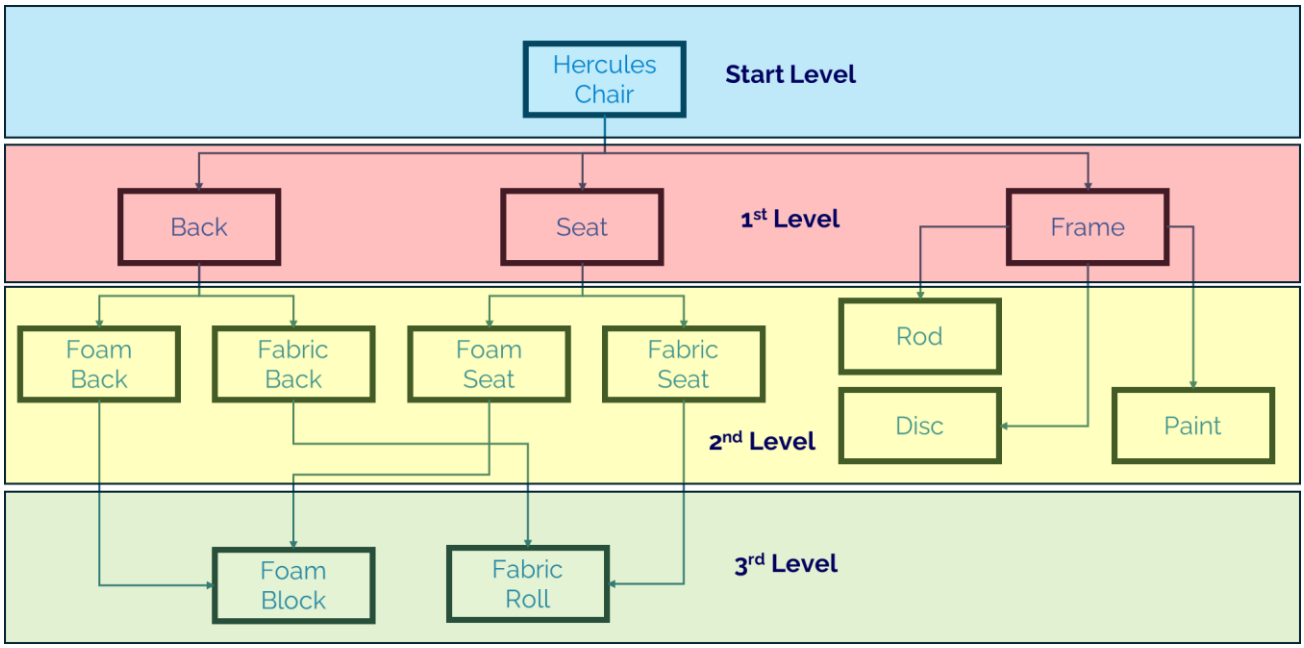
If the 1st Level is done, we can start analyzing the 2nd BOM Level, then the 3rd, 4th and so on until the whole BOM Structure is analyzed.

Our Hercules Chair has a 2nd BOM Level for each Component.

- The Back and Seat parts are manufactured from *Foam* and *Fabric* pieces that need to be sewn together.
- The Frame is an assembly of two parts: *Rod* and *Disc*.
  After the assembly the frame gets its Surface Treatment, so *Polish* or *Paint* operation required.
  When the frame is *Painted*, we need a Lacquer with a specific RAL Color.

Our Hercules Chair also has a 3rd BOM Level because the Foam pieces need to be cut from a Foam Block and the Fabric pieces from a Fabric Roll. The Foam Pieces do not depend on Variances from Overlying Level, but they come into two different Shapes for the Back part and the Seat part.
The Color of the Fabric depends on the chosen Color of the Back and the Seat. The Quantity of the Fabric to be consumed depends on the usage of Fabric on the Back and Seat part.
The Rods, Discs, Foam Blocks and Fabric Rolls are purchased. So, they do not need a BOM Level.

Now we have analyzed the whole BOM Structure of the Hercules Chair. For the Material Components it is schematically shown below.



Customer Order Decoupling Point

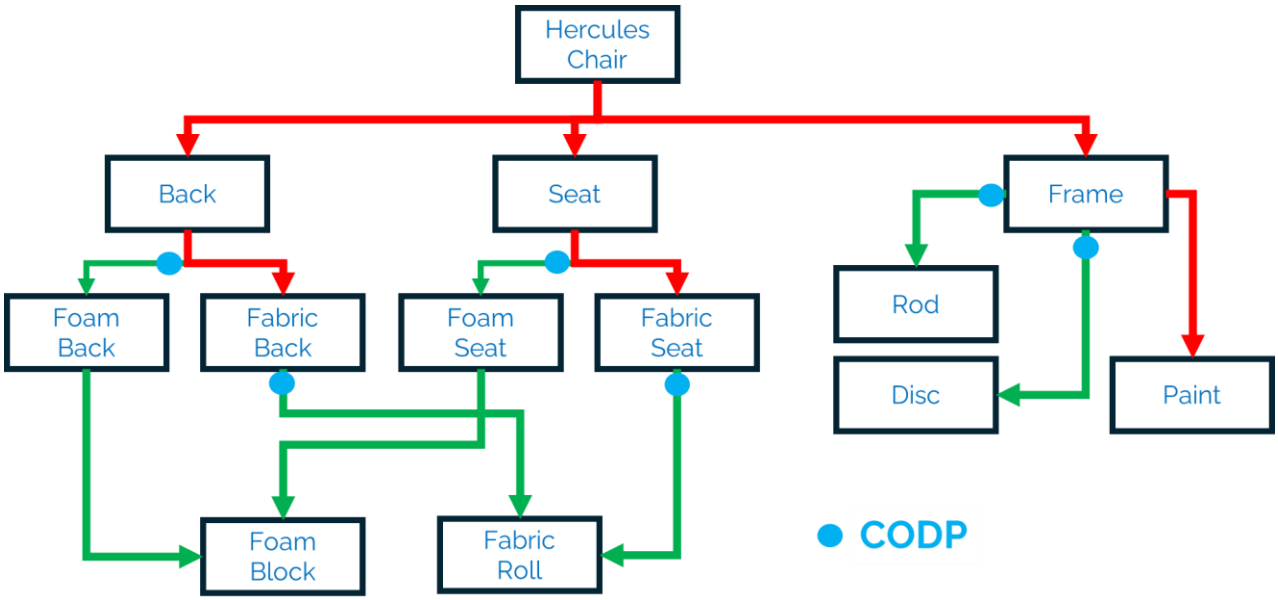
In the previous paragraphs we emphasize that it is important to analyze the BOM Structure Level by Level. With Formulas we can calculate the Components from the Variances on the Overlying Level. However, in some cases, it is also possible to refer to the Variances on the Start Level. To do so, we need to establish the **Customer Order Decoupling Points**.

The Customer Order Decoupling Point **(CODP)** is a point in the BOM Structure, where the Product becomes untied to a specific Customer. Or in other words: until where in the BOM Structure are the Products produced/purchased related to the Start Level.

In TRIMIT it means that all *Item* Components before CODP must have a **Reordering Policy** *Order.*
All other *Item* Components in the BOM Structure mostly have **Reordering Policy** *Inventory.*
In case the **Reordering Policy** is *Order* the next Level Order is related to it.
The relationship is 1:1 and there is a hard link.
In case the **Reordering Policy** is *Inventory* there is no relationship between the Order and its underlying Orders. In those cases, the demand for these Components needs to be calculated with the MRP Calculation. Often the demand of the underlying Components is aggregated based on the MRP settings on the Master/Item Card.

Let us have a look at the Hercules Chair again. We found out that the Chair itself is manufactured based on Order (Make to Order= MTO).
This is also the case with the first Level Components: Back, Seat and Frame.
On the second Level we have established that the Fabric parts are also cut Make to Order, while the Foam Parts are manufactured on Inventory (Make to Stock = MTS). The Rod and Disc are also purchased on Inventory, while the Paint is purchased to Order.
The 4th level Components (Foam Blocks and Fabric Rolls) are both Purchased on Inventory.

Let us see how this works out for the Materials in our BOM Structure. We have indicated with the Blue Dots where the CODPs are.



So, this means that the Items before the Blue Dots must have **Reordering Policy** *Order* and the others commonly *Inventory*. So, the red arrows represent **Related Orders**, while the green arrows do not have a relationship with the Levels above and may need to be calculated via **MRP Calculation**.

This could be especially important for the **Formulas**, because for the **MTS Components** we can only refer in the **Formula Lines** to the **Variables** pointing at Fields of the **Overlying Level Item** or they should have a fixed **Quantity** or *Text Replacement*.
For the **MTO Products** we have the possibility to follow the red arrows to the **Start Level** and retrieve the Values from attributes on the **Start Level Item**. This is especially useful in cases when not all Variables required on the **Current Level** need Values that cannot be retrieved from the Overlying Level Item and are not fixed.

Putting BOM Structure in BC/TRIMIT

After we have established the BOM Structure, it is now time to enter it in BC/TRIMIT. The advice is to do this Level by Level. This means that a Level should be finished before you go to the next Level.

After a Component is entered on the BOM Component Line, create the Formula on this Component when necessary. Then test this Formula. The advice is to do this initially Component by Component.

Note
Creating and testing Formulas Component per Components and creating and testing the BOM Structure Level-by-Level, could be time consuming.
However, it is advised to follow this procedure until you have enough experience to create and test it in blocks.
However, eventual flaws are harder to find the longer you postponed the testing.
So only experienced Formula builders may deviate from the advised procedure.

In the next paragraph, you can see how the Formulas on the BOM Components look like for the Chair Hercules for all levels.

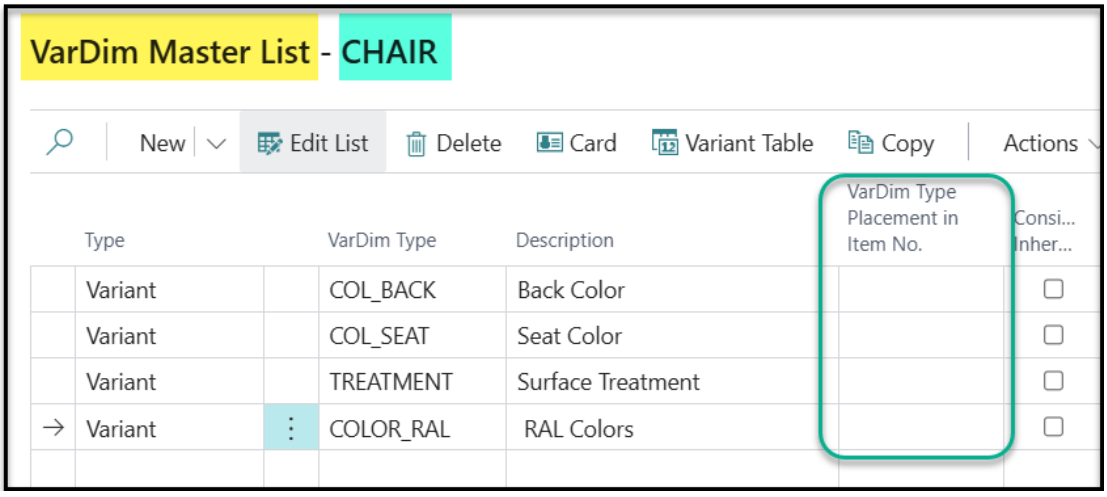
Formulas on BOM Components Chair Hercules

Introduction

In the previous paragraphs we have established the BOM Structure of the Chair Hercules. Now it is time to put the Components into BC/TRIMIT. We are going to do this Level by Level, starting with the Main Item. Per BOM Component Line the Formulas are created.

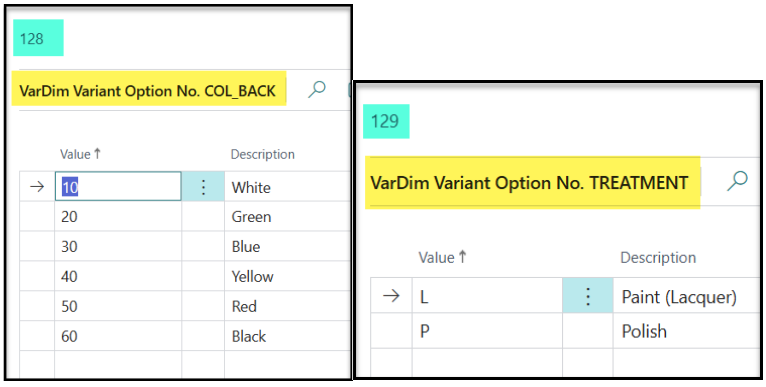
Master and VarDim Master Main Product

We have established the Variations of the Chair Hercules, so we create the **Master** Card *CHAIR* followed by its **VarDim Master**.
According to the Variations we described earlier in this chapter, the VarDim Master may look like this:



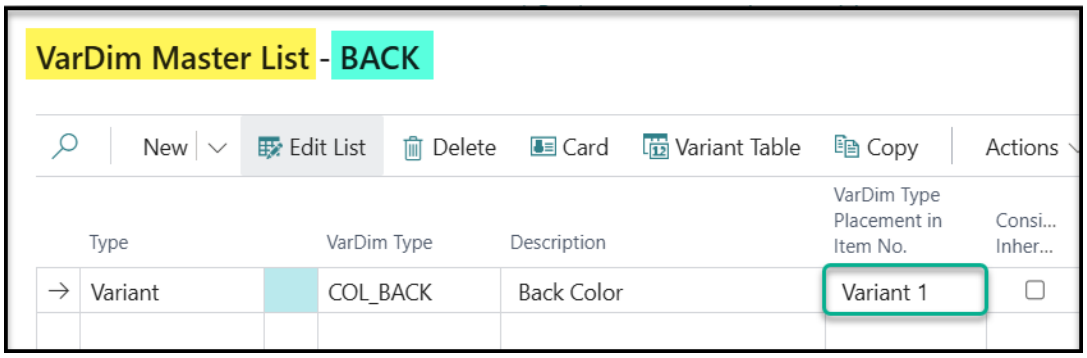
Note that the **VarDim Type Placement in Item No.** is empty for all **VarDim Types**.
Apparently, this is an MTO Product with a generic **Item No.** with **No. System** = *MMMMM* (**Master No.** *CHAIR* contains *5* digits). This means that regarding Configuration Rules you cannot work with **Consistent Inheritance** or with **Search Expression/Search Table**. So, you must work with TRIMIT **Formulas** when creating Configuration Rules for the Components of the Hercules Chair.

The **VarDim Types** in the **VarDim Master** are all **Type** *Variant*, which means that they have Option Tables. The Option Table for the *COL_SEAT* is identical to the Option Table for *COL_BACK* and refer therefore to the same **VarDim Variant** Number = *128*. The Option Tables are shown below:



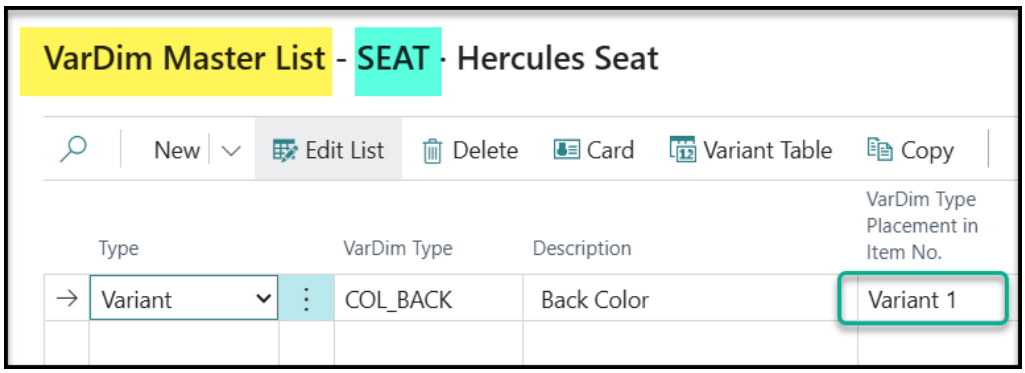
Masters and VarDim Masters Sub-assemblies - 1st Level

Now we want to create the Masters for the Components on the 1st BOM Level. These are the **Masters** of the Sub-assemblies *BACK*, *SEAT* and *FRAME*.
The *BACK* depends only on the Back Color, so it has the following **VarDim Master**:

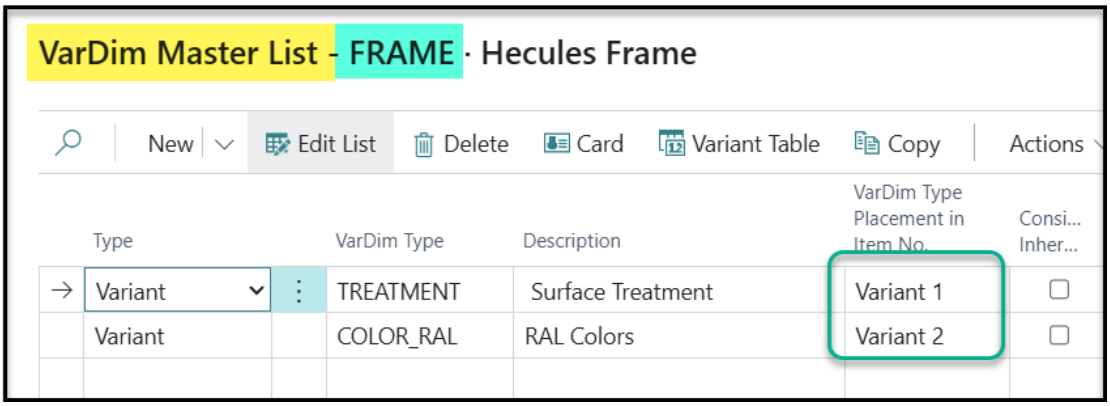


Note that in this case the **VarDim Type Placement in Item No.** entered.
So, the **No. System** should be *MMMMAA*, since **Master No.** has *4* digits and the **Value** of the Color Options has *2* digits.

The *SEAT* depends on the *COL_SEAT* chosen in the **VarDim Order** for the *CHAIR*.
So, the *SEAT* has **No. System** *MMMMAA* and has the following **VarDim Master**:



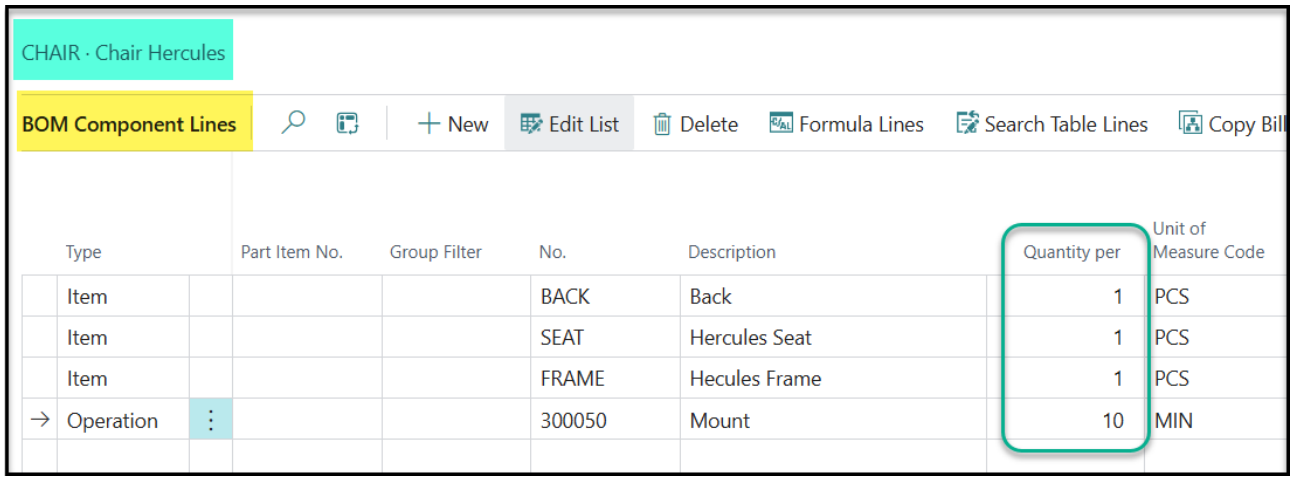
The *FRAME* depends on the Surface Treatment and in case this is Lacquering, there are different choices for RAL Colors of the Paint.
The *FRAME* has **No. System** *MMMMMABBBB* and the following **VarDim Master**:



Note that all Components on the 1st BOM Level are before the CODPs, so all the Master/Items have **Reordering Policy** *Order*.

BOM 1st level

Now we have created three Components on the 1st Level BOM.
So now we can place them into the **Master BOM** of the Hercules Chair.
We only need a **Quantity per** of *1* for all these Components.
We also add a Mounting Operation of *10* minutes. The BOM looks like this:

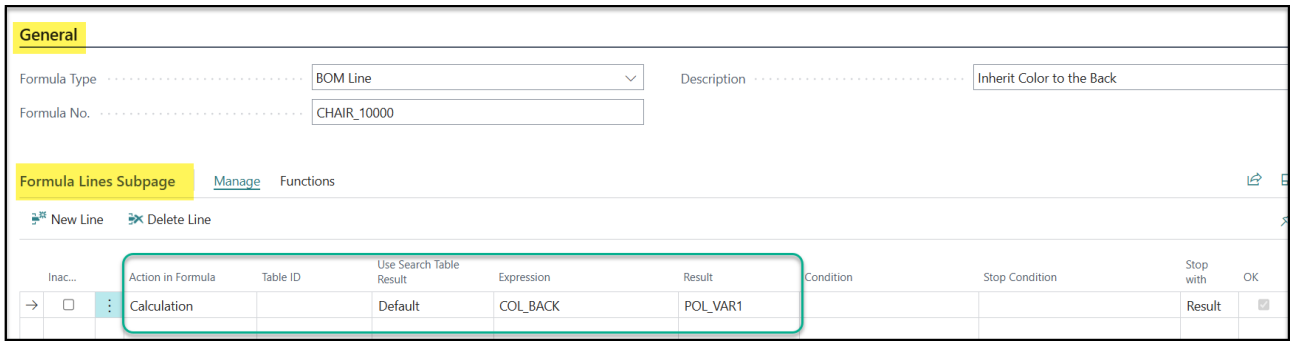


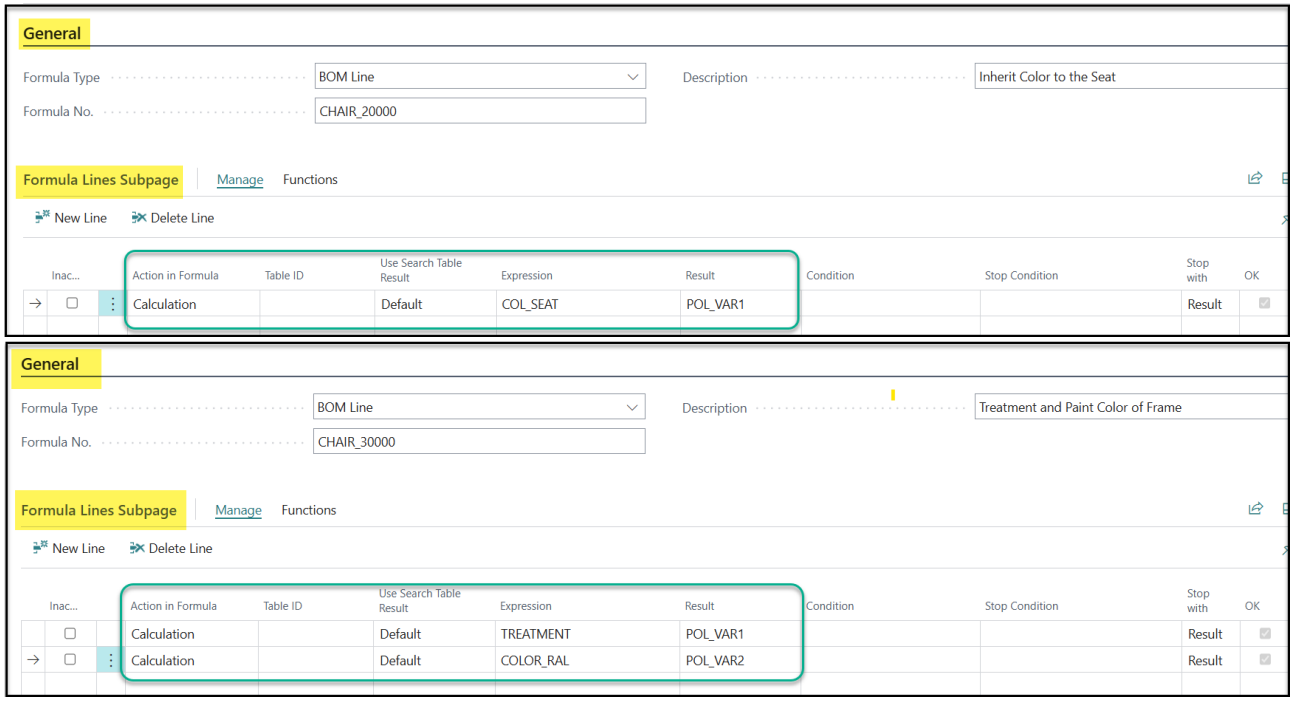
Formulas BOM Components 1st level

Per Component we need to create the Configuration Rules based on the dependencies on the Start Level Item. So, on the BACK, SEAT and FRAME we need to setup Formulas. The Mounting Operation does not depend on the Variances on the Start Level Item, so that BOM Component Line will not have a Formula.

The Formulas on the 1st Level Components are quite simple. They are all 'Inheritance' Formulas. That is why it is so useful to have the VarDim Types on the Components that are identical to the VarDim Types on the Overlying Level (or at least with an identical VarDim Option Table).

The Formulas for the three Components look like below:



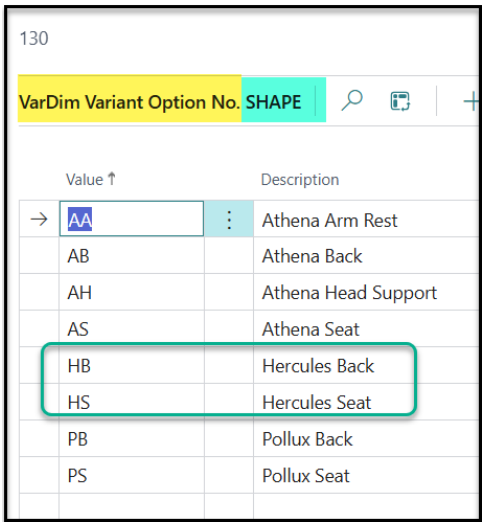


Note
As mentioned earlier; if you want to make the Formulas as simple as above, it may be necessary to renumber the Items within a new implementation. A simple data-migration of the Items will not suffice, because not only the Main Product but its Components as well need to be setup in a Master/Item Structure.

Now the focus will be on the 2nd Level Components. We will start with the *BACK* and *SEAT* followed by the Components of the FRAME.

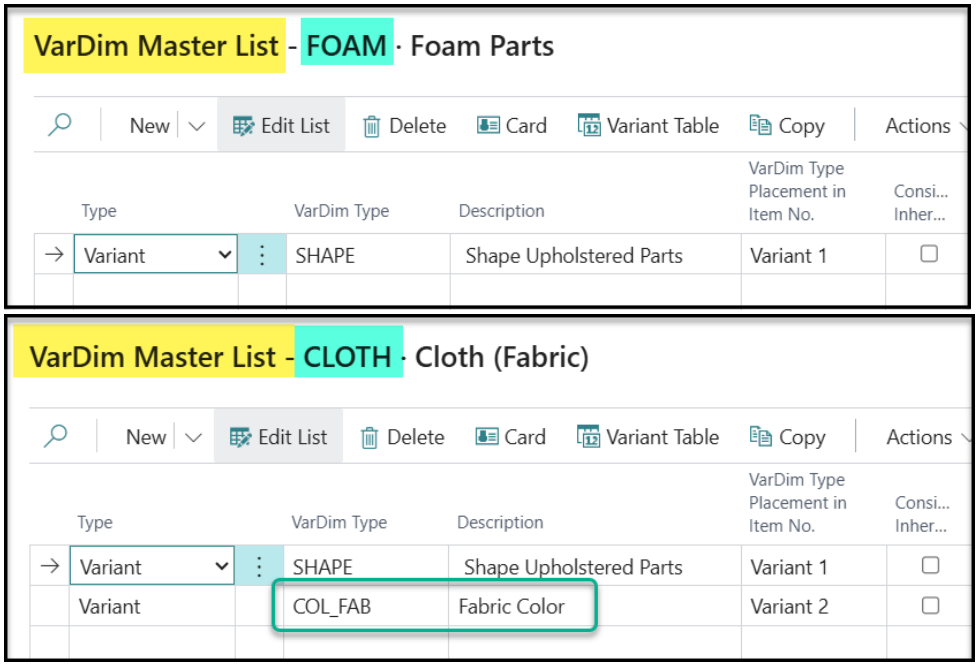
Masters and VarDim Masters Sub-assemblies BACK and SEAT

The Components on the BACK and the SEAT are the same; they consist of a piece of Cloth (Fabric) and a piece of Foam. However, the Shape of the Cloth and Foam Piece are different because the BACK and the SEAT have a different Shape. Since these Cloth and Foam Pieces belong to the same Master, we need to introduce a new VarDim Type to define the Shape of these Components *SHAPE (Shape Upholstered Components).*



Note
Since it is often the case that more Main Products within the Company have upholstered Components, you can use this VarDim Type for all these Components. Therefore, you see on the Option Table above also upholstered Components for another Chair (Pollux) and a Swivel Chair (Athena). But it can also contain upholstered parts of Sofas, Stools, etc.

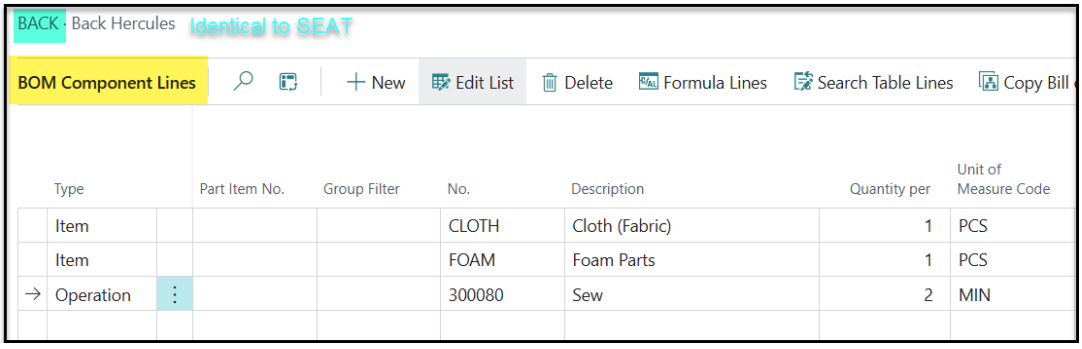
Apart from the Shape the piece of Cloth is also depending on the chosen Fabric Colors for *BACK* and *SEAT*. There is also another difference between the two Components.
If you look at the BOM Structure, the Cloth is an *MTO* Product, and the Foam part is *MTS* based on their positions before or after the CODP.
However, both are Sub-assemblies and have a **No. System** that includes the **VarDim Values**: *MMMMAA* (for Foam) and *MMMMMAABB* (for Cloth). You can see both **VarDim Masters** below.



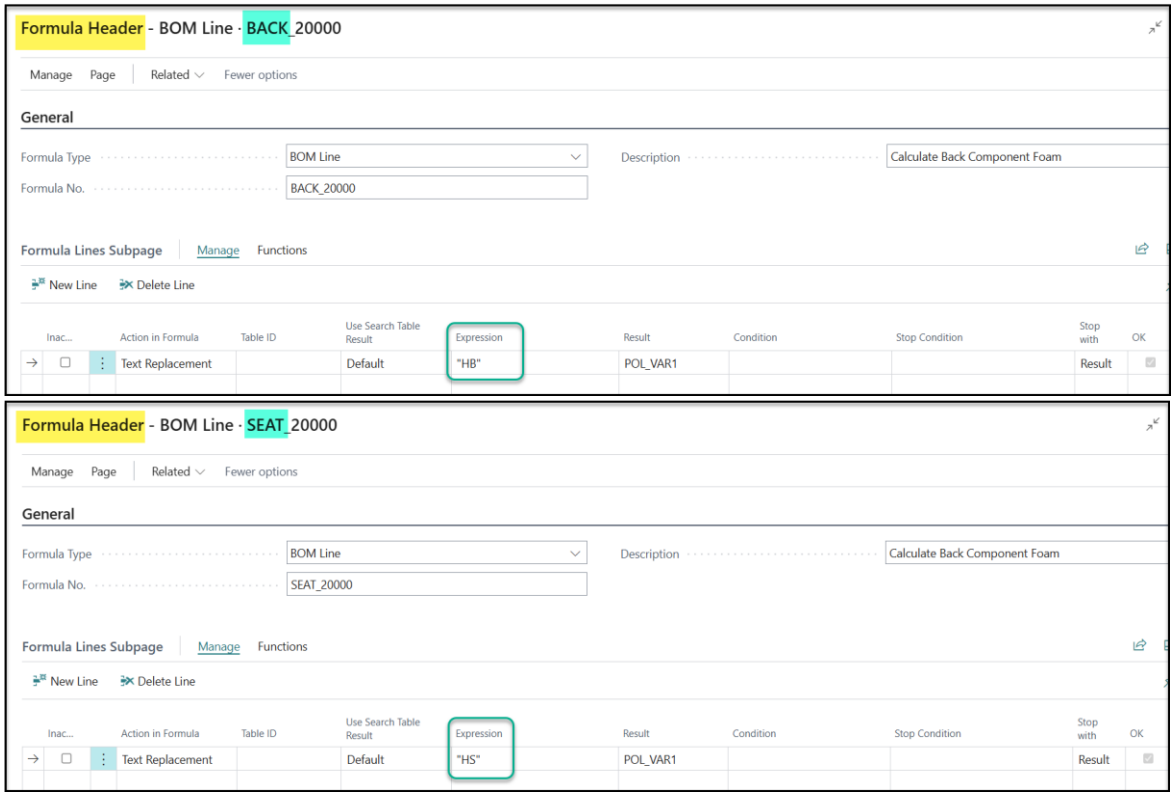
Note
In this example, we have also created a new **VarDim Type** for the Fabric Color (*COL_FAB*).
Since the Option Table is identical to *COL_BACK* and *COL_SEAT*, we make sure that this VarDim Type has the same **VarDim Variant** Number.

BOM and Component Formulas Sub-assemblies BACK and SEAT

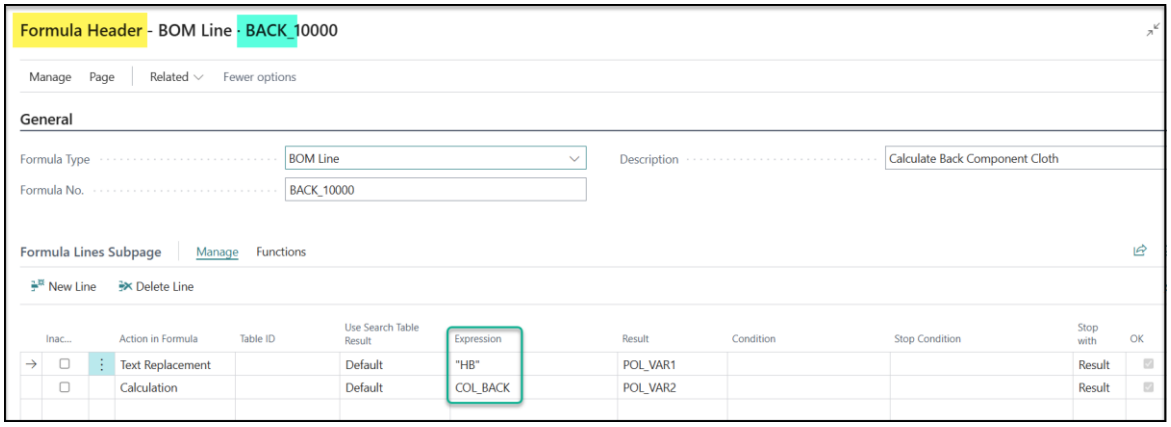
Now we have defined the Components of the BACK and the SEAT, we can create their BOMs. The BOM Components are identical: a piece of Cloth, a cut Shape of Foam and a Sewing Operation to bring the Cloth and Foam parts together. You can see an example below:

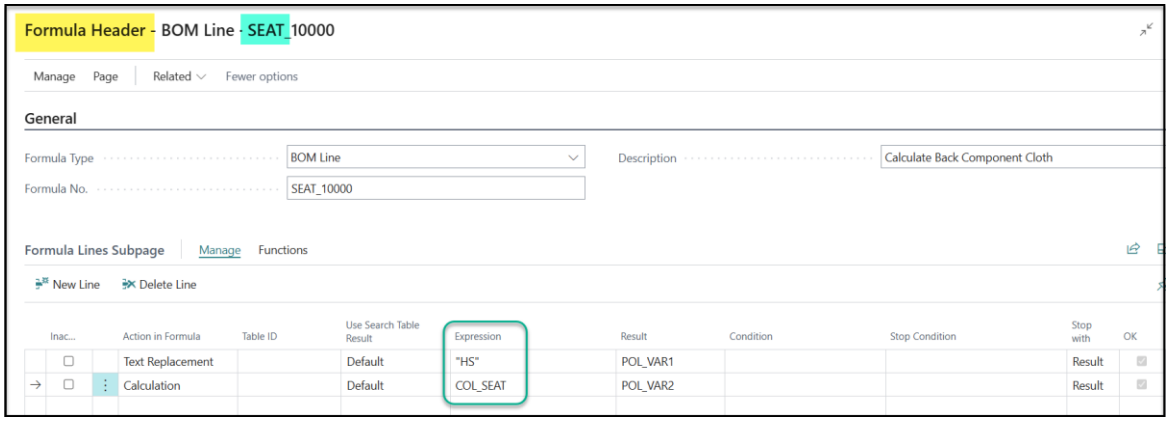


The differences are made in the Formulas of the Materials because these Masters have a **VarDim Type** *SHAPE* that cannot be retrieved from a higher Level.
So, this can be covered by a *Text Replacement* that defines the Value of the VarDim Type *SHAPE*.
In the screenshots below, you can see the two Formulas for the Foam, first of the BACK and SEAT, respectively.
Note that the **Expression** in the Formula Lines for the *Text Replacement* only difference between the two Formulas: *HB* (Back part Hercules Chair) and *HS* (Seat part of the Hercules Chair).



Next, we see below the two Formulas for the Cloth of the BACK and SEAT, respectively.
Also, in this case the Formula Line with the *Text Replacement* is different.
The Color of the Cloth is inherited from the Overlying Level: *COL_BACK* for the Back part and *COL_SEAT* for the Seat part of the Hercules Chair.



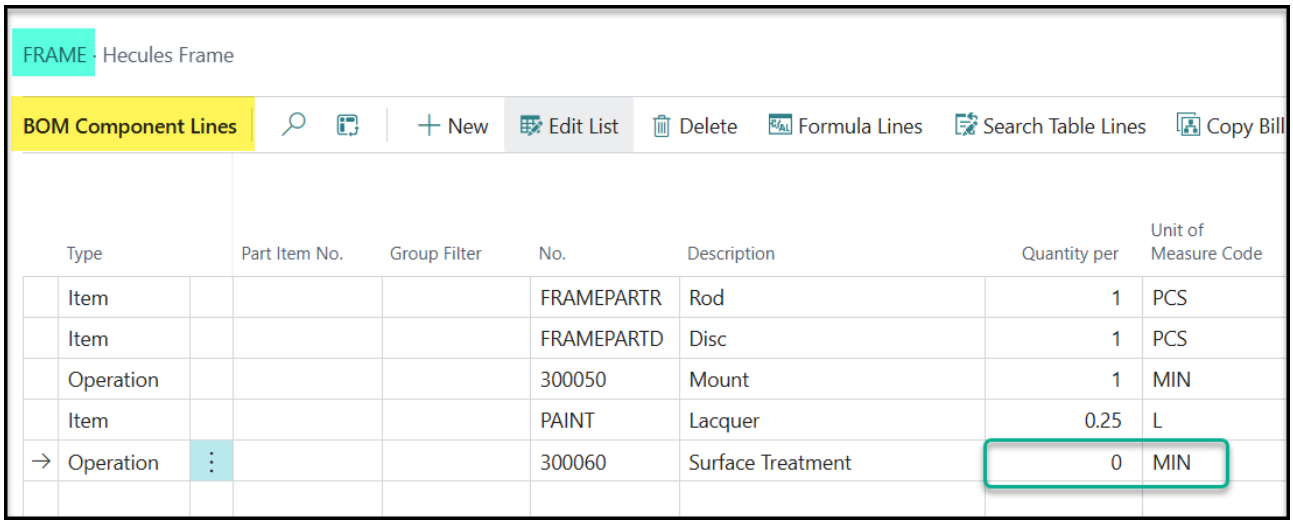


BOM and Component Formulas for Sub-assembly FRAME

According to the BOM Structure, the Frame consists of at least two Materials: *Rod* and *Disc*.
Both are MTS Items since they are after the CODP. These two Components need to be assembled.
If the Sub-assembly needs to be **Lacquered**, then we also need a third Material (*Paint*), for which we need a RAL Color. Because there are so many Colors, we purchase the Paint MTO.
We also need a **Surface Treatment**. Depending on whether the Frame is *Polished* or *Paint (Lacquered)*, there is a different Run Time for the **Operation** *Surface Treatment*.

The *Rod* and the *Disc* are both Items from the same **Master** called *FRAMEPART*. We need to create a **VarDim Type** of **Type** *Variant* with Options: *D* for *Disc* and *R* for *d*. After we create all Frame Part Items (and changed the Description), we can select both Items on the BOM Component Lines.

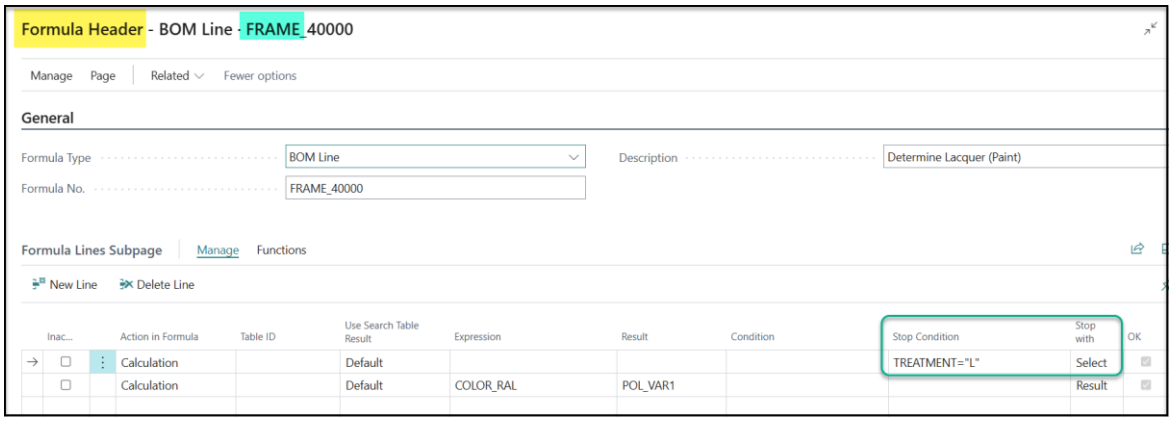
The **BOM** for the Frame will be as below. Note that the **Quantity per** is *0* for the Surface Treatment, because this is calculated by a Formula.



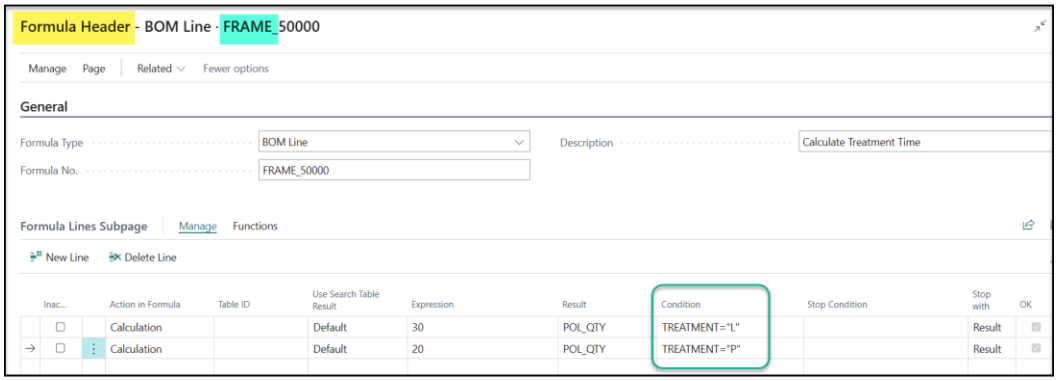
Type	Part Item No.	Group Filter	No.	Description	Quantity per	Unit of Measure Code
Item			FRAMEPARTR	Rod	1	PCS
Item			FRAMEPARTD	Disc	1	PCS
Operation			300050	Mount	1	MIN
Item			PAINT	Lacquer	0.25	L
→ Operation			300060	Surface Treatment	0	MIN

The *Rod* and the *Disc* are '*flat*' **Items**. No Formulas are needed. That is also the case for the Mounting Operation. However, the Paint (Lacquer) needs to inherit the RAL Color from the **Overlying Level**. We also need to have a **Stop Condition** that determines whether the Paint (Lacquer) is in.

For the Paint (Lacquer), the Formula will look like this:

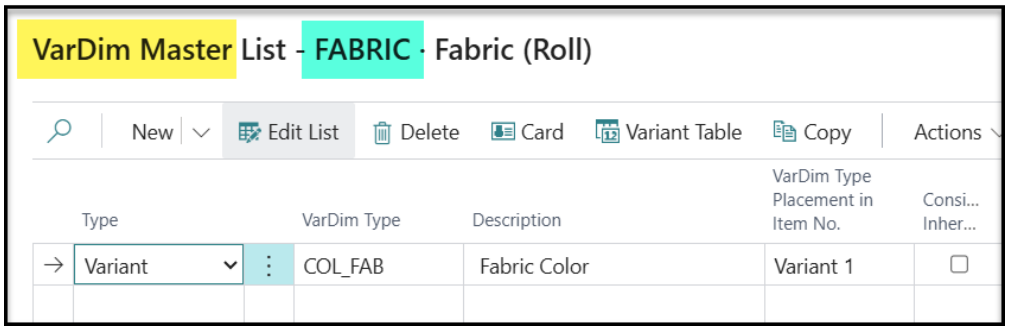


We also need to calculate the Run Time (*POL_QTY*) for the Operation *Surface Treatment*.
Let us assume that it takes *20* minutes for *Polishing* and 30 minutes for *Paint (Lacquer)*.
This is solved with a Formula on the Surface Treatment with Conditions. The Formula looks like this:



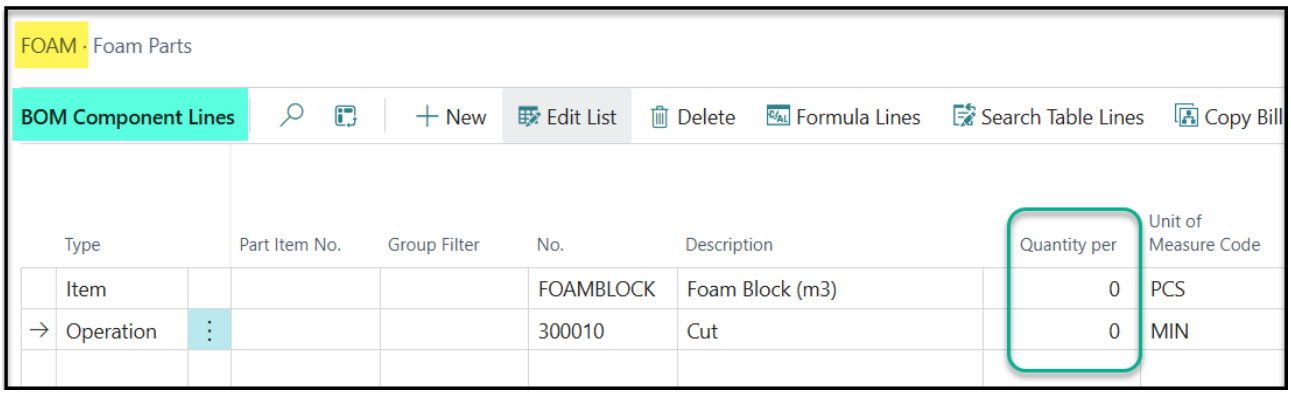
BOM and Component Formulas for 3rd Level items

Only the Foam Piece and the Cloth (= Fabric part) have a **3rd Level BOM** with *Purchase* Components *Foam Block* and *Fabric* (meters on a Roll) respectively. These Components are *MTS* (after the CODP).
The *Foam Block* does not have any Variants and there is no VarDim Master.
The *Fabric*, however, comes in Colors (= *COL_FAB*). The **VarDim Master** of the Fabric is shown below:

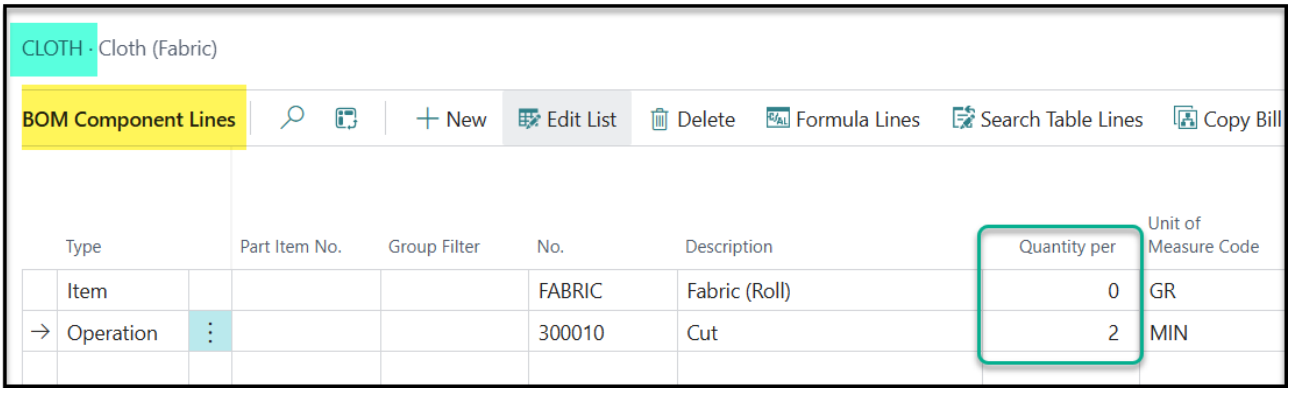


In the **BOM** of the Foam Piece, we have an **Operation** that cuts from a Foam Block the different Foam pieces. The Consumption of the Foam and the Cutting Time depends on the Shape of the Foam Piece. Therefore, for both Components the **Quantity per** is *0*; they will be calculated based on TRIMIT Formulas.

Below we see a screenshot of the **Master BOM** of the Foam parts.

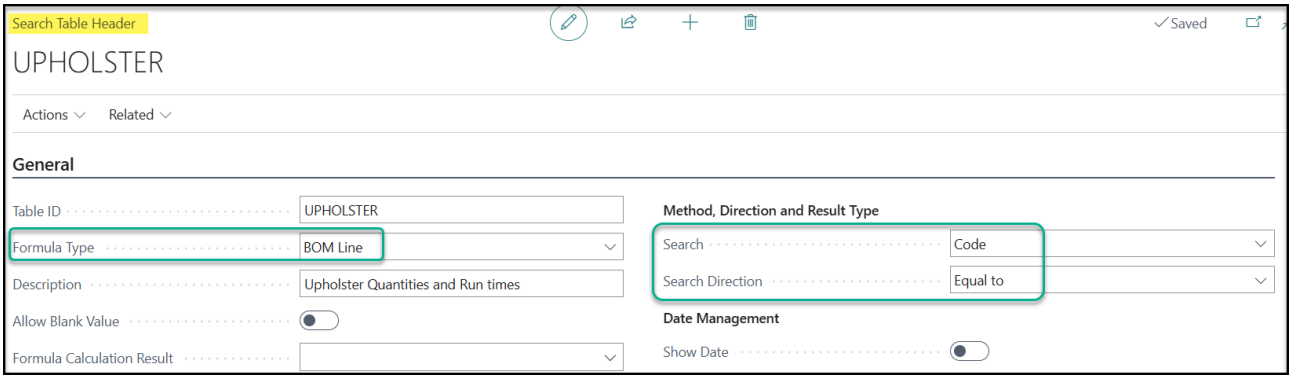


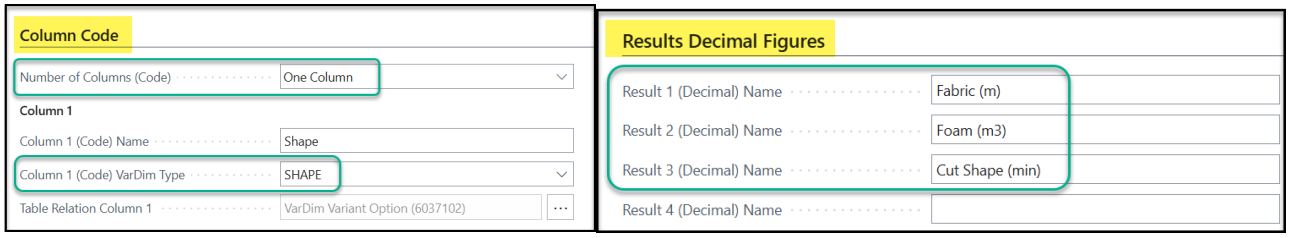
In the **BOM** of the Cloth, we have an **Operation** that Cuts from a Fabric Roll the different Cloth pieces.
The Consumption of the Fabric depends on the Shape of the Foam Piece.
Therefore, the **Quantity per** is *0*; it will be calculated based on TRIMIT Formulas.
That is also the case for the inheritance of the Fabric Color.
The Cutting Time is estimated to be *2* minutes for all Cloth pieces, and this is independent from the Shape.
Below we see a screenshot of the Master BOM of the Cloth parts.



Now it is time to build the Formulas for both BOMs for each Component Line (except **Operation** *Cut* for the Cloth). For the Fabric we need to inherit the Fabric Color. The Run Time for cutting the Foam parts and the Consumption of the Foam from the Block and the Fabric from the Roll depend on the Shape (**VarDim Type** *SHAPE*). This can easily be done via Conditions.

However, remember that we may have other Shapes for other Masters! It would be better to set up a **Search Table** in which the Run Time and **Quantity per** for the Foam and the **Quantity per** for the Fabric can be defined. This way all Data can be maintained in this **Search Table** that can be used in different Formula Lines. The **Search Table Header** will be like this:





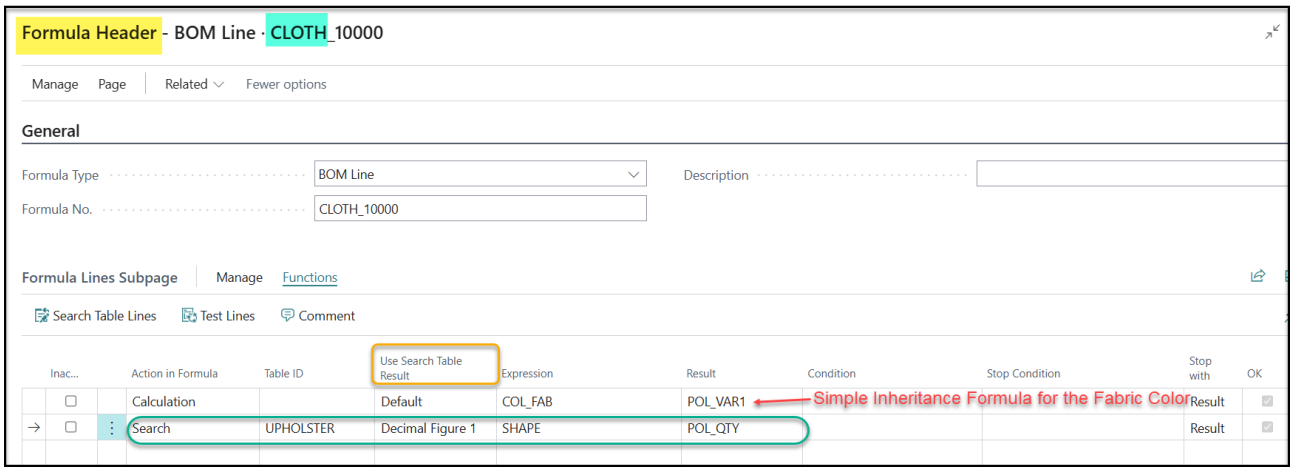
Note that we have one Input Variable in the **Column Code**: *SHAPE* and three Output columns of Type *Decimal Figure* in the **Results Decimal Figure**.

After we have set up the Search Table *UPHOLSTER* correctly, we can enter the appropriate numbers in the Output columns. An example is shown below.

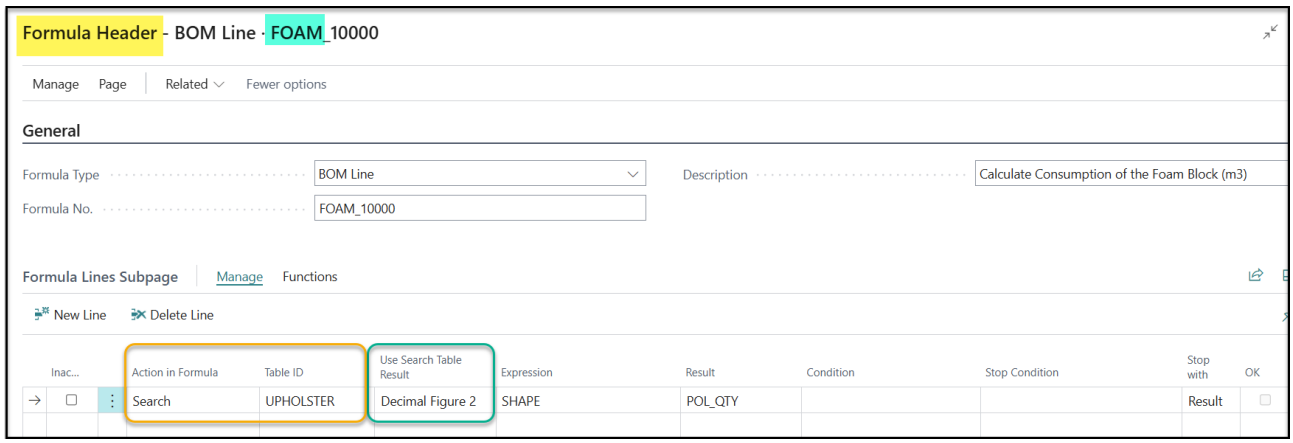
Note
Since this **Search Table** can be used in other Masters, we also need to have the Data in this Table required for these other Masters. All Upholstered Shapes can be on this Table.
This makes maintenance of Data quite easy because you do not need to go to individual Formula Lines. Besides that, you limit the number of Formula Lines, since now you can suffice with one Formula Line with a Search Table instead of a Formula Line per Shape Condition.

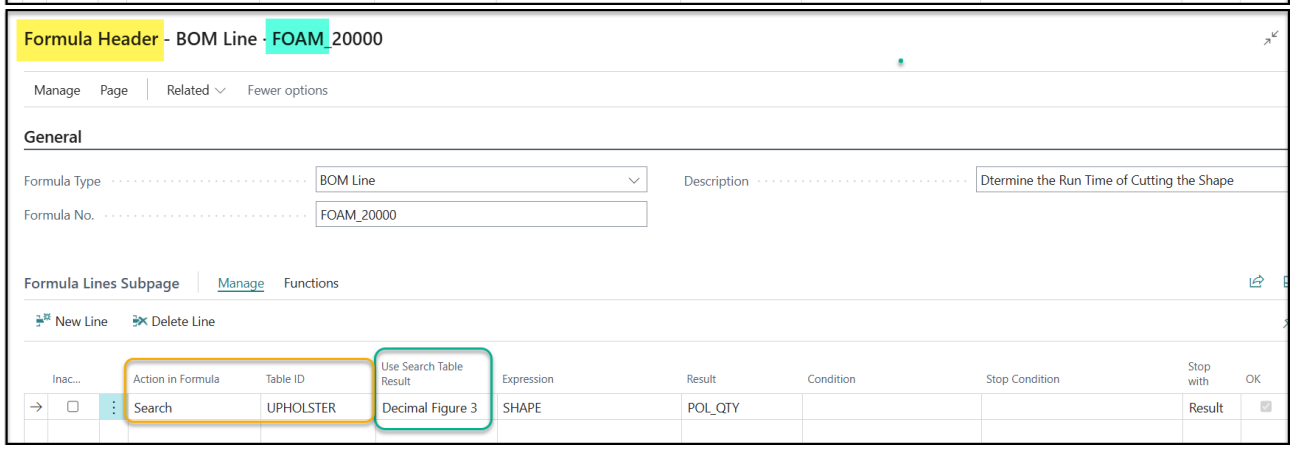
After the **Search Table** is created, you can attach it to the appropriate Formula Lines. Because we have three Output columns, we now also need on the Formula Lines the Field **Use Search Table Result** to define which Output column should be used for this Formula Line.

The Formula Line for the Component Fabric in the Master BOM of the Cloth is:



We have two Formulas for Foam parts: one for the Material and one for the Operation. These are shown below. Note the difference in the Field **Use Search Table Result** with the one in the screenshot above.





Conclusion

As you can see, many BOM Levels can have many Components with many Formula Lines. That is why you should initially test Formula by Formula, Component by Component and Level by Level.
If you are getting more experienced, you could do it in blocks.
The author hopes that you are convinced by this example of the importance of analyzing the BOM Structure first before you start building Formulas on Master BOM Component Lines.

TRIMIT Solutions provides a Document that has more cases like this for inspirational purposes.
This document is currently under construction.

Configuration Formulas

Introduction

In chapter *Variables* is described how **VarDim Types** can be used as Variables in the Formula Lines. When you are configuring an Item, it is possible to have a Formula executed, when you are entering a **Value** in one of the **VarDim Types** in the **VarDim Order**, this way you can manipulate one or more of the other answers. However, this requires some specific settings in the **VarDim Type**.

Furthermore, it could be the case that a Formula manipulates a VarDim Value for a VarDim Type that also has a Configuration Formula. This is called '***cascading***' of Formula execution.

Both topics are discussed in the paragraphs below.

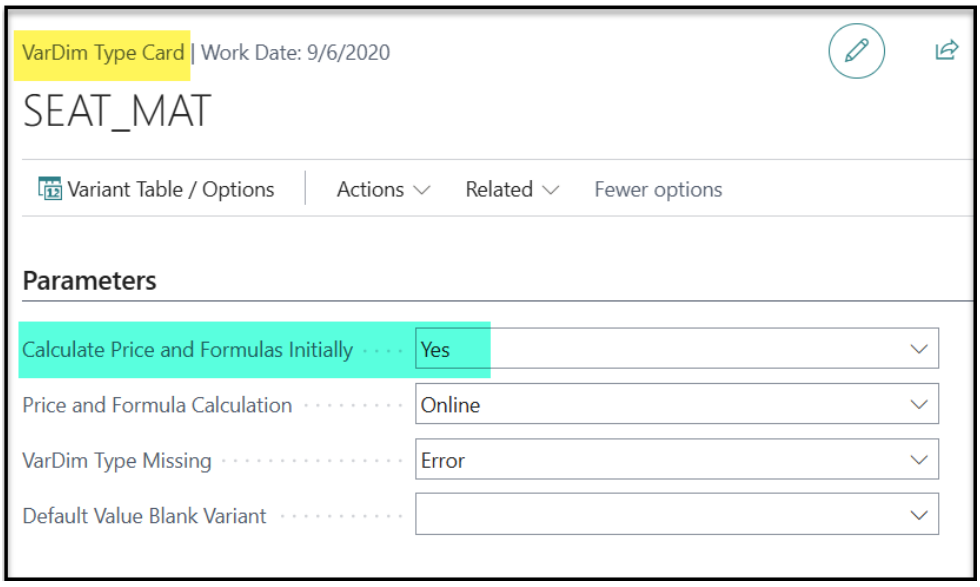
VarDim Master Card Settings

The configuration of Items starts by entering the equivalent of VarDim Master on a new:

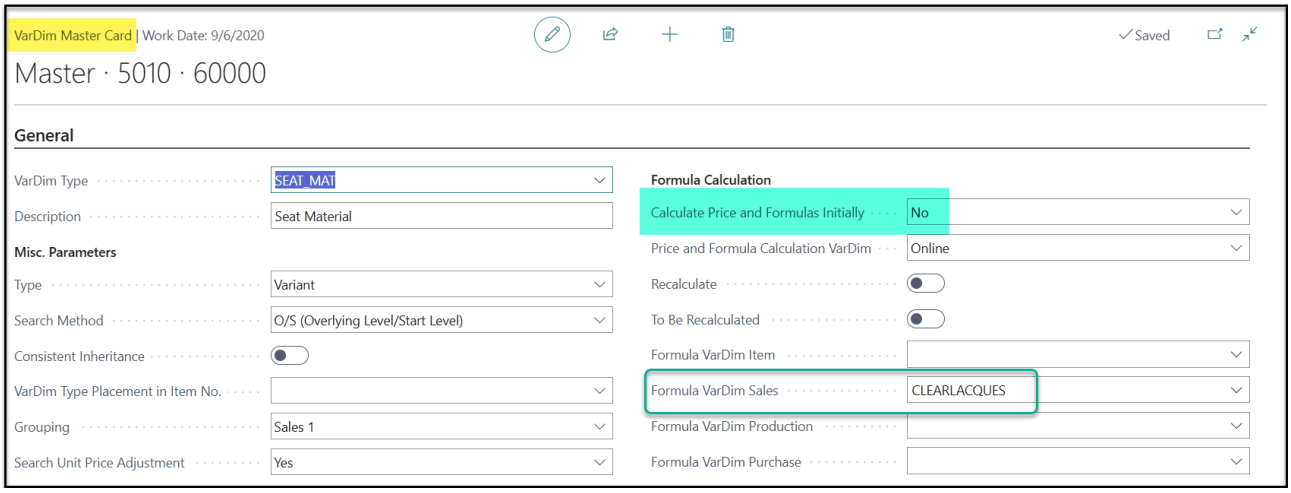
- Item Card
- Sales Line
- Purchase Line
- Production Header
- Production Collecting Order Line
- Item Journal Line
- Project Planning Line

Note
It is not possible to start a configuration on the **Transfer Order Line**.

There is a setting on the **VarDim Type Card** (FastTab ***Parameters***), that arranges if Prices are calculated and/or TRIMIT Formula are executed when the VarDim Item or VarDim Order page opens.
This Field is called **Calculate Price and Formulas Initially**. The default value of this *Option* Field is *Yes*.



When entering a VarDim Type on the ***VarDim Master*** Card, you can change the content in the VarDim Master Line or Card. This Value of the Field in the VarDim Master will be inherited to the ***VarDim Item*** Card.



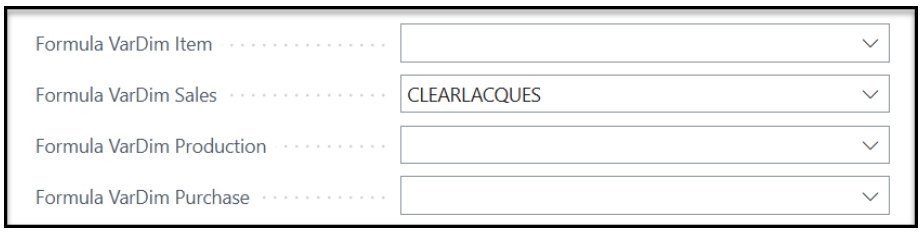
If this Field has the value *Yes*, it means that all Formulas connected to the VarDim Type are executed. Most of these Formulas require Values for the Variables representing the VarDim Types in the **Expression** or **Condition** Fields on the Formula Lines.
Unfortunately, these Values may not have been entered in the VarDim Item/Order yet.
If that is the case, an error message will pop up and block the opening of the VarDim Item/Order page.

To prevent this, the **Calculate Price and Formulas Initially** must be set on *No* when there is a possibility of executing a Configuration Formula. This is shown in the screenshot above.

Which Formula is executed during the Configuration is set up on the ***VarDim Master Card***.
In the Table below, you see which Configuration Formula is used at which moment.

Configuration on:	Configuration Formula
Item	Formula VarDim Item
Item Journal	Formula VarDim Item[16]
Sales Order Line	Formula VarDim Sales
Purchase Order Line	Formula VarDim Purchase
Production Order Header	Formula VarDim Production
Production Collecting Order Line	Formula VarDim Production
Project Line	Formula VarDim Sales[17]

These configuration Formulas can be set up in the ***VarDim Master Card***.



[16] There are no special **Global Variables** or **Code** created to update the Fields on the Item Journal Lines. However, Configuration Formulas that manipulates Values in the VarDim Order are possible.
[17] There are no special **Global Variables** or **Code** created (yet) to update the Fields on the Project Lines. However, Configuration Formulas that manipulates Values in the VarDim Order are possible.

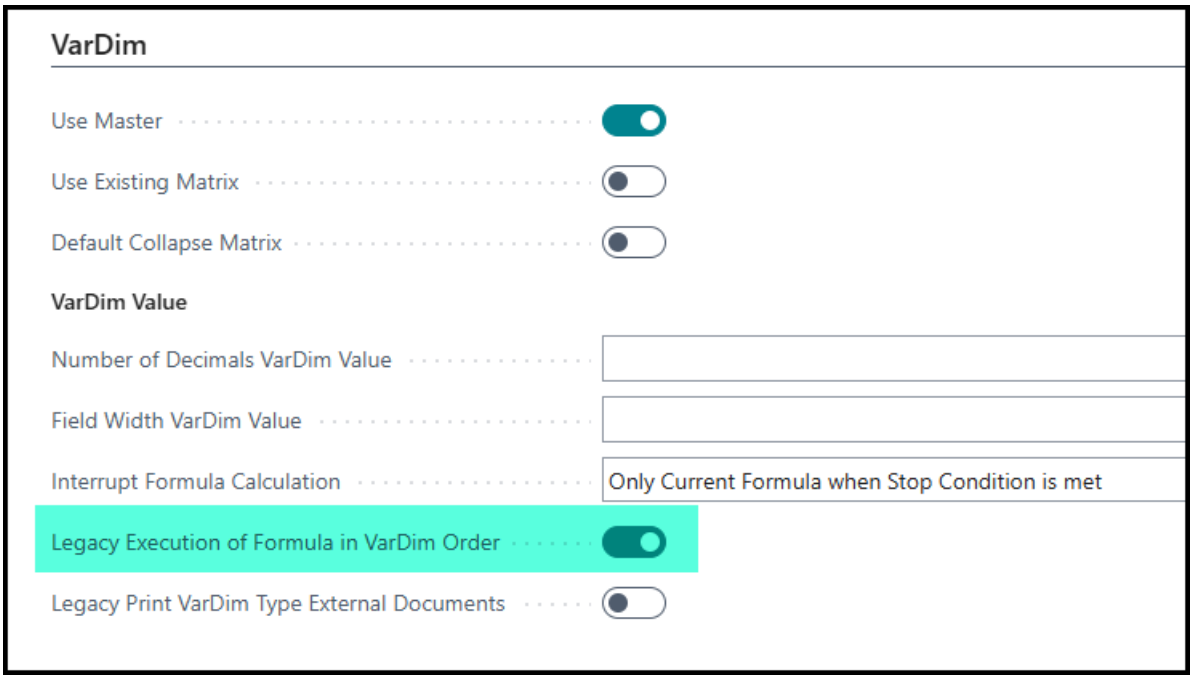
Cascading Formulas

In a ***VarDim Master Card*** or on the ***VarDim Master List***[18] we can enter a Formula working on the **VarDim Master Line**. This Formula will be executed when you manually enter a **Value** in this VarDim Master Line. The consequence of the execution of this Formula could be placing a **Result** Value in one or more of the other VarDim Master Lines. However, what will happen if one of the other VarDim Master Lines also has a VarDim Formula attached? There are two situations:

- You also want that Formula to be executed. Then we talk about cascading Formulas
- You only want to have this Formula to be executed when there is a manual input.

Note
With ***cascading Formulas,*** you must be aware of the possibility of ***infinite loops***. A Formula execution result in entering a Value in a VarDim that also executes a Formula that changes the initial VarDim, that executes the first Formula, end so on and so on…. This must be avoided at all times.

In the **Inventory Setup** you can find a Boolean Parameter **Legacy Execution of Formula in VarDim**.
This Parameter is initially invisible but can be put on the card when required via ***Personalize***.
With this Parameter you can prohibit cascading Formulas when you set it to *TRUE*. This is also the default setting when you upgrade to TRIMIT 22.2.7959 or later versions.
Once you set this Parameter to *FALSE*, it stays that way after new upgrades.

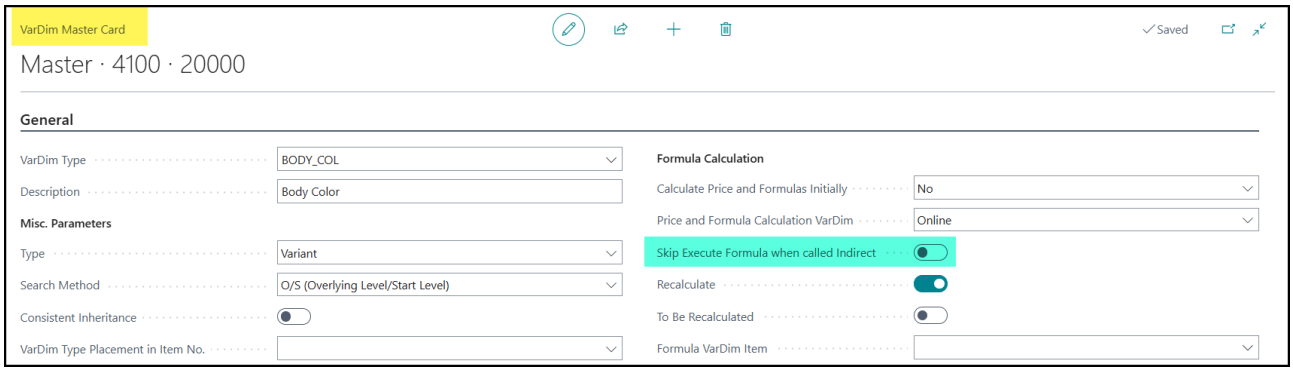


The background behind this Parameter is that in older versions of TRIMIT cascading Formulas were not possible. It is undesirable that Formulas are cascading after an upgrade and then come into a loop.
At some moment, this Parameter becomes obsolete.

If this Parameter has the value *FALSE*, it will make cascading of Formulas possible.
However, on the ***VarDim Master Card*** there is a Field that overrules this *FALSE*-setting. This Boolean Field is called **Skip Execute Formula when called Indirect**[19]. This means that the Formula on this

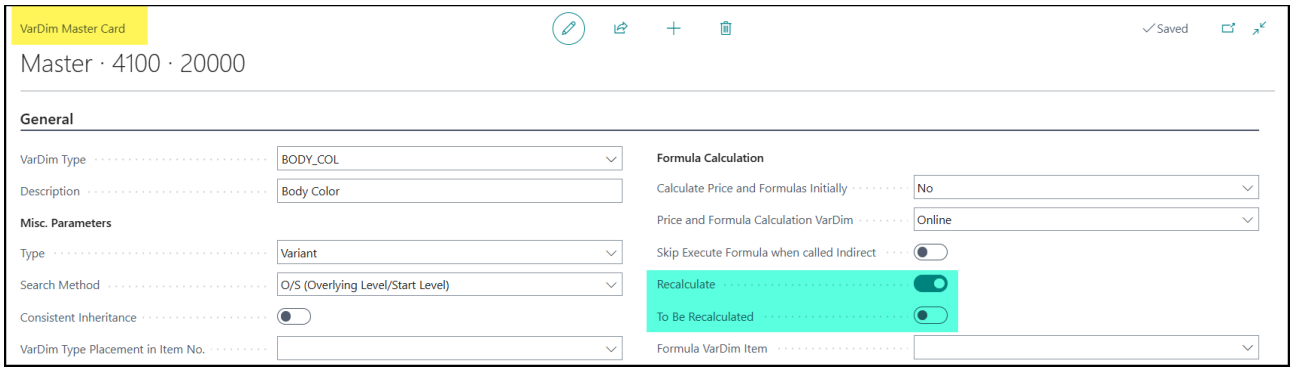
[18] The **Formula VarDim** Fields are invisible but can be put on the ***VarDim Master List*** via ***Personalize***.
[19] This Field is by default invisible, but can be placed on the ***VarDim Master Card***, if desired via ***Personalize***.

VarDim Master Line will not be executed if the Value comes in indirectly via execution of another Configuration Formula. The Formula will only be executed after manual input.



Calculate and Recalculate

There is another way to arrange cascading Formulas on the **VarDim Master Lines**. Those are the Boolean Fields **Recalculate** and **To Be Recalculated** on the **VarDim Master Card**.
If the Field **Recalculate** on a VarDim Master Card is set to *TRUE*, then not only is the Formula attached to this VarDim Master Line calculated, but also the Formulas on other VarDim Master Lines that have set the Boolean **To Be Recalculated** to *TRUE*.



The difference between this method and the method in the previous paragraph, is that in this case the cascading of Formula execution can only take place after entering the **Value** on the **VarDim Master Lines** with **Recalculate** = *TRUE*. The Formulas on the VarDim Master Lines with **To Be Calculated** = *FALSE* will not be executed if they are indirectly populated via other VarDim Formulas. So, this method is only of use, when you want to have cascading Formulas only for specific situations.

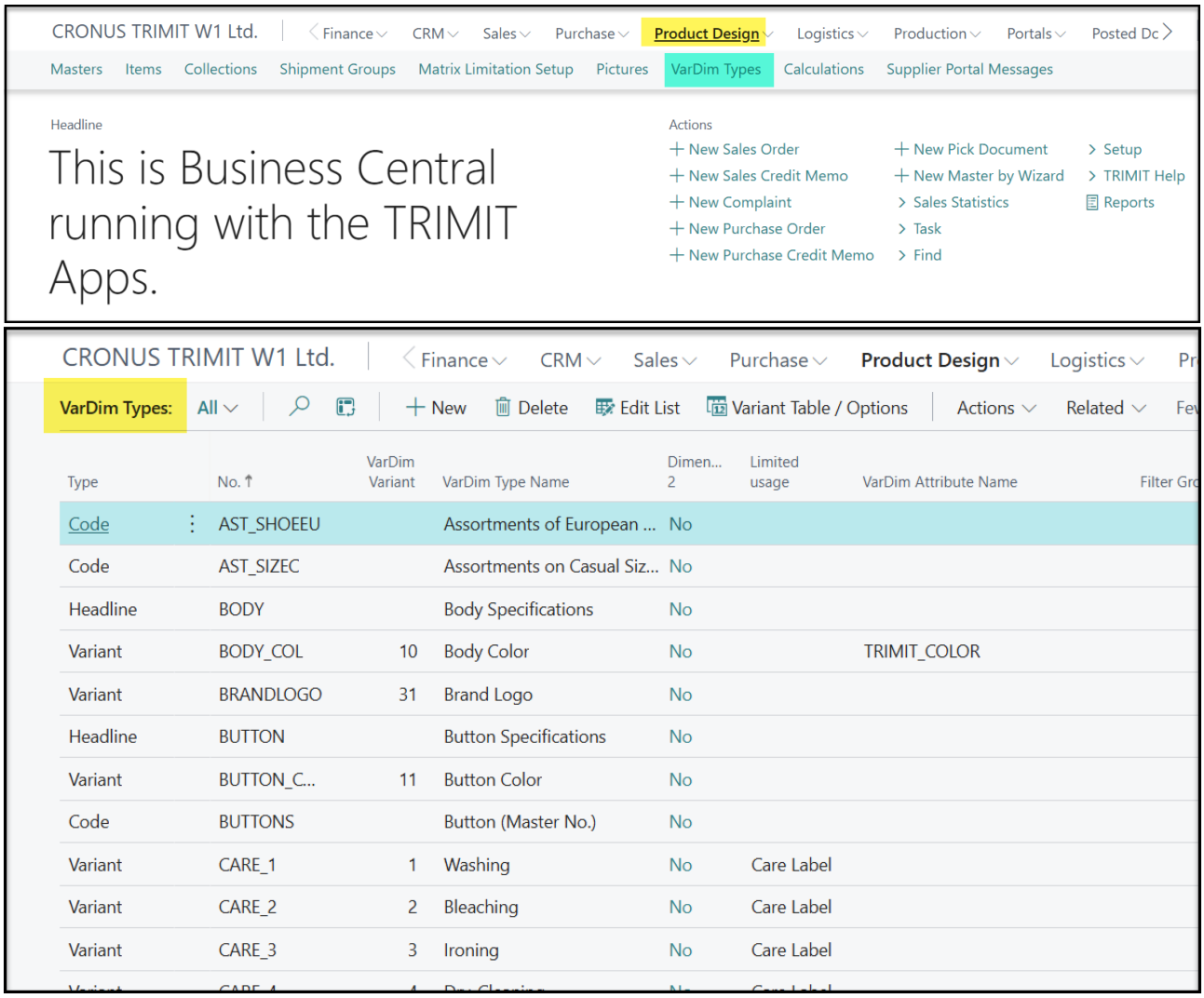
The default Value of both settings is *FALSE*.

These settings on the VarDim Master are only working when entering a VarDim Order.
If there is already a VarDim Item with Values in play, then these settings will not have any influence.

Appendix A: VarDim Types

Introduction

VarDim Types can be used as Variables in **TRIMIT Formulas**. VarDim Types are used in connection with **TRIMIT Variant Management** to describe the properties of a **Master** that are relevant regarding Sales, Production, Purchase, and Inventory.
Since the VarDim Types can be attached via the VarDim Master to more than one Product, they can be found in the **VarDim Types** List available directly on several TRIMIT Role Centers.



Every VarDim Type has its own **VarDim Type Card** in which a lot of Fields can be set for configuration purposes. This **VarDim Type Card** is discussed later in this Appendix.
Important objects regarding VarDim Types:

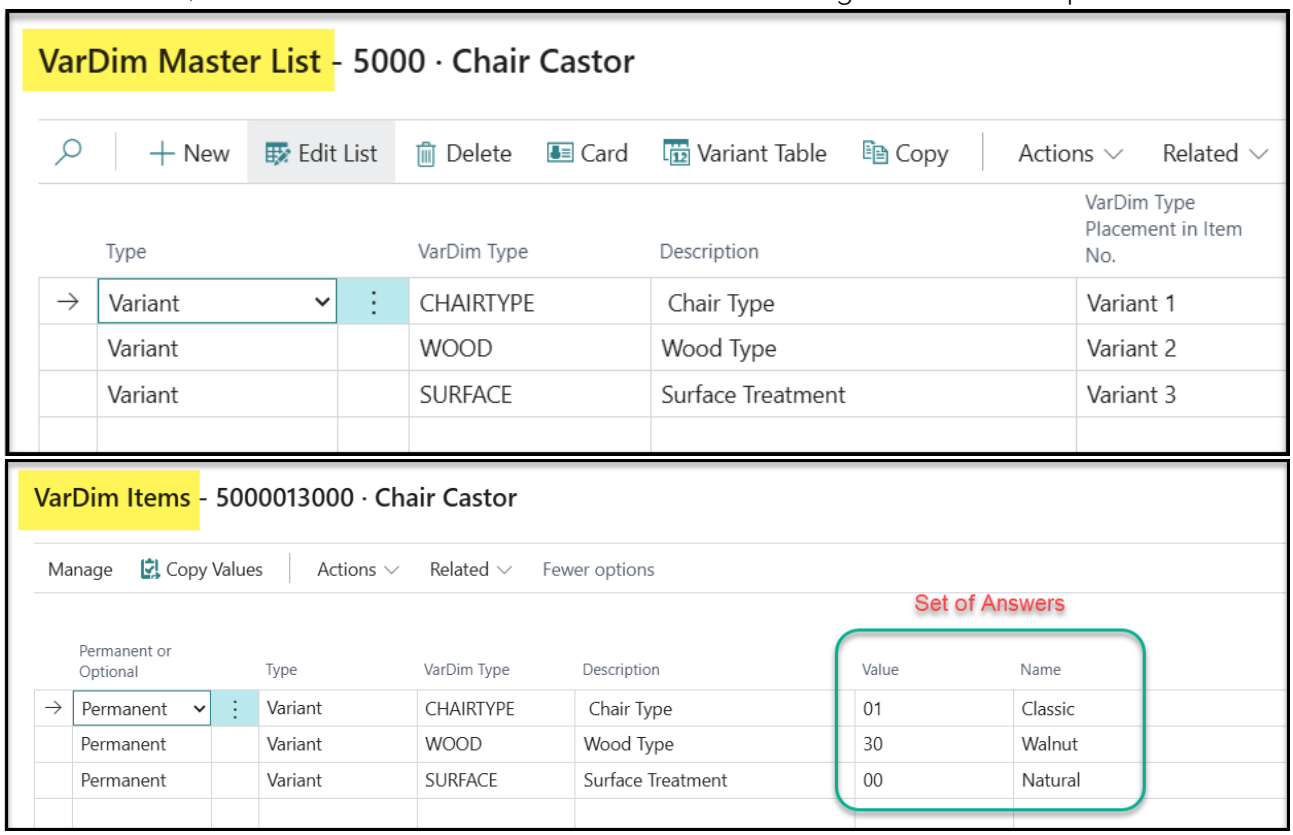
Tables

6037100 ***trm VarDim Type***
6037101 ***trm VarDim Master***
6037105 ***trm VarDim Order***
6037102 ***trm VarDim Variant Option***
6037106 ***trm VarDim Item***

Pages

6037100 ***trm VarDim Types List***
6037104 ***trm VarDim Type Card***
6037115 ***trm VarDim Items***

An arbitrary number of **VarDim Types** can be attached to a **Master**. This happens in the **VarDim Master**. When creating an **Item** with relation to the Master, a **VarDim Item** can be attached to the Item. You can consider the VarDim Master a set of configuration *Questions* specific for a Master that need to be answered. If so, the VarDim Item can be considered as a set of configuration *Answers* specific to an Item.



Note
Since the **Values** in the **VarDim Item** are specific for an Item you may choose to have them represented in **Item No.** as you can see in the example above. However, that is not a necessity, since you can also work with *Generic* **Item Nos.** or an **Item No.** with a Sequence Number in it.

It is possible to configure an Item from a Master in Sales, Purchase and Production **Documents**, the questions from the **VarDim Master** are now answered in the **VarDim Order**. The VarDim Order can also contain Values from **VarDim Types** of **Items** with a *Generic* **Item No.** and are therefore Document specific.

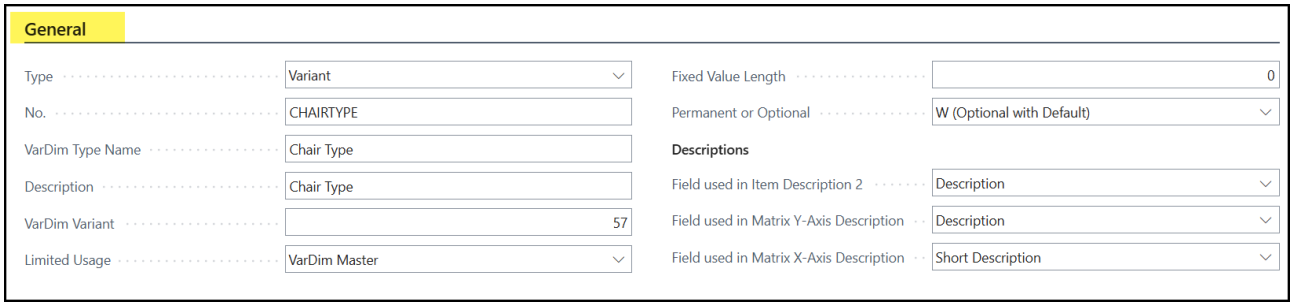
When a Production Order Line is generated or a Purchase Order Line is generated, a **VarDim Prod. Line** attached to the Production Order Line and a **VarDim Purchase** attached to the Purchase Order Line can be generated simultaneously.

The properties entered in **VarDim Item**, **VarDim Order**, **VarDim Prod. Line** and **VarDim Purchase** can be used as Variables regarding **Formula** calculation. When a Formula including one or more of these Variables is triggered, that Variable gets the Value of the property.

Let us discuss the FastTabs on the **VarDim Type Card** in the next paragraphs.

FastTab General

Let us discuss the Fields on the FastTab ***General*** of the ***VarDim Type Card***:



Type

The Field **Type** decides how the **Value** Field for the **VarDim Item**/**VarDim Order** is interpreted by the program. You can choose between the following seven options:

Dimension	Value is interpreted as Decimal Figure. With a **VarDim Type** of **Type** *Dimension* you can use the functionality on the Master concerning ***Dimension Table*** (default Values) and ***Max./Min. Dimensions***.
Variant	Value is interpreted as Code, according to predefined Values in the connected **VarDim Variant Option** Table.
Headline	No value. This **Type** can be used as Text, and the Field **Description** will be shown in "Bold" when calling **VarDim Item** and **VarDim Order**.
Date	Value is interpreted as an exact Date
Date Formula	Value is Interpreted as Date Formula
Code	Value is interpreted as Code. A **VarDim Type** of **Type** *Code* can via setup on the **VarDim Type** Card be related to a Table (For instance **Item**)
Decimal Figure	Value is interpreted as a Decimal

No. + VarDim Type Name + Description

The *Code* Field **No.** of the **VarDim Type** is the Key Field of this Table. It is used in connection with Formulas.

The **VarDim Type Name** is a *Text* Field representing the Name used in connection with using the **VarDim Type** in connection with **VarDim Master,** and which are shown in **VarDim Item**, **VarDim Order** etc.

In the *Text* Field **Description,** you have the possibility to add a more detailed Description. Initially the **Description** will be copied and maintained from the **VarDim Type Name** until you manually change it.

Note
Only **VarDim Types** of **Type** *Variant* or *Dimension* can be used as part of the **Item No.** creation.

VarDim Variant

The *Integer* Field **VarDim Variant** is only used for **VarDim Type** of **Type** *Variant*. This Field maintains the relation to the options in the **Variant Option** Table with the VarDim Type. By creating a new VarDim Type of **Type** *Variant* a new relation number is automatically created, but it can be overwritten manually for instance if the same relation Table should be used on multiple **VarDim Types.**

An example of this could be a relation Table with two outcomes 0 = *No* and 1 =*Yes*. Instead of creating multiple relation Tables with the same outcomes, the first created relation Table (**VarDim Variant**) is connected to multiple **VarDim Types**.

Another example is that on one particular Product, the same Material is used multiple times, but with different Variants. For instance, a Shirt has in its **BOM** the same Fabric for the Body as for the Sleeves, but the latter has a different Color. Since you cannot have the same **VarDim Type** more than one time in a **VarDim Master**, we need different VarDim Types (Color Groups): one for the Body and one for the Sleeve. However, since the options are the same, we could choose both VarDim Types to have the same **VarDim Variant** number. This way, they refer to the same Variant Option Table for the Colors.

The biggest advantage here is that you only must maintain one **Variant Option Table**.
Adding or deleting options are done for all **VarDim Types** connected to that Table.

Limited Usage

With the *Option* Field **Limited Usage**, you can determine that the **VarDim Type** can only be used in particular places in the system. This concerns mostly pages in which you want to enter specific information. Especially for the **PDM**[20] functionality in TRIMIT, this Field could be of interest.
You can choose between the following options:

<empty>	You can use VarDim Type anywhere.
VarDim Master	You can only use the VarDim Type in the **VarDim Master** and the **VarDim Item**. Of course it is also available for the VarDim Orders**.**
Composition	You can only use the VarDim Type in the **Composition** of your Master or Item.
Care Label	You can only use the VarDim Type in the **Care Label** of your Master or Item.
Additional Info	You can only use the VarDim Type in the **Additional Info** of your Master or Item.
Measurement Chart	You can only use the VarDim Type in the **Measurement Chart** of your Master or Item.
Assortment	You can only use the VarDim Type in the **Assortments** of your Master.

Permanent or Optional

With the option Field **Permanent or Optional**, you can set the default Value for the **VarDim Type**.
The Value will be transferred to the same Field in **VarDim Item** next time a VarDim Item is created.
The Field defines if and how the VarDim Item Line is transferred to **VarDim Order**, in connection with the generation of this VarDim Order**.** You can choose between the following four options:

Permanent	The **VarDim Item** Line is not transferred to **VarDim Order** unless the Field **Skip Insert VarDim Order** in the **VarDim Master** is *FALSE.* If the **VarDim Item** Line is transferred to **VarDim Order**, it will not be editable.
W (Optional with Default)	The **VarDim Item** Line is always transferred with the actual **Value** Field to **VarDim Order**. The **Value** in **VarDim Order** is allowed to be changed afterwards.
O (Optional without Default)	The **VarDim Item** Line is always transferred without the actual content of the **Value** Field to **VarDim Order**.
S (Optional by Manual Demand)	The **VarDim Item** Line is only transferred without the **Value** Field to **VarDim Order,** if the action ***Actions, Supply VarDim Order*** on the **VarDim Order** page is executed. The **Value** in **VarDim Order** will initially be empty.

[20] PDM: Product Data Management. Often used for design of apparel Products.

The content of this Field is also transferred to **VarDim Item** as default Value but is editable. If the **VarDim Type** is part of Item Number, the Value is automatic *Permanent.*

Default value is *W (optional with default).*

Note
If the **Value** of a chosen option is represented in the **Item No.**, it becomes *Permanent* in the **VarDim Item**. Otherwise, you could have an undesirable situation where the **Item No.** and one or more of the Variants are not aligned anymore.

Field Used in Item Description 2

With **Field Used in Item Description 2**, you can determine which Value of a Variant should be added/shown in **Description 2** of the Item.
You can choose between the following four options:

Description	The **Description** of the **VarDim Variant Options** will be used. I.e., *Dark Grey*
Short Description	The **Short Description** of the **VarDim Variant Options** will be used. I.e., *DGREY*
Value	The **Value** of the **VarDim Variant Options** will be used. I.e., *90*
Exclude	Nothing will be added of the **VarDim Variant Option** into **Description 2** of a generated Item.

The default Value is *Description*.

Field Used in Matrix Y-Axis Description

With **Field Used in Matrix Y-Axis description**, you can determine which Value of a Variant should be shown if this Variant would be on the Lines of the Matrix.
You can choose between the following three options:

Description	The **Description** of the **VarDim Variant Options** will be used. I.e., *Dark Grey*
Short Description	The **Short Description** of the **VarDim Variant Options** will be used. I.e., *DGREY*
Value	The **Value** of the **VarDim Variant Options** will be used. I.e., *90*

The default value is *Description*.

Field Used in Matrix X-Axis Description

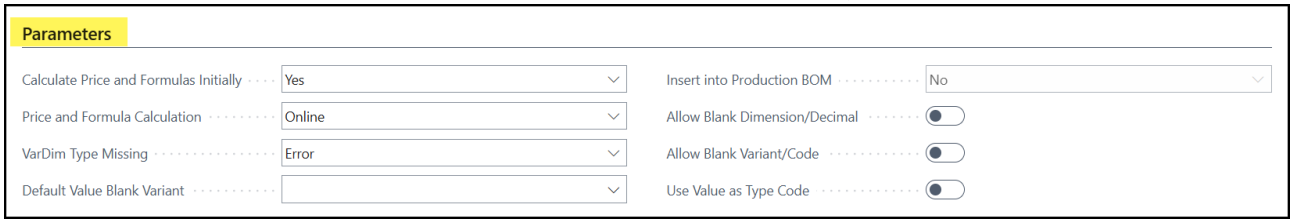
With the **Field Used in Matrix X-Axis Description**, you can determine which Value of a Variant should be shown if this Variant would be in the Columns of the Matrix.
You can choose between the following three options:

Description	The **Description** of the **VarDim Variant Options** will be used. I.e., *Dark Grey*
Short Description	The **Short Description** of the **VarDim Variant Options** will be used. I.e., *DGREY*
Value	The **Value** of the **VarDim Variant Options** will be used. I.e., *90*

The default value is *Short Description*.

FastTab Parameters

Let us discuss the Fields on the FastTab ***Parameters*** of the ***VarDim Type Card***:



Calculate Price and Formulas Initially

The *Option* Field **Calculate Price and Formulas Initially** decides if price and Formula calculation is carried out in connection with the call of the **VarDim Order**.
You can choose between the following two options:

Yes	Price and Formula calculation is carried out in connection with call of ***VarDim Order***
No	Price and Formula calculation are not carried out in connection with call of ***VarDim Order***

The default value is *Yes*.
The content of this Field is transferred to **VarDim Master** as default Value but still editable.
See also in this appendix the paragraph *VarDim Master Card Settings* concerning use of this Field.

Price and Formula Calculation

The *Option* Field **Price and Formula Calculation** decides in which situation the Formula should be executed in connection with the **VarDim Order**.
You can choose between the following three options:

Online	The Formula and the Price Calculation is executed every time the **Value** in VarDim Order is changed and when VarDim Order is opened, unless the Field **Calculate Price and Formula Initially** is *No*.
When Leaving VarDim Order	The Formula and the Price Calculation is executed every time you finish (escape the window) working in VarDim Item or VarDim Order.
Online + When Leaving VarDim Order	The Formula and the Price Calculation are executed every time the **Value** in VarDim Order/VarDim Item is changed and when VarDim Order/VarDim Item is opened and closed.

The default value is *Online*.
The content of this Field is transferred to **VarDim Master** as default Value but still editable. From VarDim Master it is transferred to **VarDim Item** and **VarDim Order** as default Value but still editable.

VarDim Type Missing

The *Option* Field **VarDim Type Missing** refers to how the Generation of BOM Structure and Formula Calculation should react if the program meets a Formula Line with a **VarDim Type** that has not been present in the VarDim Master.
You can choose between the following two options:

Error	The Formula Calculation stops with an error message.
Skip	The Value of the missing **VarDim Type** is *0* or "" (Blank) and Formula Calculation continues

The default value is *Error*.

Default Value Blank Variant

The *Code* Field **Default Value Blank Variant** can only be used in combination with **VarDim Types** of **Type** *Variant* or *Code.* Here you can set a default Value for the **VarDim Variant Option** or Value in the relation Table. This is often used when the execution of a Formula Line meets a VarDim Type that did not get a Value in the Overlying Level or configurator.

Insert into Production BOM

The *Option* Field **Insert into Production BOM** can only be used for **VarDim Types** of **Type** *Code.*
With this functionality the chosen option can be automatically inserted on the BOM Component Line. However, the Value of that option will be interpreted as an **Item No.** This means that there should be an appropriate Table related to the **Option** Table of this VarDim Type. This is discussed in this appendix in paragraph *FastTab Table Relations*.
You can choose between the following three options:

No	No action.
Before Generating BOM based on Item/Master BOM	The Value is interpreted as **Item No.** and is automatically inserted in the top of the Item/Master BOM.
After Generating BOM based on Item/Master BOM	The Value is interpreted as **Item No.** and is automatically inserted at the bottom of the Item/Master BOM.

The default value is *No*.

Note
Be aware that the **Quantity per** on the Item BOM is set by the Value in the Field **Quantity** on the **VarDim Order**. The default Value for **Quantity per** is *0*, so if you do not enter this Quantity, no Component Line will be inserted. The **Quantity** Field is by default invisible and needs to be put on the VarDim Order page via ***Personalize***.

Allow Blank Dimension/Decimal

The *Boolean* Field **Allow Blank Dimension/Decimal** can only be used in combination with **VarDim Types** of **Type** *Dimension* or *Decimal Figure.*
You can choose between the following options:

FALSE	The Values *0* or *"" (Blank)* are not allowed
TRUE	The Values *0* or *"" (Blank)* are allowed.

The default value is *FALSE*.

Allow Blank Variant/Code

The *Boolean* Field **Allow Blank Variant/Code** can only be used in combination with **VarDim Types** of **Type** *Variant* or *Code.*
You can choose between the following options:

FALSE	The Value *"" (Blank)* is not allowed
TRUE	The Value *"" (Blank)* is allowed.

The default value is *FALSE*.

Use Value as Type Code

Depending on the **Type** of the **VarDim Type**, the **Value** in the **VarDim Order** could be a *Decimal* or an *Alphanumeric Code*. This has repercussions on how the system treats this Value.
For instance , *0 (zero)* as *Decimal* will be seen as blank. However, as *Code* it will be seen as a Text.
In *Code* Fields you can also handle 00 and 000, which in *Decimal* Fields will be seen as *0 (zero)*.
With the *Boolean* Field **Use Value as Type Code**, you can handle Values as if they were *Codes*.

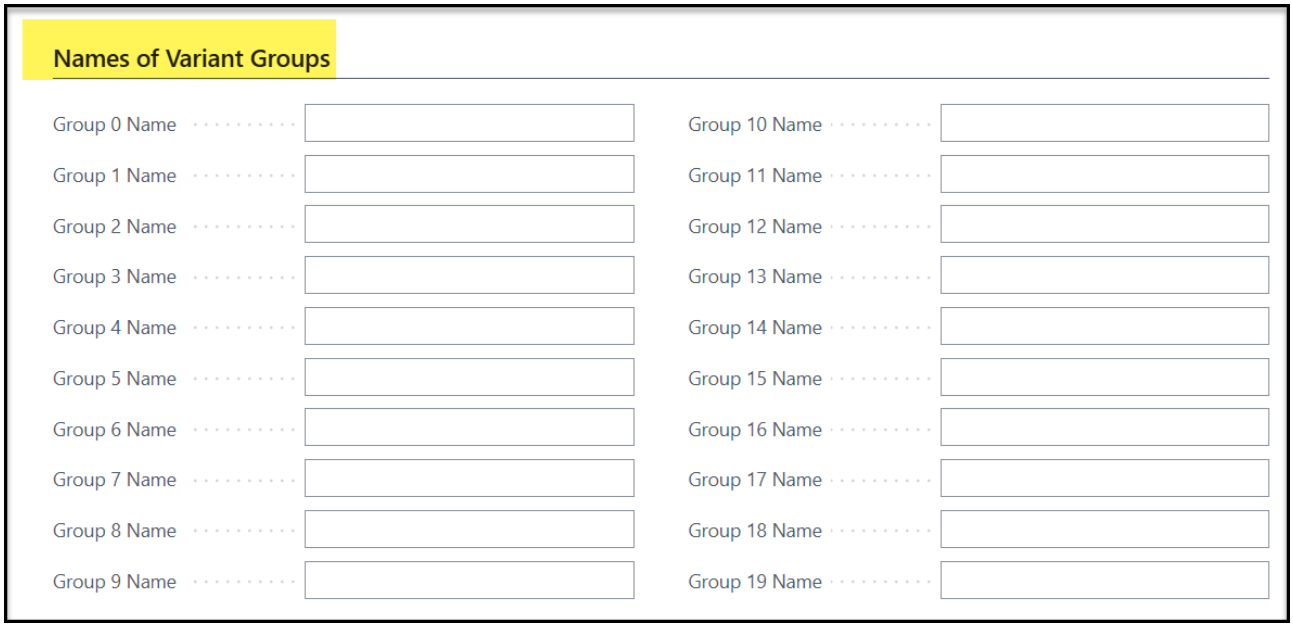
The opposite is also possible. If a *Code* Value is purely numeric, it can be used in Formula Lines as a *Decimal* Field. This does not require any setting.

When using a "-" or a "_" in a Value, you need to set this Field to *Yes.* The system might otherwise interpret it as a minus-sign.

The default value of this Field is *FALSE*.

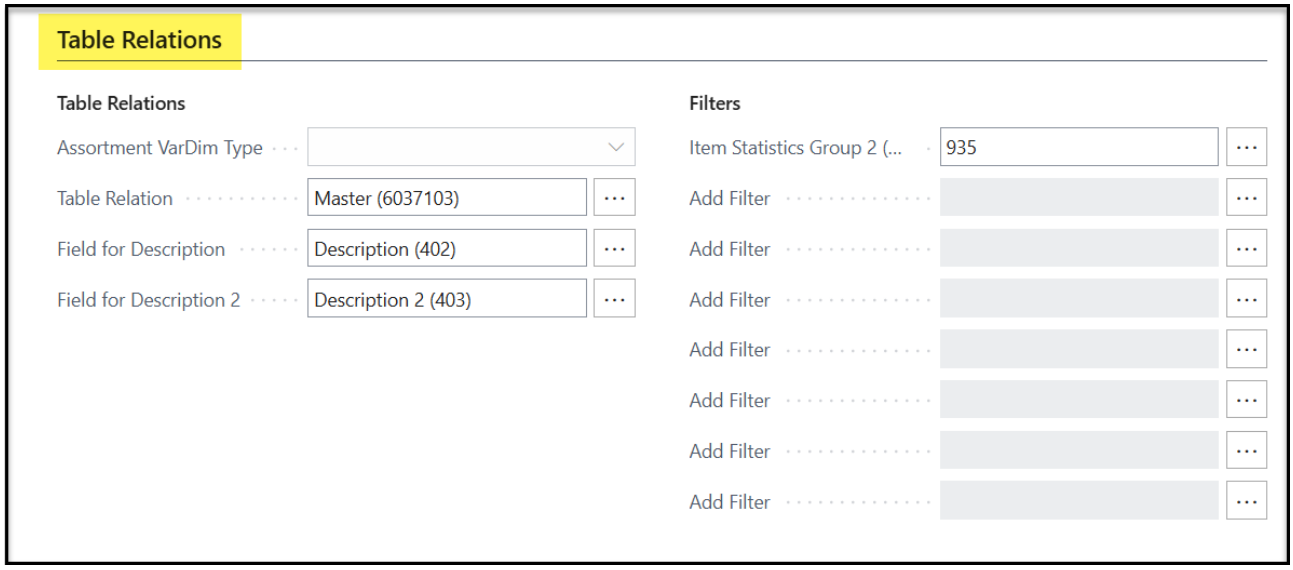
FastTab Names of Variant Groups

In the different *Text* Fields **Group 0-19 Name**, you can enter a Description that acts as **Captions** of the **Group Columns** in the **Variant Option Table** of **VarDim Types** of **Type** *Variant*.
This way you can make Groupings of Options. This feature is discussed in chapter *Formula Line Structure* Paragraph *Formula Expression AssistEdit (Alt+Arrow Down)*.



FastTab Table Relations

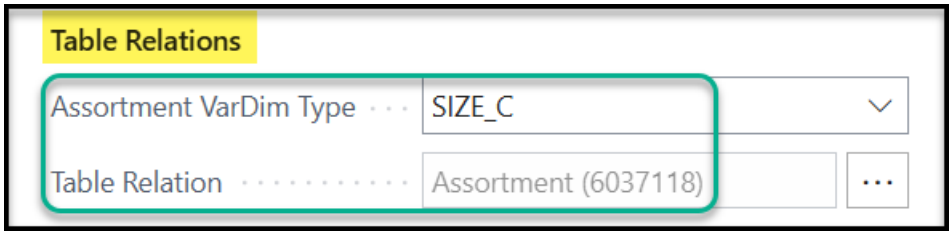
The Fields on this FastTab can only be used in combination with **VarDim Types** of **Type** *Code*.



Assortment VarDim Type

Fashion Products can often go in Assortments. There are several ways to handle these, and they are described in ***White Paper - TRIMIT Assortments***. In this document we do not go into detail about this functionality except that, if you have a **Single Color-Multiple Size Assortment** or a **Single Size-Multiple Color Assortment**, this Assortment can be set up as a **VarDim Type** of **Type** *Code*.

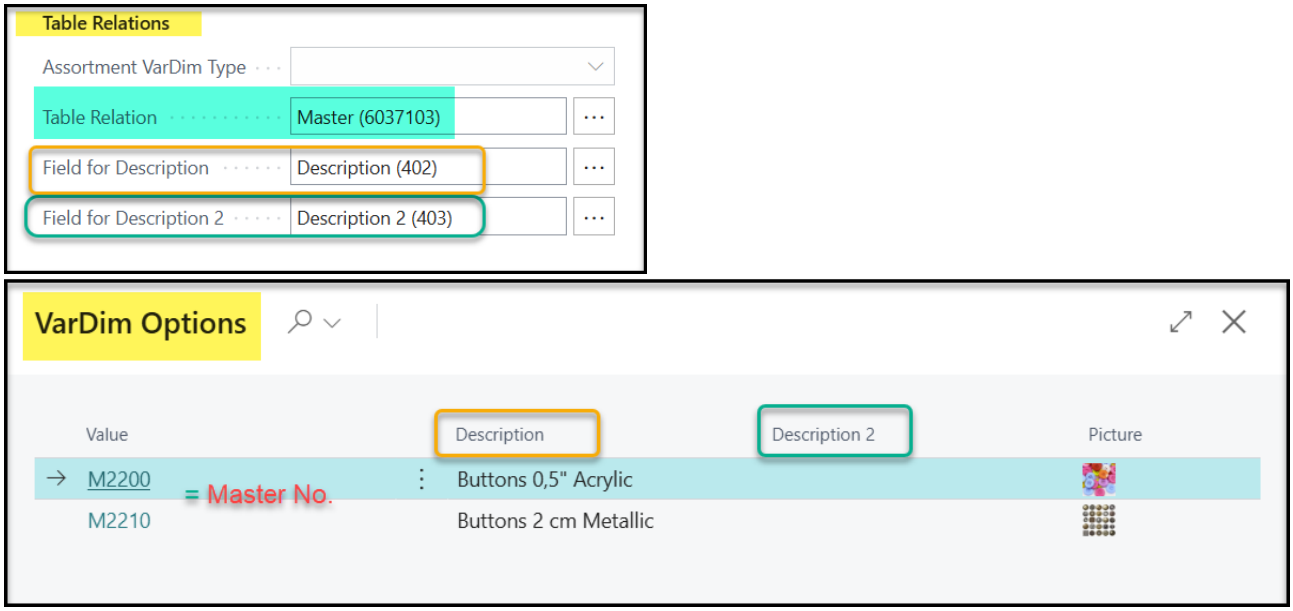
In the *Code* Field **Assortment VarDim Type**, the appropriate **VarDim Type** of **Type** *Variant* that represents a Size Table or Color Table should be entered. The Field **Table Relation** will then be automatically populated with the *Assortment Table (6037118)*. In that case the Field will be greyed out.



Other Table Relations Fields

With the *Code* Field **Table Relation**, you can relate the **VarDim Type** to any BC/TRIMIT Table. It can only work well for Tables with one key Field. You can find an appropriate Table with help from the **AssistEdit** button.

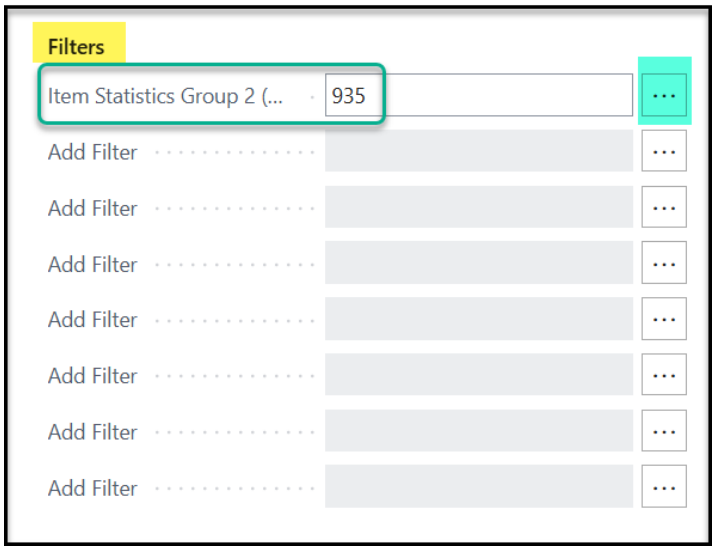
After entering the Table, the **VarDim Type** is acting now as a Table lookup when used for Configuration. It is possible to limit the Data in that lookup by setting the appropriate Filters.
The **primary key** acts as the Search input in the Table. For instance, if the VarDim Type is related to the **Master** Table, the Value in the Search Table is the **Master No.**



This way it is possible to make a selection between different **Masters**. Of course, these Masters can come in different Sizes, Colors, etc. This means that either other Configuration answers or a Formula should take care of its Variations. Of course, you could do this also directly on the **Item** Table, but in that case, you must be sure that all Items exists in the system. If you use the Master and determine the Variations, the Item does not need to be in the system and can be created on the fly at a later moment. In the *Text* Fields **Description** and **Description 2** in the Table Relations, you can determine which Field should be used in the **Name** and **Name 2** in the **VarDim Order**/**VarDim Item**. Eventually, a picture can also be in the Option Table if there is one.

Filters

However, you could imagine that using Master, Item or any other Table in the system, the Option Table can be long. And possibly not all Options are allowed.
Therefore, it is possible to create a set of **Filters**. There could be a combination of filters up to *8* Fields on the Table set in the **Table Relation** Field.
Choose the Field in the Table that should be used for Filtering via the **AssistEdit** button on one of the **Add Filter** Fields. The caption will change in the Name of the selected Field. In example below, our first Filter is set on *Item Statistic Group 2* (in the **Master** Table).



Now the **Add Filter** Field is connected to a Field from the Related Table. You can now enter the appropriate Filter criteria in the **Add Filter** Field that has just been renamed.
In the example above we took **Item Statistic Group 2** *935* as a Filter, which means that only Masters with Item Statistic Group *2 935* can be selected from the Option Table, which are the two *Button* **Masters** in this case.

FastTab Groupings

The FastTab ***Groupings*** has only two Fields.

Filter Group

The **Filter Group** Field previously played a significant role in determining Filters on TRIMIT e-commerce. However, in current and recent versions, its influence has been greatly reduced by the usage of standard BC Item Attributes and becoming almost obsolete. It is likely to be removed in future versions.

So do not use this Field!

VarDim Attribute Name

In the Field **VarDim Attribute Name,** you can enter the Item Attribute Name for the VarDim Type.
The usage of **Item Attributes** is according to standard BC.

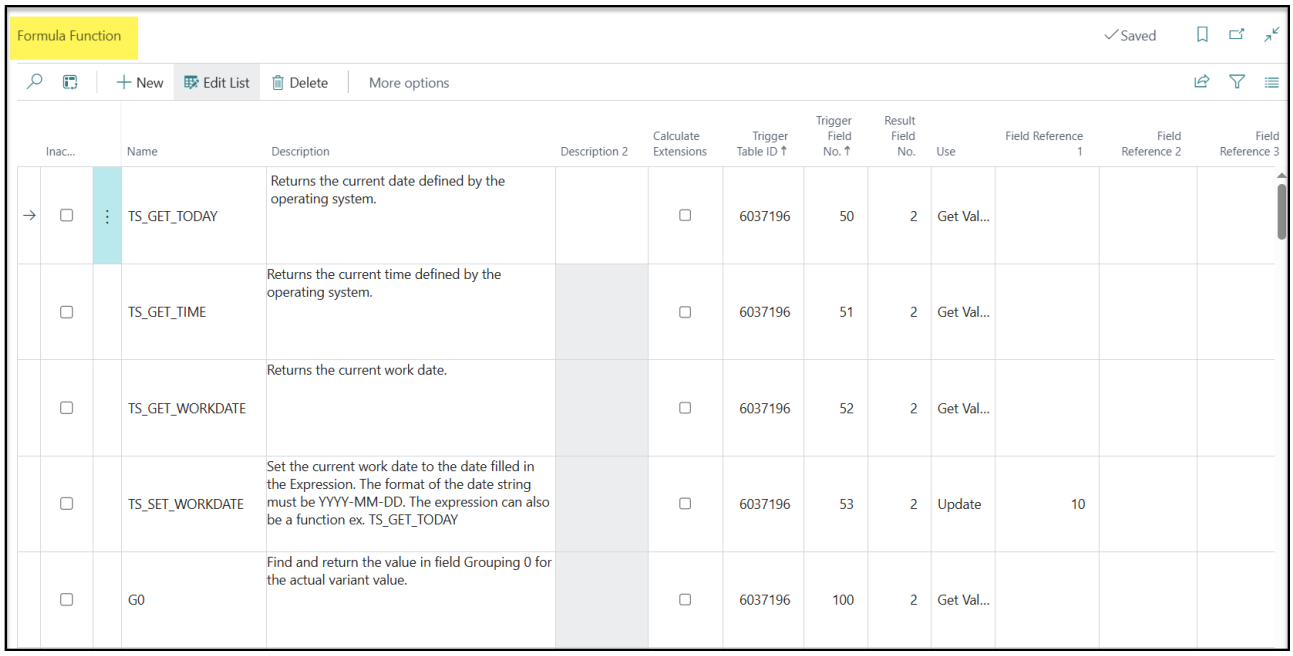
Filtering on TRIMIT e-commerce is primarily controlled by the Business Central Item Attributes and this is replacing the **Filter Group** mentioned above. The Item Attributes can be attached to Masters, Items, and Variants of VarDim Types (but only if the **VarDim Attribute Name** has been filled in the VarDim Type.

Appendix B: Functions

Introduction

In **TRIMIT Formula Lines** it is also possible to have more elaborate calculations than the normal mathematical calculations as shown in the *FactBox* **Mathematical Symbols** in the **Formula Expression Assist Edit**.

These (complex) calculations are defined as **Functions** written in Code (*AL Code*).
These Functions can be found in the Table **Formula Function**.



Inac...	Name	Description	Description 2	Calculate Extensions	Trigger Table ID	Trigger Field No.	Result Field No.	Use	Field Reference 1	Field Reference 2	Field Reference 3
☐	TS_GET_TODAY	Returns the current date defined by the operating system.		☐	6037196	50	2	Get Val...			
☐	TS_GET_TIME	Returns the current time defined by the operating system.		☐	6037196	51	2	Get Val...			
☐	TS_GET_WORKDATE	Returns the current work date.		☐	6037196	52	2	Get Val...			
☐	TS_SET_WORKDATE	Set the current work date to the date filled in the Expression. The format of the date string must be YYYY-MM-DD. The expression can also be a function ex. TS_GET_TODAY		☐	6037196	53	2	Update	10		
☐	G0	Find and return the value in field Grouping 0 for the actual variant value.		☐	6037196	100	2	Get Val...			

TRIMIT provides several **Formula Functions** as standard. These are created when initializing a new Database or Company. These **Formula Functions** are described in a separate ***White Paper - TRIMIT Functions*** in which we can see also examples of how these Formula Functions can be used.
The White Paper - TRIMIT Functions is under construction.

Formula Functions are created in Table ***6037196 API Formula Function***. The Formula Function is assigned a name and can then, in the way as other Global Variables, be used in **Expression, Condition, Stop Condition** and **Result.**
Depending on the Formula Function, it can be used as Variables or as an Extension for a **VarDim Type**. (For instance: Grouping Functions G0-G19.)

In addition to the *standard* **Formula Functions**, partners and customers with access ***AL Code*** can make their own **Formula Functions** and make them available for **TRIMIT Formulas**.
It is recommended that **Formula Functions** programmed as customization are placed in partner/customer specific Tables. The *AL Code* that is executed when the **Formula Function** is called in **TRIMIT Formulas** can be placed in a Field in any Table the user has user rights to.

In the paragraph below, the Fields of the **Formula Function** Table are described. Furthermore, in this Appendix, an example is shown how to create your own Formula Function.

Formula Function Fields

Inactive
If this Field is *TRUE*, the Formula Function will not be shown in ***Function Overview*** when editing the Formulas.

Name
The **Name** of the Formula Function.

Description
Here you can add a **Description** for the Formula Function. Typically use, is to explain the use and how the Formula Function should be called.

Description 2
Here you can add an additional **Description 2** for the Formula Function. Typically use, is to explain the use and how the Formula Function should be called.

Calculate Extensions
The **Calculate Extension** must be *TRUE* if a Formula Function or calculated Value is used within the call of a Formula Function.

Trigger Table ID
The **Table No.** where the Formula Function can be found.
All the TRIMIT standard Formula Functions are placed in Table ***6037196 API Formula Function***.

Trigger Field No.
The **Field No.** in the Table set in the Field **Trigger Table ID**, where the AL Code that should be executed can be found.

Result Field No.
The **Field No.** in the Table is set in the Field **Trigger Table ID**, where the Result Value that should be returned can be found.
In Table ***6037196 API Formula Function***. Field No. 2 to 9 can return the Value in the following formats: *Text, Decimal, Integer, Date, Time, Code, Boolean*.

Use
This Field decides if the Formula Function is allowed only to be used when retrieving Data, allowing to Update or Both. You can choose between the following three options:

Get Value	Value can be retrieved
Update	Value can be updated
Get Value and Update	Value can be retrieved and updated

Field Reference 1 - Field Reference 10
These Fields are used to point at Fields where the call record should place the Values that are in the call (Values are always of **Type** *Code*), and which should be used in the Formula Function to return the correct Value.
Up to 10 Fields can be in the initial call to the Formula Function. In Table ***6037196 API Formula Function***. Field No. 10 to 19 are used to store the Initial call Values.

Example on how to make your own Function Table

You should spend a little time looking into the possibilities for making Formula Functions instead of making Events as this is often a better and quicker way of retrieving and changing Data in the system. In connection with Formula Calculation (and implicit the Formula Function execution) you are standing with the actual structure you are calling the Formula from, which means that the records in the calling structure are available as full Variables (VAR). These records can be accessed using the Formula Function in Codeunit ***6037214 Formula Set and Get Parameters***.

The best advice is to look at and get inspired by the OnValidate sections in TRIMIT standard Formula Functions placed in Table ***6037196 TRIMIT API Formula Function***.

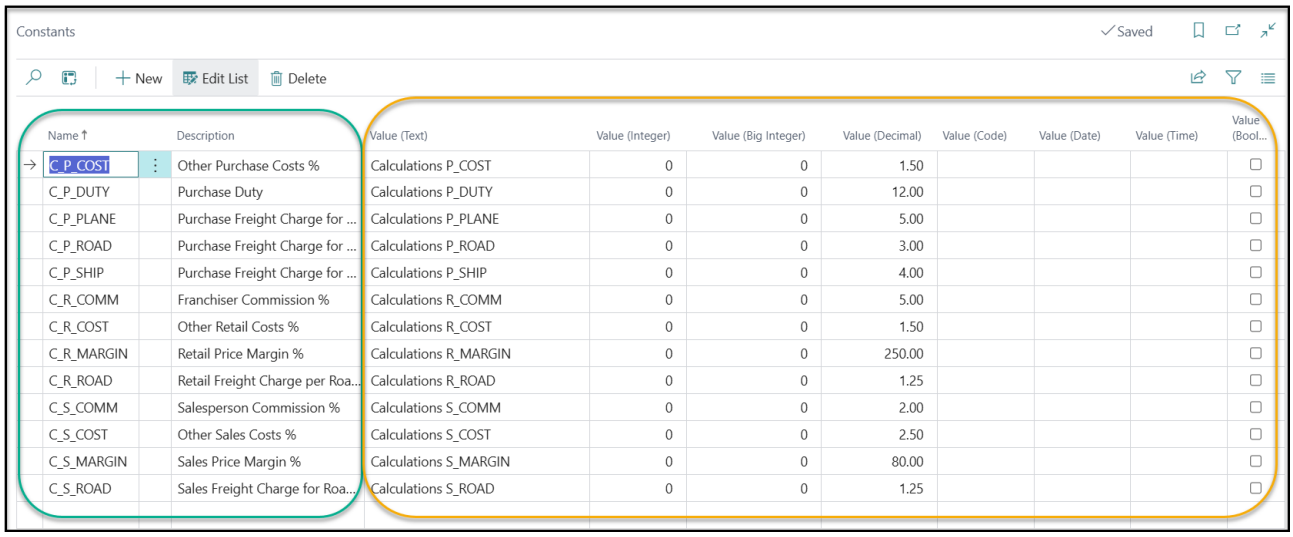
Steps:

1. Take a copy of the Table ***6037196 TRIMIT API Formula Function.*** and save it in the 50000-object range with another Object Name
2. Delete all Fields in the newly created Table containing existing TRIMIT Standard Functions (Field No. *50* and upwards) but keep all Fields below this No. (all the Result and Field Reference Fields)
3. Create a new Field (start minimum at Field No. *50* to leave space if future development demands expansion of the standard area) and Name your Function in the **Field Name**.
4. Make your trigger code for the Function on the *OnValidate* section on the Field.

Go to page **6037027 Formula Functions** and insert a new record pointing at your own Table ID, Trigger Field etc.

Appendix C: Formula Constants

Working with Formulas or Calcualation Templates sometimes require **Constants**.
You can think of Margins, Duty or forwarding Costs, etc. The advantage of using **Constants** is that they can be used in many Calculations but can be maintained in one Table: ***Constants***.

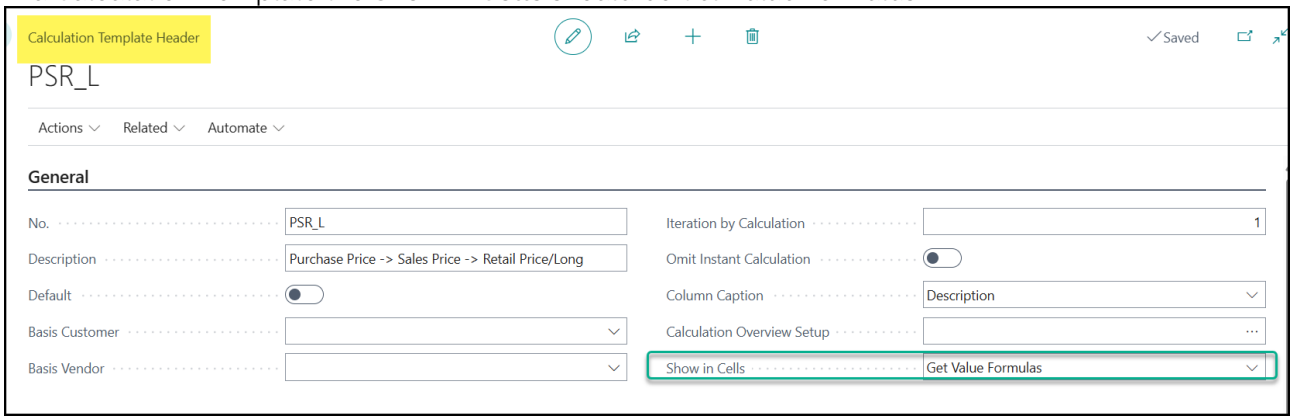


Name	Description	Value (Text)	Value (Integer)	Value (Big Integer)	Value (Decimal)	Value (Code)	Value (Date)	Value (Time)	Value (Bool...)
C_P_COST	Other Purchase Costs %	Calculations P_COST	0	0	1.50				
C_P_DUTY	Purchase Duty	Calculations P_DUTY	0	0	12.00				
C_P_PLANE	Purchase Freight Charge for ...	Calculations P_PLANE	0	0	5.00				
C_P_ROAD	Purchase Freight Charge for ...	Calculations P_ROAD	0	0	3.00				
C_P_SHIP	Purchase Freight Charge for ...	Calculations P_SHIP	0	0	4.00				
C_R_COMM	Franchiser Commission %	Calculations R_COMM	0	0	5.00				
C_R_COST	Other Retail Costs %	Calculations R_COST	0	0	1.50				
C_R_MARGIN	Retail Price Margin %	Calculations R_MARGIN	0	0	250.00				
C_R_ROAD	Retail Freight Charge per Roa...	Calculations R_ROAD	0	0	1.25				
C_S_COMM	Salesperson Commission %	Calculations S_COMM	0	0	2.00				
C_S_COST	Other Sales Costs %	Calculations S_COST	0	0	2.50				
C_S_MARGIN	Sales Price Margin %	Calculations S_MARGIN	0	0	80.00				
C_S_ROAD	Sales Freight Charge for Roa...	Calculations S_ROAD	0	0	1.25				

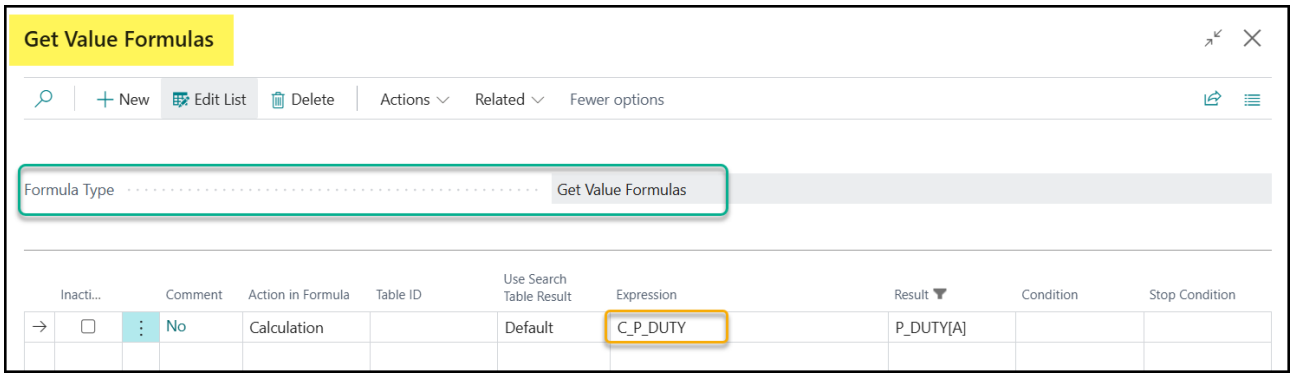
For instance:
Let us assume that you want to include a Duty Cost of *12%* within a Calculation, you could enter within a Formula **Expression** *0.12*, representing the Duty Cost. It could be the case that there are more Formula **Expressions** where you enter this Duty Cost. When the Duty Cost changes to *13%*, you need to change this into all Formula **Expressions**.
However, if you make a Constant for Duty Cost in the Table ***Constants*** and use this Constant in the Formula **Expressions**, you must make this change only in one place.

In the Table above you can see multiple **Value** Fields. It depends on the **Field Type**, which of the Values will be taken. For instance, when the Field is of **Type** *Decimal*, it takes the Field **Value (Decimal)**.

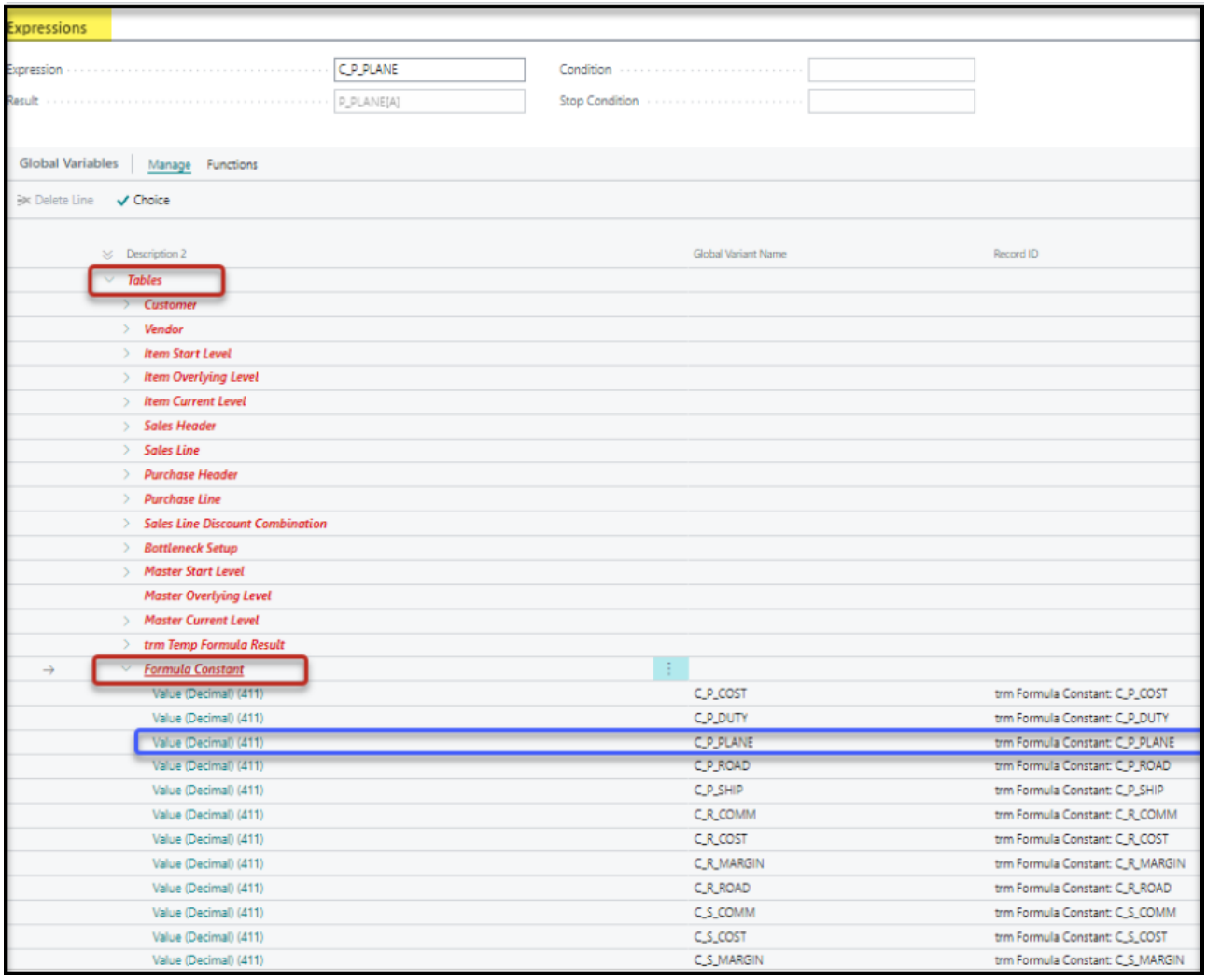
The **Formula Constants** can be found via the **Formula Expression AssistEdit**, if the Formula has **Formula Type** *Calculation* or *Global*, when they should be used in a regular Formula.
In a Calculation Template the **Show in Cells** should be *Get Value Formulas*.



Then in every Cell, you can drill-down to the Formula for that Cell and see:



You can find the Constants in the list of **Formula Expression AssistEdit** via *Tables*, *Formula Constants*.

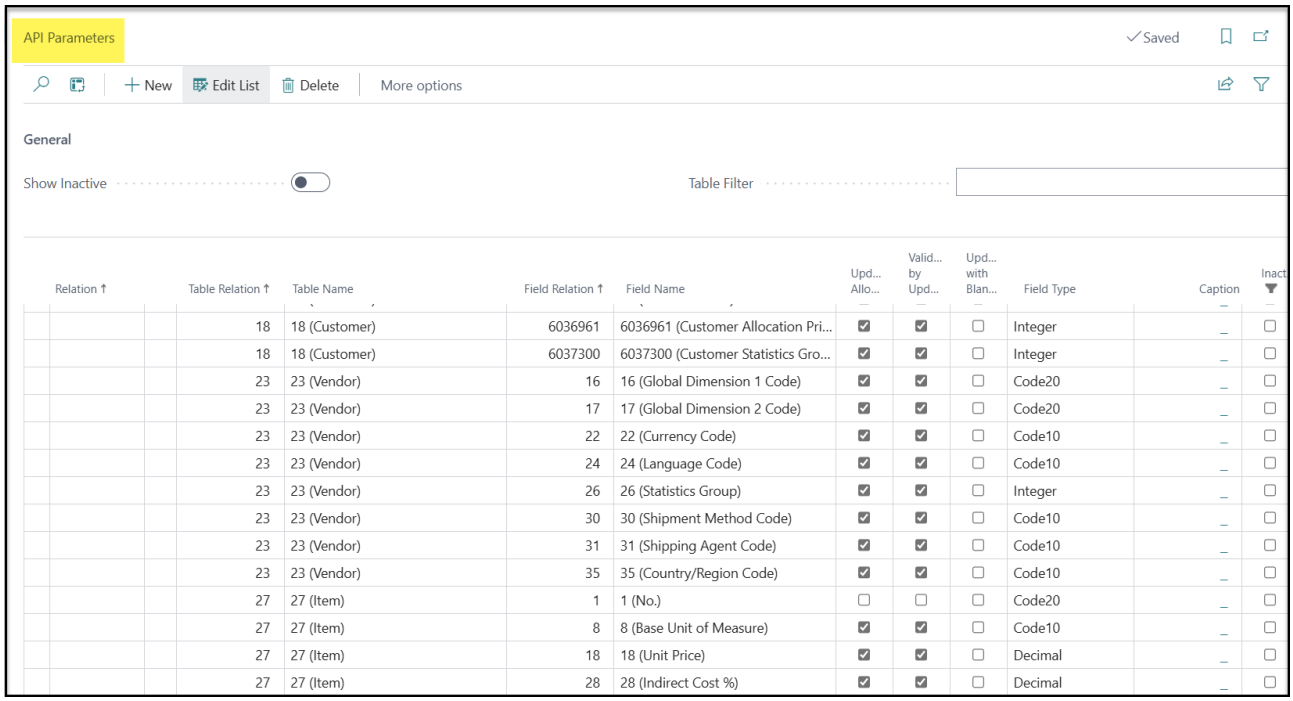


For examples how to use Constants in Calculation Templates, see ***White Paper - TRIMIT Calculations***.

Appendix D: API Parameters

Introduction

An **API Parameter** is used in connection with **TRIMIT Services** to determine if a Field in a Table is allowed to be Updated and if so if it happens by Inserting or Modifying the Value directly in the Table or by Modifying the Value into the Field. **TRIMIT Formulas** are using TRIMIT Services when the **Result** of a Formula calculation is written in the Database.



Relation ↑	Table Relation ↑	Table Name	Field Relation ↑	Field Name	Upd... Allo...	Valid... by Upd...	Upd... with Blan...	Field Type	Caption	Inact
	18	18 (Customer)	6036961	6036961 (Customer Allocation Pri...	☑	☑	☐	Integer		☐
	18	18 (Customer)	6037300	6037300 (Customer Statistics Gro...	☑	☑	☐	Integer		☐
	23	23 (Vendor)	16	16 (Global Dimension 1 Code)	☑	☑	☐	Code20		☐
	23	23 (Vendor)	17	17 (Global Dimension 2 Code)	☑	☑	☐	Code20		☐
	23	23 (Vendor)	22	22 (Currency Code)	☑	☑	☐	Code10		☐
	23	23 (Vendor)	24	24 (Language Code)	☑	☑	☐	Code10		☐
	23	23 (Vendor)	26	26 (Statistics Group)	☑	☑	☐	Integer		☐
	23	23 (Vendor)	30	30 (Shipment Method Code)	☑	☑	☐	Code10		☐
	23	23 (Vendor)	31	31 (Shipping Agent Code)	☑	☑	☐	Code10		☐
	23	23 (Vendor)	35	35 (Country/Region Code)	☑	☑	☐	Code10		☐
	27	27 (Item)	1	1 (No.)	☐	☐	☐	Code20		☐
	27	27 (Item)	8	8 (Base Unit of Measure)	☑	☑	☐	Code10		☐
	27	27 (Item)	18	18 (Unit Price)	☑	☑	☐	Decimal		☐
	27	27 (Item)	28	28 (Indirect Cost %)	☑	☑	☐	Decimal		☐

Before it is possible to write a Value in the Database, the Field that must be updated must be present in the Table **API Parameter.**

Normally it is not necessary to create an **API Parameter** manually in connection with Formulas as the creation of a **Global Variable** automatic creates an API Parameter.
However, that is not the case for Fields brought in the system by Customization.
Then you need to create the API Parameter as is Described in the last paragraph of this Appendix.

Tables

6036197 trm API Parameters
6036469 trm API Sales Header
6036470 trm API Sales Line
6036474 trm API Purchase Header
6036475 trm Api Purchase Line
6036472 trm API VarDim Order

Pages

6037800 trm API Parameters

Fields in API Parameters

Active & Show Inactive

If the *Boolean* Field **Show Inactive** you can make the *Inactive* **API Parameters** visible. Whether an API Parameter is Inactive is determined by the *Boolean* Field **Inactive** on the API Parameter Line.

If the Field **Inactive** is *TRUE*, the **API Parameter** will be hidden on the active page. To make an already created API Parameter active again, filter on **Show Inactive** by entering a checkmark in that filter and remove the *TRUE* in **Inactive**. Default value is *FALSE*.

Table Relation

In the *Code* Field **Table Relation,** the **Table No.** is entered to which the **API Parameter** relates to.

Table Name

The *Text* Field **Table Name** is automatically populated with the **Name** of the entered **Table No.**

Field Relation

In the *Code* Field **Field Relation,** the **Field No.** is entered to which the **API Parameter** relates to.

Field Name

The *Text* Field **Field Name** is automatically populated with the **Name** of the entered **Field No.**

Update Allowed

If the *Boolean* Field **Update Allowed** is *TRUE*, the Field is allowed to be updated. Otherwise, the Update of the Field is skipped.

Validate by Update

If the *Boolean* Field **Validate by Update** is *TRUE*, the *OnValidate* code on the Field is executed when placing the Result in the Database.

Update with Blank/Zero

If the *Boolean* Field **Update with Blank/Zero** is *TRUE*, a Blank or Zero Value as result of the Formula execution is allowed to overwrite the existing Value. Otherwise, the original Value stands.

Formula

In the *Code* Field **Formula**, it is possible to attach a **TRIMIT Formula** that is executed in connection with all written transactions in the Database for the **Table No.** and **Field No.** the **API Parameter** is created on.

Field Type

The *Option* Field **Field Type** is automatically filled with **Field Type** from the **Field No.** in the actual Table.

Caption

Do not use the Field **Caption**.

TRIMIT Service Tables

The **API Parameters** are used in combination with the **TRIMIT Services**. For every Table that we have Global Variables on, there are Service Tables. The Service Tables in question are:

BC/TRIMIT Table	Service Table
Sales Header	trm API Sales Header
Sales Line	trm API Sales Line
Purchase Header	trm API Purchase Header
Purchase Line	trm API Purchase Line
VarDim Order	trm API VarDim Order
VarDim Item	trm API VarDim Item
Production Header	trm Prod. Header
Production Line	trm Prod. Line

Customization Fields in Service Tables.

Using customized Fields in TRIMIT it is recommendable to create equivalents/counterparts in the Service Tables. Especially using Formulas in TRIMIT it is necessary.
For instance, if your Customer needs a Field on the Sales Line the Service Table ***API Sales Line*** needs to be created as well.

```
0 references
tableextension 50111 "Custom SL" extends "Sales Line"
{
    fields
    {
        0 references
        field(50111; "Bulk"; Boolean)
        {
            Caption = 'Bulk';
            DataClassification = customerContent;


        }
    }
}

```

```
0 references
tableextension 50101 "Custom API SL" extends "trm API Sales Line"
{
    fields
    {
        0 references
        field(50111; "Bulk"; Boolean)
        {
            Description = 'Bulk';
            DataClassification = SystemMetadata;


        }
    }
}
```

Appendix E: Outcome Tables

Introduction

Working with **Conditions** and **Stop Conditions** can result in different Outcome combinations.
Condition and **Stop Condition** have 3 possible outcomes: *TRUE, FALSE* and *Blank*.
Stop With has 4 possible outcomes: *Result*, *Skip, Select* and *Go To.*
The **Go To** Field can be filled with a valid **Formula Line No.** or with "*0*" (zero).

In total we can have *64* possible combinations. These results of **Condition, Stop With, Stop Condition** and **Go To** are shown in four overviews of the **Outcome Tables** in the next paragraphs.
In the fifth **Outcome Table** the combinations of these four are described in the last paragraph.

The **Outcome Table** has three Outcome Columns. These are described below.

Line Self-Included

The Field **Line Self-Included** will be filled depending on the **Condition** and/or the **Stop Condition** in combination with the **Stop With**.
If the Value is *YES* (*Y*), then the action in the **Expression** on the Formula Line will be executed.
If the Value is *NO* (*N*), then the action in the **Expression** on the Formula Line will not be executed.

The Value can also be '*Blank.*' This is the result when the **Formula Line** is *annulled*. This happens when the current or a previous Formula Line has a **Stop Condition** in combination with the **Stop With** that interrupts the whole Formula Calculation. (see for instance *Output Table B*).

Stop

The Value in the Field **Stop** is determined by the combination of **Stop Condition** and **Stop With**.

If there is no **Stop Condition**, the Value of **Stop** becomes '*Blank.*' The Formula Calculation continues including the Formula Line itself. This is also the case if the **Stop** Value is *NO* (*N*).

If the **Stop** Value is *YES* (*Y*), The rest of the Formula calculation is stopped including what is left of the calculation of the current Formula Line.

Go To

The Field **Go To** in the **Outcome Table** becomes '*Blank*' if the *Go To* option in the Field **Stop With** is not used on the Line or when the Formula Line Field **Go To** is not filled with a Line Number.

The Field **Go To** in the **Outcome Table** becomes *YES* (*Y*), if the **Stop Condition** is met, the **Stop With** has the option *Go To* and the **Go To** Field in the Formula Line is filled with a Line Number.

If the **Stop Condition** is not met, but the **Stop With** has the option *Go To* and the **Go To** Field in the Formula Line has a Line Number filled, then the Field **Go To** in the Outcome Table becomes *NO* (*N*).

Only when the **Go To** Field in the Outcome Table has the Value *YES* (*Y*), the Formula Calculation jumps to the Line Number in the **Go To** Field in the Formula Line and from there continues the execution of the Formula.

Table A – Condition, Stop With and Go To

This Table shows the connection between **Condition, Stop With** and **Go To**.
It shows that it does not matter what is chosen in **Stop With**, because the outcome only depends on whether the outcome in **Condition** is *TRUE* or *FALSE* and if a number is given in **Go To**.

					Outcome		
Line	Condition (Value)	Stop With	Stop Condition (Value)	Go To	Line Self-Included	Stop	Go To
1	TRUE	Result		0	Y		
2	FALSE	Result		0	N		
3	TRUE	Skip		0	Y		
4	FALSE	Skip		0	N		
5	TRUE	Select		0	Y		
6	FALSE	Select		0	N		
7	TRUE	Go To		0	Y		
8	FALSE	Go To		0	N		
9	TRUE	Result		Filled	Y		Y
10	FALSE	Result		Filled	N		N
11	TRUE	Skip		Filled	Y		Y
12	FALSE	Skip		Filled	N		N
13	TRUE	Select		Filled	Y		Y
14	FALSE	Select		Filled	N		N
15	TRUE	Go To		Filled	Y		Y
16	FALSE	Go To		Filled	N		N

Table B – Stop With, Stop Condition and Go To

This Table shows the connection between **Stop With, Stop Condition** and **Go To**.
It is shown that the outcome depends on **Stop With** and that **Go To** only is activated when **Stop With** is **Go To** and a serial number is given in **Go To**.

					Outcome		
Line	Condition (Value)	Stop With	Stop Condition (Value)	Go To	Line Self-Included	Stop	Go To
1		Result	TRUE	0	Y	Y	
2		Result	FALSE	0	N	N	
3		Skip	TRUE	0		Formula annulled ("0")	
4		Skip	FALSE	0	N	N	
5		Select	TRUE	0	Y	N	
6		Select	FALSE	0		Formula annulled ("0")	
7		Go To	TRUE	0	Y	N	
8		Go To	FALSE	0	N	N	
9		Result	TRUE	Filled	Y	Y	N
10		Result	FALSE	Filled	N	N	N
11		Skip	TRUE	Filled		Formula annulled ("0")	
12		Skip	FALSE	Filled	N	N	N
13		Select	TRUE	Filled	Y	N	N
14		Select	FALSE	Filled		Formula annulled ("0")	
15		Go To	TRUE	Filled	Y	N	Y
16		Go To	FALSE	Filled	N	N	N

Table C – Condition, Stop With, Stop Condition and Go To

This Table shows the connection between **Condition, Stop With, Stop Condition** and **Go To**.
It is shown how **Condition** controls **Stop Condition** and vice versa.
E.g., as described as the outcome for **Condition, Stop With, Stop Condition** and **Go To**, Line *16* **Stop Condition** (*FALSE*) is overruled by **Condition** (*TRUE*) because a **Go To** is triggered like in Line *15* where **Stop Condition** is *TRUE*.

					Outcome		
Line	Condition (Value)	Stop With	Stop Condition (Value)	Go To	Line Self-Included	Stop	Go To
1	TRUE	Result	TRUE	0	Y	Y	
2	TRUE	Result	FALSE	0	Y	N	
3	TRUE	Skip	TRUE	0		Formula annulled ("0")	
4	TRUE	Skip	FALSE	0	Y	N	
5	TRUE	Select	TRUE	0	Y	N	
6	TRUE	Select	FALSE	0		Formula annulled ("0")	
7	TRUE	Go To	TRUE	0	Y	N	
8	TRUE	Go To	FALSE	0	Y	N	
9	TRUE	Result	TRUE	Filled	Y	Y	N
10	TRUE	Result	FALSE	Filled	Y	N	Y
11	TRUE	Skip	TRUE	Filled		Formula annulled ("0")	
12	TRUE	Skip	FALSE	Filled	Y	N	Y
13	TRUE	Select	TRUE	Filled	Y	N	Y
14	TRUE	Select	FALSE	Filled		Formula annulled ("0")	
15	TRUE	Go To	TRUE	Filled	Y	N	Y
16	TRUE	Go To	FALSE	Filled	N	N	Y
17	FALSE	Result	TRUE	0	N	Y	
18	FALSE	Result	FALSE	0	N	N	
19	FALSE	Skip	TRUE	0		Formula annulled ("0")	
20	FALSE	Skip	FALSE	0	N	N	
21	FALSE	Select	TRUE	0	N	N	
22	FALSE	Select	FALSE	0		Formula annulled ("0")	
23	FALSE	Go To	TRUE	0	N	N	
24	FALSE	Go To	FALSE	0	N	N	
25	FALSE	Result	TRUE	Filled	N	Y	N
26	FALSE	Result	FALSE	Filled	N	N	N
27	FALSE	Skip	TRUE	Filled		Formula annulled ("0")	
28	FALSE	Skip	FALSE	Filled	N	N	N
29	FALSE	Select	TRUE	Filled	N	N	N
30	FALSE	Select	FALSE	Filled		Formula annulled ("0")	
31	FALSE	Go To	TRUE	Filled	N	N	Y
32	FALSE	Go To	FALSE	Filled	N	N	N

Table D – Stop Functions

This Table sorts the combinations according to **Stop** Function.
In addition, "*Irr.*" (irrelevant) has been added. Meaning that it is irrelevant what is chosen in **Go To**

					Outcome		
Line	Condition (Value)	Stop With	Stop Condition (Value)	Go To	Line Self-Included	Stop	Go To
1	Stop + include the Line:						
2	T / B	Result	TRUE	0	Y	Y	
3	T / B	Result	TRUE	Filled	Y	Y	N
4	Stop + do not include the Line:						
5	FALSE	Result	TRUE	0	N	Y	
6	FALSE	Result	TRUE	Filled	N	Y	N
7	Stop with Go To + include the Line:						
8	T / B	Go To	TRUE	Filled	Y	N	Y
9	Stop with Go To + do not include the Line:						
10	FALSE	Go To	TRUE	Filled	N	N	Y
11	Full Stop:						
12	T / B	Select	FALSE	Irr.	Formula annulled ("0")		
13	FALSE	Select	FALSE	Irr.	Formula annulled ("0")		
14	T / B	Skip	TRUE	Irr.	Formula annulled ("0")		
15	FALSE	Skip	TRUE	Irr.	Formula annulled ("0")		

T / B = TRUE or Blank

Table E – Combination off all.

This Table is a combination of the above Tables. Combinations with the same outcome are put together in a way that if it does not matter if the Value of **Condition** is *TRUE* or *Blank* the Tables show *T / B*.
The following terms are used in the Table: Blank = *B*, TRUE = *T*, FALSE = *F*. Furthermore "*Irr.*" (irrelevant) is used. Meaning that it is irrelevant what is chosen in **Stop With** and **Go To**

					Outcome		
Line	Condition (Value)	Stop With	Stop Condition (Value)	Go To	Line Self-Included	Stop	Go To
1	Blank	Select	TRUE	Filled	Y	N	N
2	Blank	Skip	FALSE	Filled	Y	N	N
3	Blank	Go To	FALSE	Filled	Y	N	N
4	Blank	Result	TRUE	Filled	Y	N	N
5	T / B	Go To	T / F	0	Y	N	
6	T / B	Skip	TRUE	Irr.		Formula annulled ("0")	
7	T / B	Go To	TRUE	Filled	Y	N	Y
8	T / B	Result	TRUE	0	Y	Y	
9	T / B	Result	TRUE	Filled	Y	Y	N
10	T / B	Select	TRUE	0	Y	N	
11	T / B	Skip	FALSE	0	Y	N	
12	T / B	Result	FALSE	0	Y	N	
13	T / B	Select	FALSE	Irr.		Formula annulled ("0")	
14	TRUE	Select	TRUE	Filled	Y	N	Y
15	TRUE	Skip	FALSE	Filled	Y	N	Y
16	TRUE	Go To	FALSE	Filled	Y	N	Y
17	TRUE	Result	FALSE	Filled	Y	N	Y
18	TRUE	Irr.	Blank	0	Y		
19	TRUE	Irr.	Blank	Filled	Y		Y
20	FALSE	Go To	T / F	0	N	N	
21	FALSE	Skip	FALSE	Irr.		Formula annulled ("0")	
22	FALSE	Go To	TRUE	Filled	N	N	Y
23	FALSE	Result	TRUE	0	N	Y	
24	FALSE	Result	TRUE	Filled	N	Y	N
25	FALSE	Select	TRUE	0	N	N	
26	FALSE	Select	TRUE	Filled	N	N	N
27	FALSE	Skip	FALSE	0	N	N	
28	FALSE	Skip	FALSE	Filled	N	N	N
29	FALSE	Irr.	FALSE	Filled	N	N	N
30	FALSE	Irr.	FALSE	0	N	N	
31	FALSE	Go To	FALSE	Filled	N	N	N
32	FALSE	Go To	FALSE	Irr.		Formula annulled ("0")	
33	FALSE		Blank	0	N		
34	FALSE		Blank	Filled	N		N

T / B = TRUE or Blank, T/F = TRUE or FALSE